



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

Liberty Protocols and Schemas Specification

Draft Version 1.1-06

15th November 2002

Document Description: draft-liberty-architecture-protocols-schemas-v1.1-07

34 **Notice**

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

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

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

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

61 Liberty Alliance Project
62 Licensing Administrator
63 c/o IEEE-ISTO
64 445 Hoes Lane, P.O. Box 1331
65 Piscataway, NJ 08855-1331, USA
66

66 **Editors**

67 John D. Beatty, Sun Microsystems, Inc.
68 John Kemp, IEEE-ISTO

69 **Contributors**

70
71 The following Liberty Alliance Project Sponsor companies contributed to the development of
72 this specification:
73

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

74

75 **Document History**

Revision	Date	Log
00	20-Oct-02	CRs implemented: 1096; 1113; 1116; 1117; 1128; 1129; 1130; 1134; 1135; 1136; 1137; 1141; 1142; 1143; 1149; 1152; 1154; 1146
01	28-Oct-02	<p>CRs implemented: 1161/1160, 1132</p> <p>1161/1160:</p> <ul style="list-style-type: none"> • Section 3.3.1.1 – added RegisterNameIdentifierReturnURL element in text and schema. • Section 3.3.1.2 – updated example to match the above changes • Section 3.3.2.1 – added RegisterNameIdentifierReturnURL element in text and schema. • Section 3.3.2.2 – updated example to match the above changes • Section 4.2 – added new metadata element to the service provider metadata to specify the preferred RegisterNameIdentifier protocol <p>1132:</p> <ul style="list-style-type: none"> • Section 3.2.1.1 – added AuthnContextComparisonType in text and changed

		<p>schema.</p> <ul style="list-style-type: none"> • Section 3.2.2.2 – updated example to match the above changes • Section 3.2.3 – updated text in processing rules to note this change.
02	29-Oct-02	<p>1161:</p> <ul style="list-style-type: none"> • Sections 3.3 – 3.3.3 – updated textual notes to reflect symmetry of RegisterNameIdentifier protocol between providers. • Section 4.2 – added new metadata elements for RegisterNameIdentifierReturnURL and RegisterNameIdentifierProtocolProfile to the generic provider descriptor information. Removed them from Request/Response in 3.3.1/2 <p>1132:</p> <ul style="list-style-type: none"> • Made the AuthnContextComparisonType an enumerated attrib. <p>1158:</p> <ul style="list-style-type: none"> • Section 3.2.4.2 – added paragraphs discussing the logout of the Principal after federation termination. • Section 3.2.3 – added a processing rule stating that an AuthnResponse MUST have a samlp:StatusCode of lib:FederationDoesNotExist if the SP is not currently federated with the IDP. <p>General:</p> <ul style="list-style-type: none"> • Updated namespace calls to /2002/12 for 1.1 namespace(s).
03	30-Oct-02	<p>1153:</p> <ul style="list-style-type: none"> • Updated all examples to have UTC compliant date/time data with a Z at the end of each date/time. <p>1133:</p> <ul style="list-style-type: none"> • Section 3.3.3 - added a line to processing rules noting that messages SHOULD be signed, and noted that the signature MUST be verified if present. • Section 3.2.3 added a line to processing rules, noting that an error should be returned if authentication fails. • Section 3.4 - added a line to processing rules noting that messages SHOULD be signed, and noted that the signature MUST be verified if present. • Section 3.5.2.1 – modified the processing rules to note that the signature SHOULD be verified, if present. <p>1146:</p> <ul style="list-style-type: none"> • Slightly revised language in lines 965-968 in line with John B’s comment about tightening language. <p>1152:</p> <ul style="list-style-type: none"> • Section 4.2.1 - corrected example URIs for all profiles <p>General:</p> <ul style="list-style-type: none"> • Section 3.1.6 - added a reference for W3 XMLSchema dateTime • Fixed a spelling mistake ‘occurred’ line 654 • Added references to XML Signature and Canonicalization schema documents
04	31-Oct-02	<p>1158:</p>

		<ul style="list-style-type: none"> Section 3.4.2 – modified language to state that a provider MAY invalidate a user’s session as the result of federation termination. <p>1161:</p> <ul style="list-style-type: none"> Section 3.3.1.1 – added a new element <OldProvidedNameIdentifier> to the RegisterNameIdentifierRequest message. Updated examples, and schema to match Section 3.3 – added text to note that the <OldProvidedNameIdentifier> should contain the previous version of the <XXXProvidedNameIdentifier>
04	01-Nov-02	<p>1161:</p> <ul style="list-style-type: none"> Section 3.2.2.4 – added paragraphs to specify that saml:AuthenticationMethod should be populated with the Liberty context URI, when any AuthnContext is specified. Also noted that the SP must look to the Liberty AuthnContext (and ignore the saml attribute) if the saml:AuthenticationMethod is equal to the Liberty context URI. Section 3.3 – added note to the effect that implementations may want to take account of possible propagation delays in processing of name identifier changes.1164: <p>General:</p> <ul style="list-style-type: none"> Section 4.1 – removed a reference to profiles in the text Section 3.2.3 – slightly tightened language about processing of AuthnContextComparisonType rules All sections – removed CR comments <p>Examples:</p> <ul style="list-style-type: none"> Updated AuthnRequest example
05	05-Nov-02	General edits for spelling, grammar, formatting.
06	15-Nov-02	<p>1216: section 3.2.2.2 changed Assertion to AssertionType</p> <p>1215: changed SAML reference to point to SAML 1.0 document</p> <p>1214: corrected erroneous URIs lines 1186-1190</p> <p>1213: corrected description of SingleSignOnProtocolProfile</p> <p>1209: removed line 1010 regarding federation termination</p> <p>1208: removed duplicate signing requirement</p> <p>1207: saml:Success is now samlp:Success (line 908)</p> <p>1205: the name identifier that the service provider should use when communicating with the identity provider.</p> <p>1204: removed idp:Foo element from example</p> <p>1203: added ‘supported’ regarding lib:NoAvailableIDP (lines 689-690)</p> <p>1202: removed indentation</p> <p>1201: reworded language regarding AuthnContextComparisonType (622-633)</p> <p>1199: corrected URLs in examples</p> <p>1198: created reference from ReauthenticateOnOrAfter to prior section discussing it</p> <p>1197: removed incorrect indentation</p> <p>1196: corrected examples</p>

		<p>1194: corrected typo – placed instead of place (366)</p> <p>1192: reworded lines 283-285 to better deal with the specification of sub-second values.</p> <p>1191: added further requirement on UTC time + Z specified lines 281-282</p> <p>1190: reworded to be grammatically correct (256)</p> <p>1189: added the word ‘Individual’ line 252</p> <p>1188: Some profiles of the protocols contained in this specification may require a succinct 20-byte identifier. A provider MUST derive this by generating the SHA-1 hash of its URI-based identifier.</p> <p>1187: removed section number reference (226)</p> <p>1186: reworded lines 170-172.</p> <p>1185: corrected typos line 424, 428</p> <p>1184: corrected inconsistency in location of signing requirements</p> <p>1212: corrected differences between RegisterNameIdentifier and other protocol profiles</p> <p>1210: added RegisterNameIdentifierServiceURL, and corrected RegisterNameIdentifierServiceReturnURL</p>
--	--	--

76 **Table of Contents**

77	1	Introduction	8
78	1.1	Notation	8
79	1.2	Overview	9
80	2	Schema Declarations	9
81	2.1	Schema Header and Namespace Declarations.....	9
82	2.2	Type and Element Declarations.....	9
83	3	Protocols.....	10
84	3.1	General Requirements	10
85	3.1.1	XML Signature	10
86	3.1.2	Protocol and Assertion Versioning	10
87	3.1.3	Provider ID Uniqueness.....	10
88	3.1.4	Name Identifier Construction	10
89	3.1.5	Signature Verification.....	11
90	3.1.6	Security	11
91	3.1.7	Time Synchronization.....	11
92	3.2	Single Sign-On and Federation Protocol.....	12
93	3.2.1	Request.....	12
94	3.2.2	Response	15
95	3.2.3	Processing Rules	17
96	3.2.4	Request Envelope.....	20
97	3.2.5	Response Envelope	22
98	3.3	Name Registration Protocol.....	23
99	3.3.1	Request.....	24
100	3.3.2	Response	25
101	3.3.3	Processing Rules	25
102	3.4	Federation Termination Notification Protocol	26
103	3.4.1	Message.....	26
104	3.4.2	Processing Rules	27
105	3.5	Single Logout Protocol.....	27
106	3.5.1	Message.....	27
107	3.5.2	Processing Rules	28
108	4	Provider Metadata Schema.....	29
109	4.1	Generic Provider Descriptor	29
110	4.2	Service Provider Descriptor.....	30
111	4.2.1	Example	31
112	4.3	Identity Provider Descriptor	31
113	4.3.1	Example	31
114	5	Schema Definition.....	32
115	6	References	35
116			
117			

117 1 Introduction

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

121 1.1 Notation

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

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

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

135 Listings of schemas appear like this.

136
137 Listings of instance fragments appear like this.

138
139 The following namespaces are referred to in this document:

- 140 • The prefix `lib:` stands for the Liberty namespace
141 (<http://projectliberty.org/schemas/core/2002/12>). This namespace is the
142 default for instance fragments, type names, and element names in this document.
- 143 • The prefix `ac:` stands for the Liberty authentication context namespace
144 (<http://projectliberty.org/schemas/authctx/2002/05>)
- 145 • The prefix `saml:` stands for the SAML assertion namespace
146 (<urn:oasis:names:tc:SAML:1.0:assertion>).
- 147 • The prefix `samlp:` stands for the SAML protocol namespace
148 (<urn:oasis:names:tc:SAML:1.0:protocol>).
- 149 • The prefix `ds:` stands for the W3C XML signature namespace
150 (<http://www.w3.org/2000/09/xmldsig#>).
- 151 • The prefix `xsd:` stands for the W3C XML schema namespace
152 (<http://www.w3.org/2001/XMLSchema>). In schema listings, this is the default
153 namespace and no prefix is shown.
- 154 • The prefix `xsi:` stands for the W3C XML schema instance namespace
155 (<http://www.w3.org/2001/XMLSchema-instance>).

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

158 For readability, when an XML Schema type is specified to be `xsd:boolean`, this document
159 discusses the values as “true” and “false” rather than the “1” and “0” which are also legal
160 `xsd:boolean` values.

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

162 1.2 Overview

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

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

169 Once the Principal is *authenticated* to the identity provider, the identity provider can provide an
170 authentication assertion to the Principal, who can present the assertion to the service provider. The
171 Principal is then also authenticated to the service provider if the service provider trusts the assertion.
172 An *identity federation* is said to exist between an identity provider and a service provider when the
173 service provider accepts authentication assertions regarding a particular Principal from the identity
174 provider. This specification defines a protocol where the identity of the Principal can be *federated*
175 between the identity provider and the service provider.

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

179 2 Schema Declarations

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

182 2.1 Schema Header and Namespace Declarations

183 The following schema fragment defines the XML namespaces and other header information for the
184 Liberty schema:

```
185 <schema targetNamespace="http://projectliberty.org/schemas/core/2002/12"  
186 xmlns:lib="http://projectliberty.org/schemas/core/2002/12"  
187 xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"  
188 xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"  
189 xmlns:ac="http://projectliberty.org/schemas/authctx/2002/05" xmlns="http://www.w3.org/2001/XMLSchema"  
190 elementFormDefault="qualified" attributeFormDefault="unqualified">  
191 <import namespace="urn:oasis:names:tc:SAML:1.0:assertion" schemaLocation="http://www.oasis-  
192 open.org/committees/security/docs/draft-sstc-schema-assertion-31.xsd"/>  
193 <import namespace="urn:oasis:names:tc:SAML:1.0:protocol" schemaLocation="http://www.oasis-  
194 open.org/committees/security/docs/draft-sstc-schema-protocol-31.xsd"/>  
195 <import namespace="http://www.w3.org/2000/09/xmldsig#"  
196 schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>  
197 <import namespace="http://projectliberty.org/schemas/authctx/2002/" schemaLocation="draft-  
198 liberty-architecture-authentication-context-06.xsd"/>
```

199 2.2 Type and Element Declarations

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

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

202 3 Protocols

203 The Liberty protocol suite consists of the following protocols:

- 204 • Single Sign-On and Federation: The protocol by which identities are federated and by which
205 single sign-on occurs.
- 206 • Name Registration: The protocol by which a service provider can register an alternative
207 opaque handle (or *name identifier*) for a Principal.
- 208 • Federation Termination Notification: The protocol by which a provider can notify another
209 provider than a particular identity federation has been terminated (also known as de-
210 federation).
- 211 • Single Logout: The protocol by which providers notify each other of logout events for single
212 logout functionality.

213 3.1 General Requirements

214

215 3.1.1 XML Signature

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

219 3.1.2 Protocol and Assertion Versioning

220 Version information appears in protocol messages and assertions defined in this specification. This
221 specification defines version 1.0 for the protocol messages and assertions. Version numbering of
222 assertions is independent of the version numbering of the protocol messages.

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

225 3.1.3 Provider ID Uniqueness

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

230 Some profiles of the protocols contained in this specification may require a succinct 20-byte
231 identifier. A provider MUST derive this by generating the SHA-1 hash of its URI-based identifier.

232 3.1.4 Name Identifier Construction

233 Principals are assigned name identifiers by identity providers and potentially by service providers.
234 When generated by the identity provider, a name identifier MUST be constructed using pseudo-
235 random values that have no discernable correspondence with the Principal’s identifier (e.g.,
236 username) at the identity provider. The intent is to create a non-public pseudonym so as to
237 contravene the linkability of the Principal’s identity or activities. Service providers SHOULD follow
238 the same construction rules. Name identifier values MUST NOT exceed 256 characters.

239 **3.1.5 Signature Verification**

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

246 XML signatures SHOULD NOT be performed with any transforms other than:

- 247 • Enveloped Signature [[XMLDsig](#)]
- 248 • Exclusive XML Canonicalization [[XMLCanon](#)]

249 Receivers MUST NOT accept XML signatures created using other transforms without verification
250 that the transforms used do not omit any part of the data to be signed from the signed byte stream.
251 Receivers MAY reject any messages with transforms other than the set specified above. Senders
252 MUST NOT send messages using other transforms without prior agreement as to their contents.

253 XML signatures in messages MUST use a proper URI fragment in the URI attribute of the Reference
254 element to identify the signed element. The signer MUST NOT assume that the signed element will
255 be at the root of the document during verification. It MUST be possible to validate the signature after
256 adding or removing surrounding context for the profile in use (for example, the SOAP envelope, or
257 the `<samlp:Response>` element). Implementers are encouraged to verify compliance with this
258 requirement via empirical testing.

259 **3.1.6 Security**

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

266 **3.1.7 Time Values**

267 All Liberty time values have the type `dateTime`, which is built in to the W3C XML Schema
268 Datatypes specification [[Schema2](#)]. Liberty time values MUST be expressed in UTC form, indicated
269 by a “Z” immediately following the time portion of the value..

270 Liberty Requestors and Responders SHOULD NOT rely on other applications supporting time
271 resolution finer than seconds, as implementations MAY ignore sub-second components specified in
272 timestamp values. Implementations MUST NOT generate time instants that specify leap seconds.

273 **3.1.8 Time Synchronization**

274 Providers SHOULD NOT assume that other providers have clocks that are synchronized closer than
275 one minute.

276 The Identity Provider SHOULD NOT include a `NotBefore` attribute on the `Conditions` element of the
277 assertion it generates which contains the time the assertion was generated.

278 The Identity Provider SHOULD NOT include a NotOnOrAfter attribute on the Conditions element
279 of the assertion it generates which is less than one minute later than the time when the assertion was
280 generated.

281 The Service Provider SHOULD NOT terminate the principal's session based solely on the
282 NotOnOrAfter attribute of the Conditions element of the assertion used to authenticate the principal.
283 If the assertion was valid when the principal was authenticated, the principal SHOULD remain
284 authenticated until one of the following occurs:

- 285 • LogoutNotification is received
- 286 • The user's session times out via normal means
- 287 • The ReauthenticateOnOrAfter time on the AuthenticationStatement used to authenticate the
288 principal, if any, is reached

289 **3.2 Single Sign-On and Federation Protocol**

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

- 292 1. A service provider issues an `<AuthnRequest>` request to an identity provider, instructing
293 the identity provider to provide an authentication assertion to the service provider.
294 Optionally, the service provider MAY request that the identity be federated.
- 295 2. The identity provider responds with either an `<AuthnResponse>` containing authentication
296 assertions to the service provider or an artifact that can be de-referenced into an
297 authentication assertion. Additionally, the identity provider potentially federates the
298 Principal's identity at the identity provider with the Principal's identity at the service
299 provider.

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

307 **3.2.1 Request**

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

- 312 • Prompt the Principal for credentials if the Principal is not presently authenticated.
- 313 • Prompt the Principal for credentials, even if the Principal is presently authenticated.
- 314 • Federate the Principal's identity at the identity provider with the Principal's identity at the
315 service provider.
- 316 • Use a specific protocol profile in responding to the request.

- 317 • Use a specific authentication context (for example, smartcard-based authentication vs.
318 username/password-based authentication).

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

321 The <AuthnRequest> message SHOULD be signed.

322 **3.2.1.1 Element <AuthnRequest>**

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

326 The elements of the request are as follows:

327 ProviderID [Required]

328 The service provider's URI-based identifier.

329 IsPassive [Optional]

330 If "true," specifies that the identity provider MUST NOT interact with the Principal and
331 MUST NOT take control of the user interface from the service provider. If "false," the
332 identity provider MAY interact with the user and MAY temporarily take control of the user
333 interface for that purpose. If not specified, "true" is presumed.

334 ForceAuthn [Optional]

335 Controls whether the identity provider authenticates the Principal regardless of whether the
336 Principal is already authenticated. This element is specified only when <IsPassive> is
337 "false." If <ForceAuthn> is "true," specifies that the identity provider MUST always
338 authenticate the Principal, regardless of whether the Principal is presently authenticated. If
339 "false," specifies that the identity provider MUST re-authenticate the user only if the
340 Principal is not presently authenticated. If not specified, "false" is presumed.

341 Federate [Optional]

342 Specifies that the service provider wishes to federate the Principal's identity at the service
343 provider with the Principal's identity at the identity provider. If the element is not specified, it
344 is presumed that the service provider does not wish to federate the identity.

345 ProtocolProfile [Optional]

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

349 AuthnContext [Optional]

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

352 RelayState [Optional]

353 This contains state information that will be relayed back in the response. This data SHOULD
354 be integrity-protected by the request author and MAY have other protections placed on it by
355 the request author. An example of such protection is confidentiality.

356 id [Optional]

357 Identifier used to identify this element in the signature. See section 3.1.5, Signature Verification for
358 more information.

359 AuthnContextComparisonType [Optional]

360 If set to “exact”, then the identity provider is asked to match at least one of the specified
361 <AuthnContext> elements exactly. This can also be set to “minimum”, which asks that the
362 identity provider use a context that he feels is at least as good as any specified in the
363 <AuthnContext> or “better”, which means that the they can use any context better than any
364 that were supplied. If not specified, this is assumed to be “exact”.

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

366 AuthnContextClassRef [Optional]

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

369 AuthnContextStatementRef [Optional]

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

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

```
373 <element name="AuthnRequest" type="lib:AuthnRequestType"/>  
374 <complexType name="AuthnRequestType">  
375 <complexContent>  
376 <extension base="samlp:RequestAbstractType">  
377 <sequence>  
378 <element ref="lib:ProviderID"/>  
379 <element name="ForceAuthn" type="boolean" minOccurs="0"/>  
380 <element name="IsPassive" type="boolean" minOccurs="0"/>  
381 <element name="Federate" type="boolean" minOccurs="0"/>  
382 <element ref="lib:ProtocolProfile" minOccurs="0"/>  
383 <element ref="lib:AuthnContext" minOccurs="0"/>  
384 <element ref="lib:RelayState" minOccurs="0"/>  
385 <element ref="lib:AuthnContextComparisonType" minOccurs="0"/>  
386 </sequence>  
387 <attribute name="id" type="ID" use="optional"/>  
388 </extension>  
389 </complexContent>  
390 </complexType>  
391 <simpleType name="AuthnContextComparisonType">  
392 <restriction base="string">  
393 <enumeration value="exact"/>  
394 <enumeration value="minimum"/>  
395 <enumeration value="better"/>  
396 </restriction>  
397 </simpleType>  
398 <element name="RelayState"/>  
399 <element name="ProtocolProfile" type="anyURI"/>  
400 <element name="AuthnContext">  
401 <complexType>  
402 <choice>  
403 <element name="AuthnContextClassRef" type="anyURI" maxOccurs="unbounded"/>  
404 <element name="AuthnContextStatementRef" type="anyURI" maxOccurs="unbounded"/>  
405 </choice>  
406 </complexType>  
407 </element>
```

409 3.2.1.2 Example

```
410 <lib:AuthnRequest id="12345" RequestID="RPCUk211+GVz+t1lLURp51oFvJXk" MajorVersion="1"  
411 MinorVersion="0" IssueInstant="2001-12-17T21:42:4Z"  
412 xmlns:lib="http://projectliberty.org/schemas/core/2002/12">  
413 <ds:Signature> ... </ds:Signature>  
414 <lib:ProviderID>http://ServiceProvider.com</lib:ProviderID>  
415 <lib:ForceAuthn>>false</lib:ForceAuthn>
```

```
416 <lib:IsPassive>>false</lib:IsPassive>
417 <lib:Federate>>true</lib:Federate>
418 <lib:ProtocolProfile>http://projectliberty.org/profiles/brws-post</lib:ProtocolProfile>
419 <lib:AuthnContext>
420 <lib:AuthnContextClassRef>http://projectliberty.org/schemas/authctx/classes/Password-
421 ProtectedTransport</lib:AuthnContextClassRef>
422 </lib:AuthnContext>
423 <lib:RelayState>R01GODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</lib:RelayState>
424 <lib:AuthnContextComparisonType>exact</lib:AuthnContextComparisonType>
425 </lib:AuthnRequest>
```

426 3.2.2 Response

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

429 All authentication assertions generated by an identity provider for a service provider MUST be of
430 type **AssertionType**. The `<saml:Subject>` element in any subject statement MUST be of type
431 **SubjectType**. If the service provider registered a name identifier for the Principal (see 3.3), the
432 `<saml:NameIdentifier>` element in the `<saml:Subject>` element MUST be the service
433 provider-provided name identifier for the Principal. Otherwise, `<saml:NameIdentifier>` MUST
434 be the name identifier provided by the identity provider at the time of federation. The
435 `<IDPProvidedNameIdentifier>` MUST contain the name identifier provided by the identity
436 provider at the time of federation.

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

438 Identity providers MUST include a `<saml:AudienceRestrictionCondition>` element that
439 specifies the intended consumers of the assertion. The `<saml:Audience>` element MUST be set to
440 the intended recipient's ProviderID. The recipient MUST validate that it is the intended viewer
441 before using the assertion.

442 Identity providers MAY include a `SessionIndex` attribute in resulting authentication statements,
443 which is used to aid the identity provider in managing multiple sessions with the Principal. If the
444 identity provider includes this `SessionIndex` attribute, subsequent messages from the service
445 provider to the identity provider that are session-dependent MUST include this `SessionIndex`
446 attribute.

447 All assertions in the `<AuthnResponse>` MUST be individually signed by the identity provider (that
448 is, each assertion must contain a `Signature` element which signs only the assertion). It is
449 RECOMMENDED that the signature be omitted from the `<AuthnResponse>` itself, but this is not
450 forbidden.

451 3.2.2.1 Element `<AuthnResponse>`

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

```
454 <element name="AuthnResponse" type="lib:AuthnResponseType"/>
455 <complexType name="AuthnResponseType">
456 <complexContent>
457 <extension base="samlp:ResponseType">
458 <sequence>
459 <element ref="lib:RelayState" minOccurs="0"/>
460 </sequence>
461 </extension>
462 </complexContent>
463 </complexType>
```

464 3.2.2.2 Element <AssertionType>

465 Assertions provided in an <AuthnResponse> element MUST be of type **AssertionType**, which is
466 an extension of **saml:AssertionType**, so that the RequestID attribute from the original
467 <AuthnRequest> is included in the InResponseTo attribute in the <Assertion> element. This
468 step is done because the <AuthnResponse> element is not required to be signed, whereas the
469 contained <Assertion> elements are required to be signed. The id attribute is also included to
470 facilitate signatures (see section 3.1.5, Signature Verification).

471 The schema fragment is as follows:

```
472 <complexType name="AssertionType">  
473 <complexContent>  
474 <extension base="saml:AssertionType">  
475 <attribute name="InResponseTo" type="saml:IDReferenceType"/>  
476 <attribute name="id" type="ID" use="optional"/>  
477 </extension>  
478 </complexContent>  
479 </complexType>
```

480 3.2.2.3 Type SubjectType

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

```
484 <complexType name="SubjectType">  
485 <complexContent>  
486 <extension base="saml:SubjectType">  
487 <sequence>  
488 <element ref="lib:IDPProvidedNameIdentifier"/>  
489 </sequence>  
490 </extension>  
491 </complexContent>  
492 </complexType>
```

493 3.2.2.4 Type AuthenticationStatementType

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

496 AuthnContext [Optional]

497 The authentication context that the identity provider used in the authentication event that
498 yielded this authentication statement. Contains either an authentication context statement or a
499 reference to an authentication context statement. Optionally contains a reference to an
500 authentication context class.

501 ReauthenticateOnOrAfter [Optional]

502 The time at, or after which the service provider reauthenticates the Principal with the identity
503 provider (as required in [Section 3.2](#) above).

504 SessionIndex [Optional]

505 Indexes the particular session between the Principal and the identity provider under which this
506 authentication statement is being issued. This value SHOULD be a small, positive integer but may be
507 any string of text. However, this value MUST NOT be a globally unique value for the Principal's
508 session at the Identity Provider.

509 When an <AuthnContext> element is specified, the **saml:AuthenticationMethod** attribute on the
510 <saml:AuthenticationStatement> MUST be
511 "http://projectliberty.org/schemas/authctx/2002/05".

512 When the Service Provider is processing a <saml:AuthenticationStatement> of type
513 **lib:AuthenticationStatementType** and the saml:AuthenticationMethod attribute is
514 "http://projectliberty.org/schemas/authctx/2002/05", the Service Provider MUST refer to the
515 <AuthnContext> element and ignore the saml:AuthenticationMethod attribute.

516

517 The schema fragment is as follows:

```
518 <complexType name="AuthenticationStatementType">
519   <complexContent>
520     <extension base="saml:AuthenticationStatementType">
521       <sequence>
522         <element name="AuthnContext" minOccurs="0">
523           <complexType>
524             <sequence>
525               <element name="AuthnContextClassRef" type="anyURI" minOccurs="0"/>
526               <choice>
527                 <element ref="ac:AuthenticationContextStatement"/>
528                 <element name="AuthnContextStatementRef" type="anyURI"/>
529               </choice>
530             </sequence>
531           </complexType>
532         </element>
533       </sequence>
534       <attribute name="ReauthenticateOnOrAfter" type="dateTime" use="optional"/>
535       <attribute name="SessionIndex" type="string" use="optional"/>
536     </extension>
537   </complexContent>
538 </complexType>
```

539 3.2.2.5 Example

```
540 <lib:AuthnResponse ResponseID="9hhuujalbc744hGJn5Q9A5yvEIgS"
541 InResponseTo="Zon3WjJ2KL7j+bJu7MuIr4Pt2go5" MajorVersion="1" MinorVersion="0" IssueInstant="2002-10-
542 31T21:55:41Z">
543   <samlp:Status>
544     <samlp:StatusCode Value="samlp:Success"/>
545   </samlp:Status>
546   <saml:Assertion id="12345" MajorVersion="1" MinorVersion="0" AssertionID="e06e5a28-bc80-4ba6-
547 9ecb-712949db686e" Issuer="http://IdentityProvider.com" IssueInstant="2001-12-17T09:30:47Z"
548 InResponseTo="4e7c3772-4fa4-4a0f-99e8-7d719ff6067c" xsi:type="AssertionType">
549     <saml:Conditions NotBefore="2001-12-17T09:30:47Z" NotOnOrAfter="2001-12-17T09:35:47Z">
550       <saml:AudienceRestrictionCondition>
551         <saml:Audience>http://ServiceProvider.com</saml:Audience>
552       </saml:AudienceRestrictionCondition>
553     </saml:Conditions>
554     <saml:AuthenticationStatement AuthenticationInstant="2001-12-17T09:30:47Z" SessionIndex="3"
555 ReauthenticateOnOrAfter="2001-12-17T11:30:47Z" xsi:type="AuthenticationStatementType"
556 AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
557       <saml:Subject xsi:type="SubjectType">
558         <saml:NameIdentifier>342ad3d8-93ee-4c68-be35-cc9e7db39e2b</saml:NameIdentifier>
559         <IDPProvidedNameIdentifier>342ad3d8-93ee-4c68-be35-
560 cc9e7db39e2b</IDPProvidedNameIdentifier>
561       </saml:Subject>
562     </saml:AuthenticationStatement>
563     <ds:Signature>...</ds:Signature>
564   </saml:Assertion>
565   <RelayState>R0LGODlhcgGSALMAAAQCAEMmCZtuMFQxDS8b</RelayState>
566 </AuthnResponse>
```

567 3.2.3 Processing Rules

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

- 570 • The <ProviderID> MUST be the Provider ID of a known service provider with which the
571 identity provider has established a relationship. The <ProviderID> MUST be resolvable
572 to an assertion consumer service URL at the service provider that the identity provider may
573 use when returning the corresponding assertion reference.

- 574 • <ds:Signature>, if present, MUST be the signature of the service provider as specified by
575 the <ProviderID>.
- 576 • If <IsPassive> is “true,” the identity provider MUST NOT interact with the Principal and
577 MUST NOT take control of the user interface (if applicable).
- 578 • The identity provider MUST attempt to authenticate the Principal if <ForceAuthn> is
579 “true,” regardless of whether the Principal is presently authenticated, unless <IsPassive>
580 is “true.”
- 581 • Failure to authenticate the Principal is indicated by a status code other than “Success.” For
582 failures, assertions MUST NOT appear in the <AuthnResponse>.
- 583 • The identity provider MAY federate the Principal’s identity at the service provider with the
584 user’s identity at the identity provider if <Federate> is “true” and the Principal has
585 consented for such an action to occur. The identity provider MUST NOT federate if
586 <Federate> is “false.” If <Federate> is "true" but the identity provider already has a
587 previous federation on record for the Principal's identity at the service provider, the identity
588 provider SHOULD treat the request as it would have if <Federate> was "false".
- 589 • The identity provider MUST respond using the specified <ProtocolProfile>.
- 590 • If <RelayState> contains a value, the identity provider MUST include this value in
591 unmodified form in the <RelayState> element of the returned authentication assertion.
- 592 • The InResponseTo attribute in all generated <Assertion> elements in the
593 <AuthnResponse> element MUST be set to the value of the RequestID attribute in the
594 corresponding <AuthnRequest> element.
- 595 • If the service provider is not federated with the identity provider, and the <Federate> flag
596 is not set, then the identity provider MUST return a <samlp:StatusCode> of
597 lib:FederationDoesNotExist indicating that the service provider is not federated with
598 this identity provider when the identity provider .
- 599 • If <AuthnContextComparisonType> is specified and set to “exact”, then the resulting
600 authentication statement in the assertion (if any) MUST be the exact match of at least one of
601 the authorization contexts specified.
- 602 • If <AuthnContextComparisonType> is specified and set to “minimum”, then the
603 resulting authentication statement in the assertion (if any) MUST be at least as strong (as
604 deemed by the identity provider) as one of the authorization contexts specified.
- 605 • If <AuthnContextComparisonType> is specified and set to “better”, then the resulting
606 authentication statement in the assertion (if any) MUST be stronger (as deemed by the
607 identity provider) than any specified in the supplied authorization contexts.
- 608 Additionally, if the <AuthnContext> element is specified, the identity provider MUST authenticate
609 the Principal according to the following rules:
- 610 • If one or more <AuthnContextClassRef> elements are included, then the resulting
611 authentication statement in the assertion (if any) MUST contain an authentication statement
612 that conforms to one of the specified classes. Additionally, the set of
613 <AuthnContextClassRef> elements MUST be evaluated as an ordered set, where the
614 first element is the most preferred authentication context class. If none of the specified

615 authentication context classes can be satisfied, the identity provider MUST not include an
616 authentication statement in the resulting assertion.

- 617 • If one or more `<AuthnContextStatementRef>` elements are included, then the resulting
618 authentication statement in the assertion (if any) MUST follow the rules specified in the
619 `<AuthnContextComparisonType>`. If this requirement cannot be satisfied, the identity
620 provider MUST not include an authentication statement in the resulting assertion.

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

- 623 • The name identifier MUST be unique across all Principals in the scope of the service
624 provider-identity provider pairwise relationship.
- 625 • The name identifier for the specific Principal MUST be unique across all service providers
626 with which an identity federation exists with the identity provider.

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

629 If the service provider attempts to federate a Principal's identity with an identity provider, but another
630 Principal's identity at the same service provider is already federated with the same identity provider,
631 it will receive the other Principal's established name identifier in the `<AuthnResponse>`, rather than
632 a new random one. The service provider MUST detect this error and handle it appropriately without
633 leaving either Principal's identity at the service provider in an unusable state.

634 In some profiles, an intermediary is active between the service provider's authentication request and
635 the identity provider's authentication response. Examples of an active intermediary are a user agent
636 or client proxy that implement the "Liberty-Enabled Client and Proxy Profile" described in
637 [[LibertyBindProf](#)].

638 NOTE: an active intermediary has the capability to return status codes to the service provider it
639 interacts with. For example, the intermediary may be unable to contact an identity provider identified
640 by the service provider, and the intermediary may return a status code to the service provider
641 indicating that an error occurred. Status codes MUST be conveyed within `<AuthnResponse>`
642 messages using the `<samlp:Status>` element. Specific values for specific cases are defined below.

643 For all profiles specifying an active intermediary, the profile specification must:

- 644 • Specify whether the `<AuthnRequest>` element sent from the service provider to the
645 identity provider via the intermediary is wrapped in an `<AuthnRequestEnvelope>`. See
646 section 3.2.4.
- 647 • Specify whether the `<AuthnResponse>` element sent from the identity provider to the
648 service provider via the intermediary is wrapped in an `<AuthnResponseEnvelope>`. See
649 section 3.2.5.

650 3.2.3.1.1 Processing Rules for Active Intermediaries

651 For all profiles specifying an active intermediary, the intermediary MUST follow these processing
652 rules:

- 653 • If the profile specifies that the message sent from the service provider to the identity
654 provider, via the intermediary, is wrapped in an `<AuthnRequestEnvelope>`:
- 655 • The intermediary MUST remove the enveloping `<AuthnRequestEnvelope>` before
656 forwarding the `<AuthnRequest>` element to the identity provider.

- 657 • The intermediary MAY locally generate <AuthnResponse> elements and send them to the
658 service provider using the <AssertionConsumerServiceURL> contained within the
659 <AuthnRequestEnvelope>. Such <AuthnResponse> elements MUST NOT contain any
660 <lib:Assertion> elements. These messages MAY be generated as a result of local errors
661 on the intermediary, and should indicate the underlying reasons in the <samlp:Status>
662 element in the <AuthnResponse>.

663 The following are error conditions for which <samlp:Status> values are defined in section
664 3.2.3.1.2:

- 665 • The identity provider cannot be reached
- 666 • There is no identity provider in common between the intermediary and the service provider
- 667 • If the profile specifies that the message from the identity provider to the service provider, via
668 the intermediary, is wrapped in an <AuthnResponseEnvelope>:
- 669 • The intermediary MUST remove the enveloping <AuthnResponseEnvelope> before
670 forwarding the <AuthnResponse> element to the service provider.
- 671 • The intermediary MUST send <AuthnResponse> messages received from the identity
672 provider to the service provider using the <AssertionConsumerServiceURL> contained
673 within the <AuthnResponseEnvelope> sent by the identity provider.

674 **3.2.3.1.2 Status Code Values for Error Conditions**

675 If an error occurs in the processing at the intermediary, the following status code values are defined
676 for the <samlp:Status> element:

- 677 • lib:NoAvailableIDP: Used to indicate that none of the supported identity provider URLs
678 from the <IDPList> can be resolved or that none of the identity providers is available.
- 679 • lib:NoSupportedIDP: Used to indicate that none of the identity providers is supported by
680 the intermediary.

681 **3.2.4 Request Envelope**

682 Some profiles MAY wrap the <AuthnRequest> element in an envelope. This envelope allows for
683 extra processing by an intermediary between the service provider and the identity provider. An
684 example of an intermediary is a user agent or proxy. Processing rules are given in section 3.2.3.1.1.
685 Note that the envelope is for consumption by the intermediary and is removed before the enveloped
686 <AuthnRequest> element is forwarded to the identity provider.

687 **3.2.4.1 Element <AuthnRequestEnvelope>**

688 The authentication request envelope contains the following elements:

689 AuthnRequest [Required]

690 The authentication request contained in the envelope.

691 ProviderID [Required]

692 The ProviderID of the requestor.

693 ProviderName [Optional]

694 The human-readable name of the requestor.

695 AssertionConsumerServiceURL [Required]

696 The service provider's URL specifying where <AuthnResponse> elements, locally
697 generated by the intermediary, are to be sent. See the processing rules for active
698 intermediaries specified in section 3.2.3.1.1.

699 IDPList [Optional]

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

702 IsPassive [Optional]

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

705 The schema fragment is as follows:

```
706 <element name="AuthnRequestEnvelope" type="lib:AuthnRequestEnvelopeType"/>
707 <complexType name="AuthnRequestEnvelopeType">
708   <complexContent>
709     <extension base="lib:RequestEnvelopeType">
710       <sequence>
711         <element ref="lib:AuthnRequest"/>
712         <element ref="lib:ProviderID"/>
713         <element name="ProviderName" type="string" minOccurs="0"/>
714         <element name="AssertionConsumerServiceURL" type="anyURI"/>
715         <element ref="lib:IDPList" minOccurs="0"/>
716         <element name="IsPassive" type="boolean" minOccurs="0"/>
717       </sequence>
718     </extension>
719   </complexContent>
720 </complexType>
721 <complexType name="RequestEnvelopeType">
722   <sequence>
723     <any processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
724   </sequence>
725 </complexType>
```

726 3.2.4.2 Element <IDPList>

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

730 IDPList

731 The container element for an IDP List.

732 IDPEntries

733 Contains a list of identity provider entries.

734 IDPEntry

735 Describes an identity provider that the service provider supports.

736 ProviderID

737 The identity provider's ProviderID.

738 ProviderName

739 The identity provider's human-readable name.

740 Loc

741 The identity provider's URI, to which authentication requests may be sent.

742 GetComplete

743 If the identity provider list is not complete, this element is included with a URI that points to
744 where the complete list can be retrieved.

745 The schema fragment is as follows:

```
746 <element name="IDPList" type="lib:IDPListType"/>
747 <complexType name="IDPListType">
748   <sequence>
749     <element ref="lib:IDPEntries"/>
750     <element ref="lib:GetComplete" minOccurs="0" maxOccurs="unbounded"/>
751   </sequence>
752 </complexType>
753 <element name="IDPEntry">
754   <complexType>
755     <sequence>
756       <element ref="lib:ProviderID"/>
757       <element name="ProviderName" type="string" minOccurs="0"/>
758       <element name="Loc" type="anyURI"/>
759     </sequence>
760   </complexType>
761 </element>
762 <element name="IDPEntries">
763   <complexType>
764     <sequence>
765       <element ref="lib:IDPEntry" maxOccurs="unbounded"/>
766     </sequence>
767   </complexType>
768 </element>
769 <element name="GetComplete" type="anyURI"/>
```

770 3.2.4.3 Example

```
771 <AuthnRequestEnvelope>
772   <AuthnRequest> ... </AuthnRequest>
773   <ProviderID>http://ServiceProvider.com</ProviderID>
774   <ProviderName>Service Provider X</ProviderName>
775   <AssertionConsumerServiceURL>http://ServiceProvider.com/lecp assertion consumer</AssertionConsum
776   rServiceURL>
777   <IDPList>
778     <IDPEntries>
779       <IDPEntry>
780         <ProviderID>http://IdentityProvider.com</ProviderID>
781         <ProviderName>Identity Provider X</ProviderName>
782         <Loc>http://www.IdentityProvider.com/liberty/sso</Loc>
783       </IDPEntry>
784     </IDPEntries>
785     <GetComplete>https://ServiceProvider.com/idplist?id=604bel36-fe91-441e-afb8-f88748ae3b8b
786   </GetComplete>
787   </IDPList>
788   <IsPassive>0</IsPassive>
789 </AuthnRequestEnvelope>
```

790 3.2.5 Response Envelope

791 As with the <AuthnRequest> element, some profiles MAY wrap the <AuthnResponse> element
792 in an envelope. This envelope allows for extra processing by an intermediary between the identity
793 provider and the service provider. An example of an intermediary is a user agent or proxy.
794 Processing rules are given in section 3.2.3.1.1. Note that the envelope is for consumption by the
795 intermediary and is removed before the enveloped <AuthnResponse> element is forwarded to the
796 service provider.

797 3.2.5.1 Element <AuthnResponseEnvelope>

798 The authentication response envelope contains the following elements:

799 AuthnResponse [Required]

800 The enveloped authentication response.

801 AssertionConsumerServiceURL [Required]

802 The service provider's URL where the authentication response should be sent. This element's
803 value SHOULD be obtained from the element of the same name in the service provider's
804 Provider Metadata.

805 The schema fragment is as follows:

```
806 <element name="AuthnResponseEnvelope" type="lib:AuthnResponseEnvelopeType"/>
807 <complexType name="AuthnResponseEnvelopeType">
808   <complexContent>
809     <extension base="lib:ResponseEnvelopeType">
810       <sequence>
811         <element ref="lib:AuthnResponse"/>
812         <element name="AssertionConsumerServiceURL" type="anyURI"/>
813       </sequence>
814     </extension>
815   </complexContent>
816 </complexType>
817 <complexType name="ResponseEnvelopeType">
818   <sequence>
819     <any processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
820   </sequence>
821 </complexType>
```

822 3.2.5.2 Example

```
823 <AuthnResponseEnvelope>
824   <AuthnResponse> ... </AuthnResponse>
825   <AssertionConsumerServiceURL>
826     http://ServiceProvider.com/lecp_assertion_consumer
827   </AssertionConsumerServiceURL>
828 </AuthnResponseEnvelope>
```

829 3.3 Name Registration Protocol

830 At the time of federation, the identity provider generates an opaque handle that serves as the name
831 identifier the service provider and the identity provider use in referring to the Principal when
832 communicating with each other. This name identifier is termed the
833 IDPProvidedNameIdentifier.

834 Subsequent to federation, the service provider MAY register a different opaque handle with the
835 identity provider. This opaque handle is termed the SPProvidedNameIdentifier.

836 After name registration, the identity provider MUST use the SPProvidedNameIdentifier for
837 <saml:NameIdentifier> elements when communicating to the service provider about the
838 Principal. The service provider MUST continue using the IDPProvidedNameIdentifier for
839 <saml:NameIdentifier> elements when communicating to the identity provider about the
840 Principal.

841 Either the service provider, or the identity provider MAY register a new name identifier for a
842 Principal with each other at any time following federation. The name identifiers specified by either
843 provider SHOULD adhere to the following guidelines:

- 844 • The name identifier SHOULD be unique across the identity providers with which the
845 Principal's identity is federated.
- 846 • The name identifier SHOULD be unique across all name identifiers that have been registered
847 with the identity provider by this service provider.

848 3.3.1 Request

849 To register a `SPProvidedNameIdentifier` with an identity provider, the service provider sends a
850 `<RegisterNameIdentifierRequest>` message.

851 The same `<RegisterNameIdentifierRequest>` message may be sent by an identity provider,
852 seeking to change the `IDPProvidedNameIdentifier` stored by the service provider.

853 The `<RegisterNameIdentifierRequest>` message SHOULD be signed.

854 3.3.1.1 Element `<RegisterNameIdentifierRequest>`

855 The elements of the message are as follows:

856 `ProviderID` [Required]

857 The provider's identifier.

858 `IDPProvidedNameIdentifier` [Required]

859 The name identifier the service provider should use when communicating with the identity
860 provider.

861 `SPProvidedNameIdentifier` [Required]

862 The name identifier the identity provider will use when communicating to the service
863 provider.

864 `OldProvidedNameIdentifier` [Required]

865 In the case of either provider choosing to request a change of provided name identifiers, this
866 element holds the previous version.

867 `id` [Optional]

868 Identifier used to identify this element in the signature. See section 3.1.5, Signature
869 Verification.

870

871 The schema fragment is as follows:

```
872 <element name="RegisterNameIdentifierRequest" type="lib:RegisterNameIdentifierRequestType"/>
873 <complexType name="RegisterNameIdentifierRequestType">
874   <complexContent>
875     <extension base="samlp:RequestAbstractType">
876       <sequence>
877         <element ref="lib:ProviderID"/>
878         <element name="IDPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
879         <element name="SPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
880         <element name="OldProvidedNameIdentifier" type="saml:NameIdentifierType"/>
881       </sequence>
882       <attribute name="id" type="ID" use="optional"/>
883     </extension>
884   </complexContent>
885 </complexType>
886 <element name="IDPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
887 <element name="SPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
888 <element name="OldProvidedNameIdentifier" type="saml:NameIdentifierType"/>
889
```

890 3.3.1.2 Example

```
891 <RegisterNameIdentifierRequest id="12345" RequestID="eb20e77f-d982-44f9-936e-dd135bf437d4"
892 MajorVersion="1" MinorVersion="0" IssueInstant="2001-12-17T09:30:47Z">
893   <ds:Signature>...</ds:Signature>
894   <ProviderID>http://ServiceProvider.com</ProviderID>
```



```
895 <IDPProvidedNameIdentifier>342ad3d8-93ee-4c68-be35-cc9e7db39e2b</IDPProvidedNameIdentifier>  
896 <SPPProvidedNameIdentifier>e958019a</SPPProvidedNameIdentifier>  
897 <OldprovidedNameIdentifier>e895014a</OldProvidedNameIdentifier>  
898 </RegisterNameIdentifierRequest>
```

899 3.3.2 Response

900 The responding provider MUST respond with <RegisterNameIdentifierResponse>, which is
901 of type **StatusResponseType**. **StatusResponseType** is an extension of
902 **samlp:ResponseAbstractType** so that only a <samlp:Status> element exists in the body.

903 This message SHOULD be signed.

904 3.3.2.1 Element <RegisterNameIdentifierResponse>

905 The elements of the message are as follows:

906 Status [Required]

907 The status of the request processing.

908 The schema fragment is as follows:

```
909 <element name="RegisterNameIdentifierResponse" type="lib:StatusResponseType"/>  
910 <complexType name="StatusResponseType">  
911 <complexContent>  
912 <extension base="samlp:ResponseAbstractType">  
913 <sequence>  
914 <element ref="samlp:Status"/>  
915 </sequence>  
916 </extension>  
917 </complexContent>  
918 </complexType>
```

919 3.3.2.2 Example

```
920 <RegisterNameIdentifierResponse ResponseID="74ffec0f-1165-4fa3-b088-3dd2c2388b91"  
921 InResponseTo="eb20e77f-d982-44f9-936e-dd135bf437d4" MajorVersion="1" MinorVersion="0"  
922 IssueInstant="2001-12-17T09:30:47Z" Recipient="http://ServiceProvider.com">  
923 <ds:Signature>...</ds:Signature>  
924 <samlp:Status>  
925 <samlp:StatusCode Value="samlp:Success"/>  
926 </samlp:Status>  
927 </RegisterNameIdentifierResponse>
```

928 3.3.3 Processing Rules

929 The recipient MUST validate any signature present on the message. To be considered valid, the
930 signature provided MUST be the signature of the <ProviderID> contained in the message.
931 If the request includes an <IDPProvidedNameIdentifier> for which no federation exists
932 between the service provider and the identity provider, the provider MUST respond with a
933 <samlp:Status> with a <samlp:StatusCode> of lib:FederationDoesNotExist.

934 Otherwise, the identity provider MUST use <SPPProvidedNameIdentifier> when subsequently
935 communicating to the service provider regarding this Principal.

936 Either provider MAY choose to change their provided name identifier. In this case, the
937 <OldProvidedNameIdentifier> should contain the previous version of the identifier.

938 Changes to these identifiers may take a potentially significant amount of time to propagate through
939 the systems at both the sender and the receiver. Implementations MAY wish to allow each party to
940 accept either identifier for some period of time following the successful completion of a name
941 identifier change. Not doing so could result in the inability of the Principal to access resources.

942 3.4 Federation Termination Notification Protocol

943 When the Principal terminates an identity federation between a service provider and an identity
944 provider from the service provider, the service provider MUST send a
945 <FederationTerminationNotification> message to the identity provider. Semantically, the
946 service provider is stating that it will no longer accept authentication assertions from the identity
947 provider for the specified Principal.

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

952 This notification message is a one-way asynchronous message. Reasonable best-effort delivery
953 MUST be employed by all providers sending this message.

954 3.4.1 Message

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

957 The <FederationTerminationNotification> message SHOULD be signed.

958 3.4.1.1 Element <FederationTerminationNotification>

959 The elements are as follows:

960 ProviderID [Required]

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

962 NameIdentifier [Required]

963 The name identifier of the Principal for which federation is terminating. This name identifier
964 MUST be equal to the <saml:NameIdentifier> element (and its included attributes)
965 agreed upon earlier between the two communicating providers.

966 id [Optional]

967 Identifier used to identify this element in the signature. See section 3.1.5, Signature
968 Verification.

969 The schema fragment is as follows:

```
970 <element name="FederationTerminationNotification"  
971 type="lib:FederationTerminationNotificationType"/>  
972 <complexType name="FederationTerminationNotificationType">  
973 <complexContent>  
974 <extension base="samlp:RequestAbstractType">  
975 <sequence>  
976 <element ref="lib:ProviderID"/>  
977 <element ref="saml:NameIdentifier"/>  
978 </sequence>  
979 <attribute name="id" type="ID" use="optional"/>  
980 </extension>  
981 </complexContent>  
982 </complexType>
```

983 3.4.1.2 Example

```
984 <FederationTerminationNotification id="12345" RequestID="9ec2-eb65-4bce-ab8f-4becdf229815"  
985 MajorVersion="1" MinorVersion="0" IssueInstant="2001-12-17T09:30:47Z">  
986 <ds:Signature>...</ds:Signature>
```

987
988
989

```
<ProviderID>http://IdentityProvider.com</ProviderID>  
<saml:NameIdentifier>e958019a</saml:NameIdentifier>  
</FederationTerminationNotification>
```

990 3.4.2 Processing Rules

991 The receiving provider MUST validate any signature present on the message. The signature on the
992 message MUST be the signature of the <ProviderID> contained in the message. If the signature is
993 not valid, the provider MUST ignore the message.

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

998 A provider MAY choose to invalidate the session of a user for whom federation has been terminated.

999 3.5 Single Logout Protocol

1000 The Single Logout Protocol provides a one-way asynchronous message exchange protocol by which
1001 all sessions authenticated by a particular identity provider are near-simultaneously terminated. The
1002 Single Logout Protocol is used either when a Principal logs out at a service provider or when the
1003 Principal logs out at an identity provider.

1004 When the Principal invokes the single logout process at a service provider, the service provider
1005 MUST send a <LogoutNotification> message to the identity provider that provided the
1006 authentication service for the session.

1007 When either the Principal invokes a logout at the identity provider or a service provider sends a
1008 logout notification to the identity provider specifying that Principal, the identity provider MUST
1009 send a <LogoutNotification> message to each service provider to which it provided
1010 authentication assertions in the current session with the Principal, with the exception of the service
1011 provider that sent the <LogoutNotification> message to the Identity Provider.

1012 This notification is a one-way asynchronous message, and reasonable best-effort delivery SHOULD
1013 be employed by both service providers and identity providers.

1014 3.5.1 Message

1015 The <LogoutNotification> message indicates to the message receiver that a Principal's session
1016 was terminated. The message includes an optional <SessionIndex> element that MUST be
1017 specified if and only if the authentication statement in the assertion that the service provider used in
1018 establishing the session with the Principal contained a SessionIndex attribute. This message
1019 SHOULD be signed.

1020 3.5.1.1 Element <LogoutNotification>

1021 NameIdentifier [Required]

1022 The name identifier of the Principal that logged out. This name identifier MUST be equal to
1023 the <saml:NameIdentifier> element (including the equality of contained attributes)
1024 agreed upon between the two communicating providers.

1025 ProviderID [Required]

1026 The identifier of the provider that is making the request.

1027 SessionIndex [Optional]

1028 The session index specified in the authentication statement in the assertion that was used in
1029 establishing the session being terminated. If a <SessionIndex> element was present in the
1030 authentication statement, an identical <SessionIndex> MUST be present in the
1031 <LogoutNotification>. If no <SessionIndex> element was present in the
1032 authentication statement, the <SessionIndex> MUST be omitted from the
1033 <LogoutNotification>.

1034 id [Optional]

1035 Identifier used to identify this element in the signature. See section 3.1.5, Signature
1036 Verification.

1037 The schema fragment is as follows:

```
1038 <element name="LogoutNotification" type="lib:LogoutNotificationType"/>
1039 <complexType name="LogoutNotificationType">
1040   <complexContent>
1041     <extension base="samlp:RequestAbstractType">
1042       <sequence>
1043         <element ref="lib:ProviderID"/>
1044         <element ref="saml:NameIdentifier"/>
1045         <element name="SessionIndex" type="string" minOccurs="0"/>
1046       </sequence>
1047       <attribute name="id" type="ID" use="optional"/>
1048     </extension>
1049   </complexContent>
1050 </complexType>
```

1051 3.5.1.2 Example

```
1052 <LogoutNotification id="12345" RequestID="47693d03-7c33-4d65-931f-ddeb19fa6a73" MajorVersion="1"
1053 MinorVersion="0" IssueInstant="2001-12-17T09:30:47Z">
1054   <ds:Signature>...</ds:Signature>
1055   <ProviderID>http://ServiceProvider.com</ProviderID>
1056   <saml:NameIdentifier>342ad3d8-93ee-4c68-be35-cc9e7db39e2b</saml:NameIdentifier>
1057   <SessionIndex>3</SessionIndex>
1058 </LogoutNotification>
```

1059 3.5.2 Processing Rules

1060 Unique processing rules apply based on whether the message receiver is an identity provider or a
1061 service provider.

1062 3.5.2.1 Identity Provider Processing Rules

1063 When an identity provider receives the <LogoutNotification> message, the identity provider
1064 MUST validate that any signature present on the message is the signature of a service provider to
1065 which the identity provider provided an authentication assertion for the current session. If that holds,
1066 the identity provider SHOULD do the following:

- 1067 • Send a <LogoutNotification> message to each service providers for which the identity
1068 provider provided authentication assertions in the current session.
- 1069 • Terminate the Principal's current session as specified by the <saml:NameIdentifier>
1070 element.

1071 3.5.2.2 Service Provider Processing Rules

1072 When the service provider receives the <LogoutNotification> message, the service provider
1073 MUST validate the identity provider's signature contained in the <ds:Signature> element. If the

1074 signature is that of the identity provider that provided the authentication for the Principal's current
1075 session, the service provider MUST invalidate the Principal's session referred to in the
1076 <saml:NameIdentifier> element.

1077 4 Provider Metadata Schema

1078 For providers to communicate with each other, they must a priori have obtained metadata regarding
1079 each other. These provider metadata include items such as X.509 certificates and service endpoints.
1080 This specification defines metadata schemas for identity providers and service providers that may be
1081 used for provider metadata exchange. However, provider metadata exchange protocols are outside
1082 the scope of this specification.

1083 4.1 Generic Provider Descriptor

1084 Certain provider metadata are generic to both service providers and identity providers. The complex
1085 type **ProviderDescriptorType** contains the following elements:

1086 `ProviderID` [Required]

1087 The provider's URI-based identifier.

1088 `ProviderSuccinctID` [Required]

1089 The provider's succinct identifier. This value MUST be the SHA-1 hash of the `ProviderID`.
1090 Since the SHA-1 hash output may include arbitrary binary data, the `ProviderSuccinctID`
1091 element value is encoded into hexadecimal (according to the `hexBinary` XML Schema type)
1092 when included in an XML metadata document. Since the SHA-1 hash output always consists
1093 of 20 bytes of binary data, its `hexBinary` encoding will always consist of 40 hexadecimal
1094 digits.

1095 `KeyInfo` [Optional]

1096 The provider's public key.

1097 `SoapEndpoint` [Optional]

1098 The provider's SOAP endpoint URI.

1099 `SingleLogoutServiceURL` [Optional]

1100 The URL used for user-agent-based Single Logout Protocol profiles.

1101 `SingleLogoutServiceReturnURL` [Optional]

1102 The URL to which the provider redirects at the end of user-agent-based Single Logout
1103 Protocol profiles.

1104 `FederationTerminationServiceURL` [Optional]

1105 The URL used for user-agent-based Federation Termination Notification Protocol profiles.

1106 `FederationTerminationServiceReturnURL` [Optional]

1107 The URL to which the provider redirects at the end of user-agent-based Federation
1108 Termination Notification Protocol profiles.

1109 `FederationTerminationNotificationProtocolProfile` [Optional]

1110 The Federation Termination Notification Protocol profiles supported by the provider. Each
1111 value of the element MUST contain a valid Federation Termination Notification Protocol

1112 profile identification URI. The absence of this element SHALL mean that provider does not
1113 support any profile of the Federation Termination Notification Protocol.

1114 SingleLogoutProtocolProfile [Optional]

1115 The Single Logout Protocol profiles supported by the provider. Each element MUST contain
1116 a valid Single Logout Protocol profile identification URI. The absence of this element
1117 SHALL mean that the provider does not support any profile of the Single Logout Protocol.

1118 RegisterNameIdentifierProtocolProfile [Optional]

1119 The provider's preferred Register Name Identifier Protocol profile, which should be used by
1120 other providers when registering a new identifier. Each element MUST contain a valid
1121 Register Name Identifier Protocol profile identification URI. The absence of this element
1122 SHALL mean that the provider does not support any profile of the Register Name Identifier
1123 Protocol.

1124 RegisterNameIdentifierServiceURL [Optional]

1125 The URL used for user-agent-based Register Name Identifier Protocol profiles.

1126 RegisterNameIdentifierServiceReturnURL [Optional]

1127 The provider's redirecting URL for use after HTTP name registration has taken place.

1128 The schema fragment is as follows:

```
1129 <complexType name="ProviderDescriptorType">  
1130 <sequence>  
1131 <element name="ProviderID" type="anyURI"/>  
1132 <element name="ProviderSuccinctID" type="hexBinary"/>  
1133 <element ref="ds:KeyInfo" minOccurs="0"/>  
1134 <element name="SoapEndpoint" type="anyURI" minOccurs="0"/>  
1135 <element name="SingleLogoutServiceURL" type="anyURI" minOccurs="0"/>  
1136 <element name="SingleLogoutServiceReturnURL" type="anyURI" minOccurs="0"/>  
1137 <element name="FederationTerminationServiceURL" type="anyURI" minOccurs="0"/>  
1138 <element name="FederationTerminationServiceReturnURL" type="anyURI" minOccurs="0"/>  
1139 <element name="FederationTerminationNotificationProtocolProfile" type="anyURI"  
1140 minOccurs="0"/>  
1141 <element name="SingleLogoutProtocolProfile" type="anyURI" minOccurs="0"/>  
1142 <element name="RegisterNameIdentifierProtocolProfile" type="anyURI" minOccurs="0"/>  
1143 <element name="RegisterNameIdentifierServiceURL" type="anyURI" minOccurs="0"/>  
1144 <element name="RegisterNameIdentifierServiceReturnURL" type="anyURI" minOccurs="0"/>  
1145 </sequence>  
1146 </complexType>
```

1147 4.2 Service Provider Descriptor

1148 The additional service provider-specific metadata are as follows:

1149 AssertionConsumerServiceURL [Required]

1150 The service provider's URL for consuming assertions from identity providers.

1151 AuthnRequestsSigned [Required]

1152 Specifies whether the service provider will always sign authentication requests it sends to the
1153 identity provider.

1154 The schema fragment is as follows:

```
1155 <element name="SPDescriptor" type="lib:SPDescriptorType"/>  
1156 <complexType name="SPDescriptorType">  
1157 <complexContent>  
1158 <extension base="lib:ProviderDescriptorType">  
1159 <sequence>  
1160 <element name="AssertionConsumerServiceURL" type="anyURI"/>  
1161 <element name="AuthnRequestsSigned" type="boolean"/>
```

```

1162     </sequence>
1163   </extension>
1164 </complexContent>
1165 </complexType>

```

4.2.1 Example

```

1167 <SPDescriptor>
1168   <ProviderID>http://ServiceProvider.com</ProviderID>
1169   <ProviderSuccinctID>A9FD64E12C</ProviderSuccinctID>
1170   <ds:KeyInfo>...</ds:KeyInfo>
1171   <SoapEndpoint>http://ServiceProvider.com/soap</SoapEndpoint>
1172   <SingleLogoutServiceURL>http://ServiceProvider.com/liberty/slo</SingleLogoutServiceURL>
1173   <SingleLogoutServiceReturnURL>http://ServiceProvider.com/liberty/slo_return</SingleLogoutServiceR
1174   eturnURL>
1175   <FederationTerminationServiceURL>http://ServiceProvider.com/liberty/term</FederationTerminationSe
1176   rviceURL>
1177   <FederationTerminationServiceReturnURL>http://ServiceProvider.com/liberty/term_return</Federation
1178   TerminationServiceReturnURL>
1179   <FederationTerminationNotificationProtocolProfile>http://projectliberty.org/profiles/fedterm-idp-
1180   soap</FederationTerminationNotificationProtocolProfile>
1181   <FederationTerminationNotificationProtocolProfile>http://projectliberty.org/profiles/fedterm-idp-
1182   http</FederationTerminationNotificationProtocolProfile>
1183   <SingleLogoutProtocolProfile>http://projectliberty.org/profiles/slo-idp-
1184   http</SingleLogoutProtocolProfile>
1185   <RegisterNameIdentifierProtocolProfile>http://projectliberty.org/profiles/rni-idp-soap</
1186   RegisterNameIdentifierProtocolProfile>
1187   <RegisterNameIdentifierServiceURL>http://Provider.com/liberty/register_name</RegisterNameIdentifi
1188   erServiceURL>
1189   <RegisterNameIdentifierServiceReturnURL>http://Provider.com/liberty/ReturnURL</
1190   RegisterNameIdentifierReturnServiceURL>
1191   <AssertionConsumerServiceURL>http://ServiceProvider.com/liberty/assertion_consumer</AssertionCons
1192   umerServiceURL>
1193   <AuthnRequestsSigned>1</AuthnRequestsSigned>
1194 </SPDescriptor>

```

4.3 Identity Provider Descriptor

The additional identity provider-specific metadata are as follows:

SingleSignOnServiceURL [Required]

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

SingleSignOnProtocolProfile [Required]

The Single Sign-On Protocol profiles supported by the provider. Each element MUST contain a valid Single Sign-On Protocol profile identification URI.

The schema fragment is as follows:

```

1204 <element name="IDPDescriptor" type="lib:IDPDescriptorType"/>
1205 <complexType name="IDPDescriptorType">
1206   <complexContent>
1207     <extension base="lib:ProviderDescriptorType">
1208       <sequence>
1209         <element name="SingleSignOnServiceURL" type="anyURI"/>
1210         <element name="SingleSignOnProtocolProfile" type="anyURI"/>
1211       </sequence>
1212     </extension>
1213   </complexContent>
1214 </complexType>

```

4.3.1 Example

```

1216 <IDPDescriptor>
1217   <ProviderID>http://IdentityProvider.com</ProviderID>
1218   <ProviderSuccinctID>A9FD64E12C</ProviderSuccinctID>
1219   <ds:KeyInfo>...</ds:KeyInfo>
1220   <SoapEndpoint>http://IdentityProvider.com/soap</SoapEndpoint>
1221   <SingleLogoutServiceURL>http://IdentityProvider.com/liberty/slo</SingleLogoutServiceURL>

```

```
1222 <SingleLogoutServiceReturnURL>http://IdentityProvider.com/liberty/slo_return</SingleLogoutService
1223 ReturnURL>
1224 <FederationTerminationServiceURL>http://IdentityProvider.com/liberty/term</FederationTerminationS
1225 erviceURL>
1226 <FederationTerminationServiceReturnURL>http://IdentityProvider.com/liberty/term_return</Federatio
1227 nTerminationServiceReturnURL>
1228 <SingleSignOnServiceURL>http://IdentityProvider.com/liberty/sso</SingleSignOnServiceURL>
1229 <SingleSignOnProtocolProfile>http://projectliberty.org/profiles/brws-
1230
1231 </IDPDescriptor>
```

5 Schema Definition

```
1233 <?xml version="1.0" encoding="UTF-8"?>
1234 <schema targetNamespace="http://projectliberty.org/schemas/core/2002/12"
1235 xmlns:lib="http://projectliberty.org/schemas/core/2002/12"
1236 xmlns:ds="http://www.w3.org/2000/09/xmldsig#" xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
1237 xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
1238 xmlns:ac="http://projectliberty.org/schemas/authctx/2002/05" xmlns="http://www.w3.org/2001/XMLSchema"
1239 elementFormDefault="qualified" attributeFormDefault="unqualified">
1240 <import namespace="urn:oasis:names:tc:SAML:1.0:assertion" schemaLocation="http://www.oasis-
1241 open.org/committees/security/docs/draft-sstc-schema-assertion-31.xsd"/>
1242 <import namespace="urn:oasis:names:tc:SAML:1.0:protocol" schemaLocation="http://www.oasis-
1243 open.org/committees/security/docs/draft-sstc-schema-protocol-31.xsd"/>
1244 <import namespace="http://www.w3.org/2000/09/xmldsig#"
1245 schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
1246 <import namespace="http://projectliberty.org/schemas/authctx/2002/05" schemaLocation="draft-
1247 liberty-architecture-authentication-context-07.xsd"/>
1248
1249 <!-- Begin protocols schema -->
1250 <element name="ProviderID" type="anyURI"/>
1251 <element name="AuthnRequest" type="lib:AuthnRequestType"/>
1252 <complexType name="AuthnRequestType">
1253 <complexContent>
1254 <extension base="samlp:RequestAbstractType">
1255 <sequence>
1256 <element ref="lib:ProviderID"/>
1257 <element name="ForceAuthn" type="boolean" minOccurs="0"/>
1258 <element name="IsPassive" type="boolean" minOccurs="0"/>
1259 <element name="Federate" type="boolean" minOccurs="0"/>
1260 <element ref="lib:ProtocolProfile" minOccurs="0"/>
1261 <element ref="lib:AuthnContext" minOccurs="0"/>
1262 <element ref="lib:RelayState" minOccurs="0"/>
1263 <element ref="lib:AuthnContextComparisonType" minOccurs="0"/>
1264 </sequence>
1265 <attribute name="id" type="ID" use="optional"/>
1266 </extension>
1267 </complexContent>
1268 </complexType>
1269 <simpleType name="AuthnContextComparisonType">
1270 <restriction base="string">
1271 <enumeration value="exact"/>
1272 <enumeration value="minimum"/>
1273 <enumeration value="better"/>
1274 </restriction>
1275 </simpleType>
1276 <element name="RelayState"/>
1277 <element name="ProtocolProfile" type="anyURI"/>
1278 <element name="AuthnContext">
1279 <complexType>
1280 <choice>
1281 <element name="AuthnContextClassRef" type="anyURI" maxOccurs="unbounded"/>
1282 <element name="AuthnContextStatementRef" type="anyURI" maxOccurs="unbounded"/>
1283 </choice>
1284 </complexType>
1285 </element>
1286 <element name="AuthnRequestEnvelope" type="lib:AuthnRequestEnvelopeType"/>
1287 <complexType name="AuthnRequestEnvelopeType">
1288 <complexContent>
1289 <extension base="lib:RequestEnvelopeType">
1290 <sequence>
1291 <element ref="lib:AuthnRequest"/>
1292 <element ref="lib:ProviderID"/>
1293 <element name="ProviderName" type="string" minOccurs="0"/>
1294 <element name="AssertionConsumerServiceURL" type="anyURI"/>
1295 <element ref="lib:IDPList" minOccurs="0"/>
```



```
1296         <element name="IsPassive" type="boolean" minOccurs="0"/>
1297     </sequence>
1298 </extension>
1299 </complexContent>
1300 </complexType>
1301 <complexType name="RequestEnvelopeType">
1302     <sequence>
1303         <any processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
1304     </sequence>
1305 </complexType>
1306 <element name="AuthnResponseEnvelope" type="lib:AuthnResponseEnvelopeType"/>
1307 <complexType name="AuthnResponseEnvelopeType">
1308     <complexContent>
1309         <extension base="lib:ResponseEnvelopeType">
1310             <sequence>
1311                 <element ref="lib:AuthnResponse"/>
1312                 <element name="AssertionConsumerServiceURL" type="anyURI"/>
1313             </sequence>
1314         </extension>
1315     </complexContent>
1316 </complexType>
1317 <complexType name="ResponseEnvelopeType">
1318     <sequence>
1319         <any processContents="skip" minOccurs="0" maxOccurs="unbounded"/>
1320     </sequence>
1321 </complexType>
1322 <complexType name="SubjectType">
1323     <complexContent>
1324         <extension base="saml:SubjectType">
1325             <sequence>
1326                 <element ref="lib:IDPProvidedNameIdentifier"/>
1327             </sequence>
1328         </extension>
1329     </complexContent>
1330 </complexType>
1331 <element name="AuthnResponse" type="lib:AuthnResponseType"/>
1332 <complexType name="AuthnResponseType">
1333     <complexContent>
1334         <extension base="samlp:ResponseType">
1335             <sequence>
1336                 <element ref="lib:RelayState" minOccurs="0"/>
1337             </sequence>
1338         </extension>
1339     </complexContent>
1340 </complexType>
1341 <complexType name="AuthenticationStatementType">
1342     <complexContent>
1343         <extension base="saml:AuthenticationStatementType">
1344             <sequence>
1345                 <element name="AuthnContext" minOccurs="0">
1346                     <complexType>
1347                         <sequence>
1348                             <element name="AuthnContextClassRef" type="anyURI" minOccurs="0"/>
1349                             <choice>
1350                                 <element ref="AC:AuthenticationContextStatement"/>
1351                                 <element name="AuthnContextStatementRef" type="anyURI"/>
1352                             </choice>
1353                         </sequence>
1354                     </complexType>
1355                 </element>
1356             </sequence>
1357             <attribute name="ReauthenticateOnOrAfter" type="dateTime" use="optional"/>
1358             <attribute name="SessionIndex" type="string" use="optional"/>
1359         </extension>
1360     </complexContent>
1361 </complexType>
1362 <element name="RegisterNameIdentifierRequest" type="lib:RegisterNameIdentifierRequestType"/>
1363 <complexType name="RegisterNameIdentifierRequestType">
1364     <complexContent>
1365         <extension base="samlp:RequestAbstractType">
1366             <sequence>
1367                 <element ref="lib:ProviderID"/>
1368                 <element name="IDPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
1369                 <element name="SPPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
1370                 <element name="OldProvidedNameIdentifier" type="saml:NameIdentifierType"/>
1371             </sequence>
```

```
1372     <attribute name="id" type="ID" use="optional"/>
1373     </extension>
1374   </complexContent>
1375 </complexType>
1376 <element name="IDPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
1377 <element name="SPProvidedNameIdentifier" type="saml:NameIdentifierType"/>
1378 <element name="RegisterNameIdentifierResponse" type="lib:StatusResponseType"/>
1379 <element name="OldProvidedNameIdentifier" type="saml:NameIdentifierType"/>
1380 <complexType name="StatusResponseType">
1381   <complexContent>
1382     <extension base="samlp:ResponseAbstractType">
1383       <sequence>
1384         <element ref="samlp:Status"/>
1385       </sequence>
1386     </extension>
1387   </complexContent>
1388 </complexType>
1389 <element name="FederationTerminationNotification"
1390 type="lib:FederationTerminationNotificationType"/>
1391 <complexType name="FederationTerminationNotificationType">
1392   <complexContent>
1393     <extension base="samlp:RequestAbstractType">
1394       <sequence>
1395         <element ref="lib:ProviderID"/>
1396         <element ref="saml:NameIdentifier"/>
1397       </sequence>
1398       <attribute name="id" type="ID" use="optional"/>
1399     </extension>
1400   </complexContent>
1401 </complexType>
1402 <element name="LogoutNotification" type="lib:LogoutNotificationType"/>
1403 <complexType name="LogoutNotificationType">
1404   <complexContent>
1405     <extension base="samlp:RequestAbstractType">
1406       <sequence>
1407         <element ref="lib:ProviderID"/>
1408         <element ref="saml:NameIdentifier"/>
1409         <element name="SessionIndex" type="string" minOccurs="0"/>
1410       </sequence>
1411       <attribute name="id" type="ID" use="optional"/>
1412     </extension>
1413   </complexContent>
1414 </complexType>
1415 <!-- End protocols schema -->
1416 <!-- Begin assertion schema -->
1417 <element name="Assertion" type="lib:AssertionType"/>
1418 <complexType name="AssertionType">
1419   <complexContent>
1420     <extension base="saml:AssertionType">
1421       <attribute name="InResponseTo" type="saml:IDReferenceType"/>
1422       <attribute name="id" type="ID" use="optional"/>
1423     </extension>
1424   </complexContent>
1425 </complexType>
1426 <!-- End assertion schema -->
1427 <!-- Begin IDP list schema -->
1428 <element name="IDPList" type="lib:IDPListType"/>
1429 <complexType name="IDPListType">
1430   <sequence>
1431     <element ref="lib:IDPEntries"/>
1432     <element ref="lib:GetComplete" minOccurs="0" maxOccurs="unbounded"/>
1433   </sequence>
1434 </complexType>
1435 <element name="IDPEntry">
1436   <complexType>
1437     <sequence>
1438       <element ref="lib:ProviderID"/>
1439       <element name="ProviderName" type="string" minOccurs="0"/>
1440       <element name="Loc" type="anyURI"/>
1441     </sequence>
1442   </complexType>
1443 </element>
1444 <element name="IDPEntries">
1445   <complexType>
1446     <sequence>
1447       <element ref="lib:IDPEntry" maxOccurs="unbounded"/>

```

```
1448     </sequence>
1449   </complexType>
1450 </element>
1451 <element name="GetComplete" type="anyURI"/>
1452 <!-- End IDP list schema -->
1453 <!-- Begin provider metadata schema -->
1454 <complexType name="ProviderDescriptorType">
1455   <sequence>
1456     <element name="ProviderID" type="anyURI"/>
1457     <element name="ProviderSuccinctID" type="hexBinary"/>
1458     <element ref="ds:KeyInfo" minOccurs="0"/>
1459     <element name="SoapEndpoint" type="anyURI" minOccurs="0"/>
1460     <element name="SingleLogoutServiceURL" type="anyURI" minOccurs="0"/>
1461     <element name="SingleLogoutServiceReturnURL" type="anyURI" minOccurs="0"/>
1462     <element name="FederationTerminationServiceURL" type="anyURI" minOccurs="0"/>
1463     <element name="FederationTerminationServiceReturnURL" type="anyURI" minOccurs="0"/>
1464     <element name="FederationTerminationNotificationProtocolProfile" type="anyURI"
1465 minOccurs="0"/>
1466     <element name="SingleLogoutProtocolProfile" type="anyURI" minOccurs="0"/>
1467     <element name="RegisterNameIdentifierProtocolProfile" type="anyURI" minOccurs="0"/>
1468     <element name="RegisterNameIdentifierServiceURL" type="anyURI" minOccurs="0"/>
1469     <element name="RegisterNameIdentifierServiceReturnURL" type="anyURI" minOccurs="0"/>
1470   </sequence>
1471 </complexType>
1472 <element name="SPDescriptor" type="lib:SPDescriptorType"/>
1473 <complexType name="SPDescriptorType">
1474   <complexContent>
1475     <extension base="lib:ProviderDescriptorType">
1476       <sequence>
1477         <element name="AssertionConsumerServiceURL" type="anyURI"/>
1478         <element name="AuthnRequestsSigned" type="boolean"/>
1479       </sequence>
1480     </extension>
1481   </complexContent>
1482 </complexType>
1483 <element name="IDPDescriptor" type="lib:IDPDescriptorType"/>
1484 <complexType name="IDPDescriptorType">
1485   <complexContent>
1486     <extension base="lib:ProviderDescriptorType">
1487       <sequence>
1488         <element name="SingleSignOnServiceURL" type="anyURI"/>
1489         <element name="SingleSignOnProtocolProfile" type="anyURI"/>
1490       </sequence>
1491     </extension>
1492   </complexContent>
1493 </complexType>
1494 <!-- End provider metadata schema -->
1495 </schema>
```

1496 6 References

- 1497 [LibertyBindProf] J. Rouault, "Liberty Bindings and Profiles Specification.",
1498 [http://projectliberty.org/specs/liberty-architecture-bindings-profiles-](http://projectliberty.org/specs/liberty-architecture-bindings-profiles-v1.0.pdf)
1499 [v1.0.pdf](http://projectliberty.org/specs/liberty-architecture-bindings-profiles-v1.0.pdf), October 2002
- 1500 [LibertyGloss] H. Mauldin, "Liberty Glossary," [http://projectliberty.org/specs/liberty-](http://projectliberty.org/specs/liberty-tech-glossary-v1.0.pdf)
1501 [tech-glossary-v1.0.pdf](http://projectliberty.org/specs/liberty-tech-glossary-v1.0.pdf), October 2002.
- 1502 [RFC2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels",
1503 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 1504 [RFC3280] R. Housley, "Internet X.509 Public Key Infrastructure Certificate and
1505 Certificate Revocation List (CRL) Profile",
1506 <http://www.ietf.org/rfc/rfc3280.txt>, IETF RFC 3280, April 2002.
- 1507 [SAMLCore] P. Hallam-Baker et al., "Assertions and Protocol for the OASIS Security
1508 Assertion Markup Language (SAML)", [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/cs-sstc-core-01.pdf)
1509 [open.org/committees/security/docs/cs-sstc-core-01.pdf](http://www.oasis-open.org/committees/security/docs/cs-sstc-core-01.pdf), OASIS, May 2002.

- 1510 [Schema1] H. S. Thompson et al., “XML Schema Part 1: Structures,”
1511 <http://www.w3.org/TR/xmlschema-1/>, World Wide Web Consortium
1512 Recommendation, May 2001.
- 1513 [Schema2] H. S. Thompson et al., “XML Schema Part 1: Structures,”
1514 <http://www.w3.org/TR/xmlschema-2/>, World Wide Web Consortium
1515 Recommendation, May 2001.
- 1516
- 1517 [XMLDsig] D. Eastlake, J. Reagle, D. Solo, M. Bartel, J. Boyer, B. Fox, and E. Simon,
1518 “XML-Signature Syntax and Processing”, IETF Draft/W3C Candidate
1519 Recommendation, <http://www.w3.org/TR/xmlsig-core>, 31 October 2000.
1520
- 1521 [XMLCanon] J. Boyer, “Canonical XML, IETF Draft/W3C Candidate Recommendation”,
1522 <http://www.w3.org/xml-c14n>, 15 March 2001.