



Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0

OASIS Draft, 8 July 2004

Document identifier:

sstc-saml-profiles-2.0-draft-143

Location:

http://www.oasis-open.org/committees/documents.php?wg_abbrev=security

Editors:

Frederick Hirsch, Nokia
Scott Cantor, Internet2

Contributors:

TBD

Abstract:

This specification defines profiles for the use of SAML assertions and request-response messages in communications protocols and frameworks as well as attribute syntaxes for use in attribute statements.

Status:

This is a Draft.

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

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

Table of Contents

31	1 Introduction.....	6
32	1.1 Profile Concepts.....	6
33	2 Specification of Additional Profiles.....	8
34	2.1 Guidelines for Specifying Profiles.....	8
35	2.2 Guidelines for Specifying Attribute Profiles.....	8
36	3 Confirmation Method Identifiers.....	10
37	3.1 Holder of Key.....	10
38	3.2 Sender Vouches.....	10
39	3.3 Bearer.....	11
40	4 SSO Profiles of SAML.....	12
41	4.1 Web Browser SSO Profile.....	12
42	4.1.1 Required Information.....	12
43	4.1.2 Profile Overview.....	12
44	4.1.3 Profile Description.....	13
45	4.1.3.1 HTTP Request to Service Provider.....	13
46	4.1.3.2 Service Provider Determines Identity Provider.....	14
47	4.1.3.3 <AuthnRequest> issued by Service Provider to Identity Provider.....	14
48	4.1.3.4 Identity Provider identifies Principal.....	14
49	4.1.3.5 Identity Provider issues <Response> to Service Provider.....	14
50	4.1.3.6 Service Provider grants or denies access to User Agent.....	15
51	4.1.4 Use of Authentication Request Protocol.....	15
52	4.1.4.1 <AuthnRequest> Usage.....	15
53	4.1.4.2 <Response> Usage.....	15
54	4.1.4.3 <Response> Message Processing Rules.....	16
55	4.1.4.4 Artifact-Specific <Response> Message Processing Rules.....	17
56	4.1.4.5 POST-Specific Processing Rules.....	17
57	4.1.5 Unsolicited Responses.....	17
58	4.1.6 Use of Metadata.....	17
59	4.2 Enhanced Client and Proxy (ECP) Profile.....	18
60	4.2.1 Required Information.....	18
61	4.2.2 Preliminaries.....	18
62	4.2.3 Step 1: Accessing the Service Provider: ECP>SP.....	20
63	4.2.4 Steps 2,3: SOAP Message containing <AuthnRequest>: SP>ECP>IDP.....	20
64	4.2.4.1 PAOS Request Header Block: SP>ECP.....	21

65	4.2.4.2 ECP Request Header Block : SP > ECP	21
66	4.2.4.3 ECP RelayState Header Block : SP > ECP	22
67	4.2.4.4 SP>ECP Request Example	22
68	4.2.4.5 ECP>IDP Request Example	23
69	4.2.5 Steps 4,5: Authentication Response SOAP Message: IDP>ECP>SP	23
70	4.2.5.1 ECP Response Header Block : IDP > ECP	24
71	4.2.5.2 IDP>ECP Response Example	24
72	4.2.5.3 PAOS Response Header Block : ECP>SP	25
73	4.2.5.4 ECP>SP Response Example	25
74	4.2.6 Step 6: HTTP service response: SP>ECP	25
75	4.2.7 Security Considerations	26
76	4.2.8 ECP Profile XML Schema	26
77	4.3 Identity Provider Discovery Profile	27
78	4.3.1 Common Domain Cookie	27
79	4.3.2 Setting the Common Domain Cookie	27
80	4.3.3 Obtaining the Common Domain Cookie	28
81	4.4 Single Logout Profile	28
82	4.4.1 Required Information	28
83	4.4.2 Profile Overview	28
84	4.4.3 Profile Description	29
85	4.4.3.1 <LogoutRequest> issued by Service Provider to Identity Provider	29
86	4.4.3.2 Identity Provider determines Session Participants	30
87	4.4.3.3 <LogoutRequest> issued by Identity Provider to Session Participant/Authority	30
88	4.4.3.4 Session Participant/Authority issues <LogoutResponse> to Identity Provider	30
89	4.4.3.5 Identity Provider issues <LogoutResponse> to Service Provider	31
90	4.4.4 Use of Single Logout Protocol	31
91	4.4.4.1 <LogoutRequest> Usage	31
92	4.4.4.2 <LogoutResponse> Usage	32
93	4.4.5 Use of Metadata	32
94	4.5 Name Identifier Management Profile	32
95	4.5.1 Required Information	32
96	4.5.2 Profile Overview	32
97	4.5.3 Profile Description	33
98	4.5.3.1 <ManageNameIDRequest> issued by Requesting Identity/Service Provider	33
99	4.5.3.2 <ManageNameIDResponse> issued by Responding Identity/Service Provider	34
100	4.5.4 Use of Name Identifier Management Protocol	34
101	4.5.4.1 <ManageNameIDRequest> Usage	34
102	4.5.4.2 <ManageNameIDResponse> Usage	34
103	4.5.5 Use of Metadata	35

104	5 Artifact Resolution Profile.....	36
105	5.1 Required Information.....	36
106	5.2 Profile Overview.....	36
107	5.3 Profile Description.....	36
108	5.3.1 <ArtifactResolve> issued by Requesting Entity.....	36
109	5.3.2 <ArtifactResponse> issued by Responding Entity.....	37
110	5.4 Use of Artifact Resolution Protocol.....	37
111	5.4.1 <ArtifactResolve> Usage.....	37
112	5.4.2 <ArtifactResponse> Usage.....	37
113	5.5 Use of Metadata.....	37
114	6 Assertion Query/Request Profile.....	38
115	6.1 Required Information.....	38
116	6.2 Profile Overview.....	38
117	6.3 Profile Description.....	38
118	6.3.1 Query/Request issued by Requesting Entity.....	38
119	6.3.2 <Response> issued by SAML Authority.....	39
120	6.4 Use of Query/Request Protocol.....	39
121	6.4.1 Query/Request Usage.....	39
122	6.4.2 <Response> Usage.....	39
123	6.5 Use of Metadata.....	39
124	7 Name Identifier Mapping Profile.....	40
125	7.1 Required Information.....	40
126	7.2 Profile Overview.....	40
127	7.3 Profile Description.....	40
128	7.3.1 <NameIDMappingRequest> issued by Requesting Entity.....	40
129	7.3.2 <NameIDMappingResponse> issued by Identity Provider.....	41
130	7.4 Use of Name Identifier Mapping Protocol.....	41
131	7.4.1 <NameIDMappingRequest> Usage.....	41
132	7.4.2 <NameIDMappingResponse> Usage.....	41
133	7.4.2.1 Limiting Use of Mapped Identifier.....	41
134	7.5 Use of Metadata.....	41
135	8 SAML Attribute Profiles.....	43
136	8.1 Basic Attribute Profile.....	43
137	8.1.1 Required Information.....	43
138	8.1.2 SAML Attribute Naming.....	43
139	8.1.3 Profile-Specific XML Attributes.....	43
140	8.1.4 <AttributeDesignator> Comparison.....	43
141	8.1.5 SAML Attribute Values.....	43

142	8.1.6 Example.....	43
143	8.2 X.500/LDAP Attribute Profile.....	43
144	8.2.1 Required Information.....	44
145	8.2.2 SAML Attribute Naming.....	44
146	8.2.3 Profile-Specific XML Attributes.....	44
147	8.2.4 <AttributeDesignator> Comparison.....	44
148	8.2.5 SAML Attribute Values.....	44
149	8.2.6 Example.....	45
150	8.3 UUID Attribute Profile.....	45
151	8.3.1 Required Information.....	45
152	8.3.2 UUID and GUID Background.....	45
153	8.3.3 SAML Attribute Naming.....	45
154	8.3.4 Profile-Specific XML Attributes.....	45
155	8.3.5 <AttributeDesignator> Comparison.....	46
156	8.3.6 SAML Attribute Values.....	46
157	8.3.7 Standard DCE Attributes.....	46
158	8.3.7.1 Principal.....	46
159	8.3.7.2 Primary Group.....	46
160	8.3.7.3 Groups.....	46
161	8.3.8 Example.....	46
162	8.4 XACML Attribute Profile.....	46
163	8.4.1 Required Information.....	47
164	8.4.2 SAML Attribute Naming.....	47
165	8.4.3 Profile-Specific XML Attributes.....	47
166	8.4.4 <AttributeDesignator> Comparison.....	47
167	8.4.5 SAML Attribute Values.....	47
168	8.4.6 Profile-Specific Schema.....	47
169	8.4.7 Example.....	48
170	9 References.....	49

1 Introduction

171

172 This document specifies profiles for the use of SAML assertions and request-response messages in
173 communications protocols and frameworks.

174 A separate specification [SAMLCore] defines the SAML assertions and request-response messages
175 themselves and another [SAMLBind] defines protocol bindings.

1.1 Profile Concepts

176

177 ~~One type of SAML profile defines a set of constraints on the use of a general SAML protocol or assertion
178 capability for a particular environment or context of use. Profiles of this nature may constrain optionality,
179 require the use of specific SAML functionality (e.g. attributes, conditions, bindings), and in other respects
180 define the processing rules to be followed by profile actors.~~

181 ~~Another~~One type of SAML profile outlines a set of rules describing how to embed SAML assertions into
182 and extract them from a framework or protocol. Such a profile describes how SAML assertions are
183 embedded in or combined with other objects (for example, files of various types, or protocol data units of
184 communication protocols) by an originating party, communicated from the originating party to a receiving
185 party, and subsequently processed at the destination. A particular set of rules for embedding SAML
186 assertions into and extracting them from a specific class of <FOO> objects is termed a <FOO> *profile of*
187 *SAML*.

188 For example, a SOAP profile of SAML describes how SAML assertions can be added to SOAP messages,
189 how SOAP headers are affected by SAML assertions, and how SAML-related error states should be
190 reflected in SOAP messages.

191 ~~Another type of SAML profile defines a set of constraints on the use of a general SAML protocol or
192 assertion capability for a particular environment or context of use. Profiles of this nature may constrain
193 optionality, require the use of specific SAML functionality (e.g. attributes, conditions, bindings), and in
194 other respects define the processing rules to be followed by profile actors.~~

195 ~~A particular example of the latter are those that address SAML attributes. The SAML <Attribute> (and
196 <AttributeDesignator>) elements provide a great deal of flexibility in attribute naming, value syntax,
197 and including in-band metadata through the use of XML attributes. Interoperability is achieved by
198 constraining this flexibility when warranted by adhering to profiles that define how to use these elements
199 with greater specificity than the generic rules defined by [SAMLCore].~~

200 ~~Attribute profiles provide the definitions necessary to constrain SAML attribute expression when dealing
201 with particular types of attribute information or when interacting with external systems or other open
202 standards that require greater strictness.~~

203 The intent of this specification is to specify a selected set of profiles of various kinds in sufficient detail to
204 ensure that independently implemented products will interoperate.

205 For other terms and concepts that are specific to SAML, refer to the SAML glossary [SAMLGloss].

1.2 Notation

206

207 The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD
208 NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as
209 described in IETF RFC 2119 [RFC2119].

210 Listings of productions or other normative code appear like this.

211 Example code listings appear like this.

212 **Note:** Non-normative notes and explanations appear like this.

213 Conventional XML namespace prefixes are used throughout this specification to stand for their respective
214 namespaces as follows, whether or not a namespace declaration is present in the example:

215 • The prefix `saml`: stands for the SAML assertion namespace [SAMLCore].

216 • The prefix `samlp`: stands for the SAML request-response protocol namespace [SAMLCore].

217 • The prefix `md`: stands for the SAML metadata namespace [SAMLMeta].

218 • The prefix `ds`: stands for the W3C XML Signature namespace,
219 <http://www.w3.org/2000/09/xmldsig#> [XMLSig].

220 • The prefix `xenc`: stands for the W3C XML Encryption namespace,
221 <http://www.w3.org/2001/04/xmlenc#>.

222 • The prefix `SOAP-ENV`: stands for the SOAP 1.1 namespace,
223 <http://schemas.xmlsoap.org/soap/envelope> [SOAP1.1].

224 This specification uses the following typographical conventions in text: `<SAMLElement>`,
225 `<ns:ForeignElement>`, `Attribute`, **Datatype**, `OtherCode`. In some cases, angle brackets are used
226 to indicate non-terminals, rather than XML elements; the intent will be clear from the context.

2 Specification of Additional Profiles

227

228 This specification defines a selected set of profiles, but others will possibly be developed in the future. It is
229 not possible for the OASIS Security Services Technical Committee to standardize all of these additional
230 profiles for two reasons: it has limited resources and it does not own the standardization process for all of
231 the technologies used. The following sections offer guidelines for specifying profiles.

232 The SSTC welcomes submission of proposals from OASIS members for new profiles. OASIS members
233 may wish to submit these proposals for consideration by the SSTC in a future version of this specification.
234 Other members may simply wish to inform the committee of their work related to SAML. Please refer to
235 the SSTC web site for further details on how to submit such proposals to the SSTC.

2.1 Guidelines for Specifying Profiles

236

237 This section provides a checklist of issues that MUST be addressed by each profile.

- 238 1. Specify a URI that uniquely identifies the profile, postal or electronic contact information for the
239 author, and provide reference to previously defined profiles that the new profile updates or
240 obsoletes.
- 241 2. Describe the set of interactions between parties involved in the profile. Any restrictions on
242 applications used by each party and the protocols involved in each interaction must be explicitly
243 called out.
- 244 3. Identify the parties involved in each interaction, including how many parties are involved and
245 whether intermediaries may be involved.
- 246 4. Specify the method of authentication of parties involved in each interaction, including whether
247 authentication is required and acceptable authentication types.
- 248 5. Identify the level of support for message integrity, including the mechanisms used to ensure
249 message integrity.
- 250 6. Identify the level of support for confidentiality, including whether a third party may view the contents
251 of SAML messages and assertions, whether the profile requires confidentiality, and the
252 mechanisms recommended for achieving confidentiality.
- 253 7. Identify the error states, including the error states at each participant, especially those that receive
254 and process SAML assertions or messages.
- 255 8. Identify security considerations, including analysis of threats and description of countermeasures.
- 256 9. Identify SAML confirmation method identifiers defined and/or utilized by the profile.
- 257 10. Identify relevant SAML metadata defined and/or utilized by the profile.

2.2 Guidelines for Specifying Attribute Profiles

258

259 This section provides a checklist of items that MUST in particular be addressed by attribute profiles.

- 260 1. Specify a URI that uniquely identifies the profile, postal or electronic contact information for the
261 author, and provide reference to previously defined profiles that the new profile updates or
262 obsoletes.
- 263 2. Syntax and restrictions on the acceptable values of the NameFormat and Name attributes of SAML
264 <AttributeDesignator> and <Attribute> elements.
- 265 3. Any additional namespace-qualified XML attributes defined by the profile that may be used in SAML
266 <AttributeDesignator> and <Attribute> elements.

- 267 | 4. Rules for determining the equality of <saml:AttributeDesignator> elements as defined by the
268 | profile, for use when processing attributes, queries, etc.
- 269 | 5. Syntax and restrictions on values acceptable in the SAML <AttributeValue> element, including
270 | whether the xsi:type XML attribute can or should be used.

271

3 Confirmation Method Identifiers

272 The SAML assertion and protocol specification [SAMLCore] defines the `<SubjectConfirmation>`
273 element as a `Method` plus optional `<SubjectConfirmationData>`. The `<SubjectConfirmation>`
274 element SHOULD be used by the relying party to confirm that the request or message came from a
275 system entity that corresponds to the subject of the assertion, within the context of a particular profile.

276 The `Method` attribute indicates the specific method that the relying party should use to make this
277 determination. This may or may not have any relationship to an authentication that was performed
278 previously. Unlike the authentication context, the subject confirmation method will often be accompanied
279 by additional information, such as a certificate or key, in the `<SubjectConfirmationData>` element
280 that will allow the relying party to perform the necessary verification. A common set of attributes are also
281 defined and MAY be used to constrain the conditions under which the verification can take place.

282 It is anticipated that profiles will define and use several different values for `<ConfirmationMethod>`,
283 each corresponding to a different SAML usage scenario. The following methods are defined for use by
284 profiles defined within this specification and other profiles that find them useful.

3.1 Holder of Key

286 **URI:** urn:oasis:names:tc:SAML:2.0:cm:holder-of-key

287 One or more `<ds:KeyInfo>` elements MUST be present within the `<SubjectConfirmationData>`
288 element. An `xsi:type` attribute MAY be present in the `<SubjectConfirmationData>` element and
289 MUST be set to **saml:KeyInfoConfirmationDataType** (the QName prefix, if any, is arbitrary but must
290 reference the SAML assertion namespace).

291 As described in [XMLSig], each `<ds:KeyInfo>` element holds a key or information that enables an
292 application to obtain a key. The holder of a specified key is considered to be the subject of the assertion
293 by the asserting party.

294 Note that in accordance with [XMLSig], each `<ds:KeyInfo>` element MUST identify a single
295 cryptographic key. Multiple keys MAY be identified with separate `<ds:KeyInfo>` elements, such as when
296 different confirmation keys are needed for different relying parties.

297 **Example:** The holder of the key named "By-Tor" or the holder of the key named "Snow Dog" can confirm
298 itself as the subject.

```
299 <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">  
300   <SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">  
301     <ds:KeyInfo>  
302       <ds:KeyName>By-Tor</ds:KeyName>  
303     </ds:KeyInfo>  
304     <ds:KeyInfo>  
305       <ds:KeyName>Snow Dog</ds:KeyName>  
306     </ds:KeyInfo>  
307   </SubjectConfirmationData>  
308 </SubjectConfirmation>
```

3.2 Sender Vouches

310 **URI:** urn:oasis:names:tc:SAML:2.0:cm:sender-vouches

311 Indicates that no other information is available about the context of use of the assertion. The relying party

312 SHOULD utilize other means to determine if it should process the assertion further, subject to optional
313 constraints on confirmation using the attributes that MAY be present in the
314 <SubjectConfirmationData> element, as defined by [SAMLCore].

315 **3.3 Bearer**

316 **URI:** urn:oasis:names:tc:SAML:2.0:cm:bearer

317 The subject of the assertion is the bearer of the assertion, subject to optional constraints on confirmation
318 using the attributes that MAY be present in the <SubjectConfirmationData> element, as defined by
319 [SAMLCore].

320 **Example:** The bearer of the assertion can confirm itself as the subject, provided the assertion is delivered
321 in a message sent to "<https://www.serviceprovider.com/saml/consumer>" before 1:37 PM GMT on March
322 19th, 2004, in response to a request with ID "_1234567890".

```
323 <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">  
324   <SubjectConfirmationData InResponseTo="_1234567890"  
325     Recipient="https://www.serviceprovider.com/saml/consumer"  
326     NotOnOrAfter="2004-03-19T13:27:00Z"  
327   </SubjectConfirmationData>  
328 </SubjectConfirmation>
```

329 4 SSO Profiles of SAML

330 A set of profiles are defined to support single sign-on of browsers and other client devices.

- 331 • A web browser-based profile of the Authentication Request protocol in [SAMLCore] is defined to
- 332 support web single sign-on, supporting Scenario 1-1 of the SAML requirements document.
- 333 • An additional web SSO profile is defined to support enhanced clients.
- 334 • A profile of the Single Logout and Name Identifier Management protocols in [SAMLCore] is defined
- 335 over both front-channel (browser) and back-channel bindings.
- 336 • An additional profile is defined for identity provider discovery using cookies.

337 4.1 Web Browser SSO Profile

338 In the scenario supported by the web browser SSO profile, a web user either accesses a resource at a
339 service provider, or accesses an identity provider such that the service provider and desired resource are
340 understood or implicit. The web user authenticates (or has already authenticated) to the identity provider,
341 which then produces an authentication assertion (possibly with input from the service provider) and the
342 service provider consumes the assertion to establish a security context for the web user. During this
343 process, a name identifier might also be established between the providers for the principal, subject to the
344 parameters of the interaction and the consent of the parties.

345 To implement this scenario, a profile of the SAML Authentication Request protocol is used, in conjunction
346 with the HTTP Redirect, HTTP POST and HTTP Artifact bindings.

347 It is assumed that the user is using a standard commercial browser and can authenticate to the identity
348 provider by some means outside the scope of SAML.

349 4.1.1 Required Information

350 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:browser

351 **Contact information:** security-services-comment@lists.oasis-open.org

352 **SAML Confirmation Method Identifiers:** The SAML 2.0 "bearer" confirmation method identifier is used
353 by this profile. The following RECOMMENDED identifier has been assigned to this confirmation method:

354 urn:oasis:names:tc:SAML:2.0:cm:bearer

355 **Description:** Given below.

356 **Updates:** SAML 1.1 browser artifact and POST profiles and bearer confirmation method.

357 4.1.2 Profile Overview

358 The following figure illustrates the basic template for achieving SSO:

359 <need figure>

360 The following steps are described by the profile. Within an individual step, there may be one or more
361 actual message exchanges depending on the binding used for that step and other implementation-
362 dependent behavior.

363 1. HTTP Request to Service Provider

364 In step 1, the principal, via an HTTP User Agent, makes an HTTP request for a secured resource
365 at the service provider without a security context.

366 **2. Service Provider Determines Identity Provider**

367 In step 2, the service provider obtains the location of an endpoint at an identity provider for the
368 authentication request protocol that supports its preferred binding. The means by which this is
369 accomplished is implementation-dependent. The service provider MAY use the SAML identity
370 provider discovery profile described in 4.3.

371 **3. <AuthnRequest> issued by Service Provider to Identity Provider**

372 In step 3, the service provider issues an <AuthnRequest> message to be delivered by the user
373 agent to the identity provider. Either the HTTP Redirect, HTTP POST, or HTTP Artifact binding
374 can be used to transfer the message to the identity provider through the user agent.

375 **4. Identity Provider identifies Principal**

376 In step 4, the principal is identified by the identity provider by some means outside the scope of
377 this profile. This may require a new act of authentication, or it may reuse an existing authenticated
378 session.

379 **5. Identity Provider issues <Response> to Service Provider**

380 In step 5, the identity provider issues a <Response> message to be delivered by the user agent
381 to the service provider. Either the HTTP POST, or HTTP Artifact binding can be used to transfer
382 the message to the service provider through the user agent. The message may indicate an error,
383 or will include (at least) an authentication assertion. The HTTP Redirect binding MUST NOT be
384 used, as the response will typically exceed the URL length permitted by most user agents.

385 **6. Service Provider grants or denies access to Principal**

386 In step 6, having received the response from the identity provider, the service provider can
387 respond to the principal's user agent with its own error, or can establish its own security context
388 for the principal and return the requested resource.

389 Note that an identity provider can initiate this profile at step 5 and issue a <Response> message to a
390 service provider without the preceding steps.

391 **4.1.3 Profile Description**

392 If the profile is initiated by the service provider, start with section 4.1.3.1. If initiated by the identity provider,
393 start with section 4.1.3.5. In the descriptions below, the following are referred to:

394 **Single Sign-On Service**

395 This is the authentication request protocol endpoint at the identity provider to which the
396 <AuthnRequest> message (or artifact representing it) is delivered by the user agent.

397 **Assertion Consumer Service**

398 This is the authentication request protocol endpoint at the service provider to which the
399 <Response> message (or artifact representing it) is delivered by the user agent.

400 **4.1.3.1 HTTP Request to Service Provider**

401 If the first access is to the service provider, an arbitrary request for a resource can initiate the profile.
402 There are no restrictions on the form of the request. The service provider is free to use any means it
403 wishes to associate the subsequent interactions with the original request. Each of the bindings provide a
404 RelayState mechanism that the service provider MAY use to associate the profile exchange with the
405 original request. The service provider SHOULD reveal as little of the request as possible in the RelayState
406 value unless the use of the profile does not require such privacy measures.

407 **4.1.3.2 Service Provider Determines Identity Provider**

408 This step is implementation-dependent. The service provider MAY use the SAML identity provider
409 discovery profile, described in section 4.3. The service provider MAY also choose to redirect the user
410 agent to another service that is able to determine an appropriate identity provider. In such a case, the
411 service provider may issue an `<AuthnRequest>` (as in the next step) to this service to be relayed to the
412 identity provider, or it may rely on the intermediary service to issue an `<AuthnRequest>` message on its
413 behalf.

414 **4.1.3.3 `<AuthnRequest>` issued by Service Provider to Identity Provider**

415 Once an identity provider is selected, the location of its single sign-on service is determined, based on the
416 SAML binding chosen by the service provider for sending the `<AuthnRequest>`. Metadata (as in
417 [SAMLMeta]) MAY be used for this purpose. In response to an HTTP request by the user agent, an HTTP
418 response is returned containing an `<AuthnRequest>` message or an artifact, depending on the SAML
419 binding used, to be delivered to the identity provider's single sign-on service.

420 The exact format of this HTTP response and the subsequent HTTP request to the single sign-on service
421 is defined by the SAML binding used. Profile-specific rules for the contents of the `<AuthnRequest>`
422 message are included in section 4.1.4.1. If the HTTP Redirect or POST binding is used, the
423 `<AuthnRequest>` message is delivered directly to the identity provider in this step. If the HTTP Artifact
424 binding is used, the Artifact Resolution profile defined in section 5 is used by the identity provider, which
425 makes a callback to the service provider to retrieve the `<AuthnRequest>` message, using for example
426 the SOAP binding.

427 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0 ([SSL3]) or TLS
428 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The `<AuthnRequest>` message MAY
429 be signed, if authentication of the request issuer is required. The HTTP Artifact binding, if used, also
430 provides for an alternate means of authenticating the request issuer when the artifact is dereferenced.

431 The identity provider MUST process the `<AuthnRequest>` message as described in [SAMLCore]. This
432 may constrain the subsequent interactions with the user agent, for example if the `IsPassive` attribute is
433 included.

434 **4.1.3.4 Identity Provider identifies Principal**

435 At any time during the previous step or subsequent to it, the identity provider MUST establish the identity of
436 the principal (unless it returns an error to the service provider). The `ForceAuthn` `<AuthnRequest>`
437 attribute, if present with a value of `true`, obligates the identity provider to freshly establish this identity,
438 rather than relying on an existing session it may have with the principal. Otherwise, and in all other
439 respects, the identity provider may use any means to authenticate the user agent, subject to any
440 requirements included in the `<AuthnRequest>` in the form of the `<RequestedAuthnContext>`
441 element.

442 **4.1.3.5 Identity Provider issues `<Response>` to Service Provider**

443 Regardless of the success or failure of the `<AuthnRequest>`, the identity provider SHOULD produce an
444 HTTP response to the user agent containing a `<Response>` message or an artifact, depending on the
445 SAML binding used, to be delivered to the service provider's assertion consumer service.

446 The exact format of this HTTP response and the subsequent HTTP request to the assertion consumer
447 service is defined by the SAML binding used. Profile-specific rules on the contents of the `<Response>`
448 are included in section 4.1.4.2. If the HTTP POST binding is used, the `<Response>` message is delivered
449 directly to the service provider in this step. If the HTTP Artifact binding is used, the Artifact Resolution
450 profile defined in section 5 is used by the service provider, which makes a callback to the identity provider
451 to retrieve the `<Response>` message, using for example the SOAP binding.

452 The location of the assertion consumer service MAY be determined using metadata (as in [SAMLMeta]).
453 The identity provider MUST have some means to establish that this location is in fact controlled by the
454 service provider. A service provider MAY indicate the SAML binding and the specific assertion consumer
455 service to use in its <AuthnRequest> and the identity provider MUST honor them if it can.

456 It is RECOMMENDED that the HTTP requests in this step be made over either SSL 3.0 ([SSL3]) or TLS
457 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The <Assertion> element(s) in the
458 <Response> MUST be signed, if the HTTP POST binding is used, and MAY be signed if the HTTP-
459 Artifact binding is used.

460 The service provider MUST process the <Response> message and any enclosed <Assertion>
461 elements as described in [SAMLCore].

462 **4.1.3.6 Service Provider grants or denies access to User Agent**

463 To complete the profile, the service provider processes the <Response> and <Assertion>(s) and
464 grants or denies access to the resource. The service provider MAY establish a security context with the
465 user agent using any session mechanism it chooses. Any subsequent use of the <Assertion>(s)
466 provided are at the discretion of the service provider and other relying parties, subject to any restrictions
467 on use contained within them.

468 **4.1.4 Use of Authentication Request Protocol**

469 This profile is based on the Authentication Request protocol defined in [SAMLCore]. In the nomenclature
470 of actors enumerated in section 3.4 of that document, the service provider is the request issuer and the
471 relying party, and the principal is the presenter, requested subject, and confirming subject. There may be
472 additional relying parties or confirming subjects at the discretion of the identity provider (see below).

473 **4.1.4.1 <AuthnRequest> Usage**

474 A service provider MAY include any message content described in [SAMLCore], section 3.4.1. All
475 processing rules are as defined in [SAMLCore]. The <Issuer> element MUST be present and MUST
476 contain the unique identifier of the requesting service provider; the Format attribute MUST be omitted or
477 have a value of urn:oasis:names:tc:SAML:2.0:nameid-format:entity.

478 If the identity provider cannot or will not satisfy the request, it MUST respond with a <Response>
479 message containing an appropriate error status code or codes.

480 Note that the service provider MAY include a <Subject> element in the request that names the actual
481 identity about which it wishes to receive an assertion. This element MUST NOT contain any
482 <SubjectConfirmation> elements. If the identity provider does not recognize the principal as that
483 identity, then it MUST respond with a <Response> message containing an error status and no assertions.

484 The <AuthnRequest> message MAY be signed (as directed by the SAML binding used). If the HTTP
485 Artifact binding is used, authentication of the parties is OPTIONAL and any mechanism permitted by the
486 binding MAY be used.

487 Note that if the <AuthnRequest> is not authenticated and/or integrity protected, the information in it
488 MUST NOT be trusted except as advisory. Whether the request is signed or not, the identity provider
489 MUST insure that any <AssertionConsumerServiceURL> or
490 <AssertionConsumerServiceIndex> elements in the request are verified as belonging to the service
491 provider to whom the response will be sent. Failure to do so can result in a man-in-the-middle attack.

492 **4.1.4.2 <Response> Usage**

493 If the identity provider wishes to return an error, it MUST NOT include any assertions in the <Response>
494 message. Otherwise, if the request is successful (or if the response is not associated with a request), the

495 <Response> element MUST conform to the following:

- 496 • The <Issuer> element MAY be omitted, but if present it MUST contain the unique identifier
497 of the issuing identity provider; the Format attribute MUST be omitted or have a value of
498 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.
- 499 • It MUST contain at least one <Assertion>. Each assertion's <Issuer> element MUST
500 contain the unique identifier of the issuing identity provider; the Format attribute MUST be
501 omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-format:entity.
- 502 • The set of one or more assertions MUST contain at least one <AuthnStatement> that
503 reflects the authentication of the principal to the identity provider.
- 504 • At least one assertion containing an <AuthnStatement> MUST contain a <Subject>
505 element with at least one <SubjectConfirmation> element containing a Method of
506 urn:oasis:names:tc:SAML:2.0:cm:bearer. If the identity provider supports the Single
507 Logout profile, defined in section 4.4, any such authentication statements MUST include a
508 SessionIndex attribute to enable per-session logout requests by the service provider.
- 509 • Any bearer <SubjectConfirmationData> elements MUST contain a Recipient attribute
510 containing the service provider's assertion consumer service URL and a NotOnOrAfter
511 attribute that limits the window during which the assertion can be delivered. It MAY contain an
512 Address attribute limiting the client address from which the assertion can be delivered. It
513 MUST NOT contain a NotBefore attribute. If the containing message is in response to an
514 <AuthnRequest>, then the InResponseTo attribute MUST match the request's ID.
- 515 • Other statements and confirmation methods MAY be included in the assertion(s) at the
516 discretion of the identity provider. In particular, <AttributeStatement> elements MAY be
517 included. The <AuthnRequest> MAY contain an AttributeConsumingServiceIndex
518 XML attribute referencing information about desired or required attributes in [SAMLMeta]. The
519 identity provider MAY ignore this, or send other attributes at its discretion.
- 520 • The assertion(s) containing a bearer subject confirmation MUST contain an
521 <AudienceRestriction> including the service provider's unique identifier as an
522 <Audience>.
- 523 • Other conditions (and other <Audience> elements) MAY be included as requested by the
524 service provider or at the discretion of the identity provider. (Of course, any such conditions
525 MUST be understood by and accepted by the service provider in order for the assertion to be
526 considered valid.) The identity provider is NOT obligated to honor the requested set of
527 <Conditions> in the <AuthnRequest>, if any.

528 4.1.4.3 <Response> Message Processing Rules

529 Regardless of the SAML binding used, the service provider MUST:

- 530 • verify any signatures present on the assertion(s) or the response
- 531 • verify that the Recipient attribute in any bearer <SubjectConfirmationData> matches
532 the assertion consumer service URL to which the <Response> or artifact was delivered
- 533 • verify that the NotOnOrAfter attribute in any bearer <SubjectConfirmationData> has
534 not passed, subject to allowable clock skew between the providers
- 535 • verify that the InResponseTo attribute in the bearer <SubjectConfirmationData> equals
536 the ID of its original <AuthnRequest> message, unless the response is unsolicited (see
537 section 4.5) in which case the attribute MUST NOT be present
- 538 • verify that any assertions relied upon are valid in other respects

539 If any bearer `<SubjectConfirmationData>` includes an `Address` attribute, the service provider MAY
540 check the user agent's client address against it.

541 Any assertion which is not valid, or whose subject confirmation requirements cannot be met SHOULD be
542 discarded and SHOULD NOT be used to establish a security context for the principal.

543 If an `<AuthnStatement>` used to establish a security context for the principal contains a
544 `SessionNotOnOrAfter` attribute, the security context SHOULD be discarded once this time is reached,
545 unless the service provider reestablishes the principal's identity by repeating the use of this profile.

546 **4.1.4.4 Artifact-Specific `<Response>` Message Processing Rules**

547 If the HTTP Artifact binding is used to deliver the `<Response>`, the dereferencing of the artifact using the
548 Artifact Resolution profile MUST be mutually authenticated, integrity protected, and confidential.

549 The identity provider MUST ensure that only the service provider to whom the `<Response>` message has
550 been issued is given the message as the result of an `<ArtifactResolve>` request.

551 Either the SAML binding used to dereference the artifact or message signatures can be used to
552 authenticate the parties and protect the messages.

553 **4.1.4.5 POST-Specific Processing Rules**

554 If the HTTP POST binding is used to deliver the `<Response>`, the enclosed assertion(s) MUST be
555 signed.

556 The service provider MUST ensure that bearer assertions are not replayed, by maintaining the set of used
557 ID values for the length of time for which the assertion would be considered valid based on the
558 `NotOnOrAfter` attribute in the `<SubjectConfirmationData>`.

559 **4.1.5 Unsolicited Responses**

560 An identity provider may initiate this profile by delivering an unsolicited `<Response>` message to a service
561 provider.

562 An unsolicited `<Response>` MUST NOT contain an `InResponseTo` attribute, nor should any bearer
563 `<SubjectConfirmationData>` elements. If metadata as in [SAMLMeta] is used, the `<Response>` or
564 artifact SHOULD be delivered to the `<md:AssertionConsumerService>` endpoint of the service
565 provider labeled with the `isDefault` attribute.

566 Of special mention is that the identity provider SHOULD include a binding-specific "RelayState" parameter
567 that indicates, based on mutual agreement with the service provider, how to handle subsequent
568 interactions with the user agent. This MAY be the URL of a resource at the service provider.

569 **4.1.6 Use of Metadata**

570 [SAMLMeta] defines an endpoint element, `<md:SingleSignOnService>`, to describe supported
571 bindings and location(s) to which a service provider may send requests to an identity provider using this
572 profile.

573 The `<md:IDPDescriptor>` element's `WantAuthnRequestsSigned` attribute MAY be used by an
574 identity provider to document a requirement that requests be signed. The `<md:SPDescriptor>`
575 element's `AuthnRequestsSigned` attribute MAY be used by a service provider to document the
576 intention to sign all of its requests.

577 The providers MAY document the key(s) used to sign requests, responses, and assertions with
578 `<md:KeyDescriptor>` elements with a `use` attribute of `sign`. When encrypting SAML elements,

579 <md:KeyDescriptor> elements with a use attribute of encrypt MAY be used to document supported
580 encryption algorithms and settings, and public keys used to receive bulk encryption keys.

581 The indexed endpoint element <md:AssertionConsumerService> is used to describe supported
582 bindings and location(s) to which an identity provider may send responses to a service provider using this
583 profile. The index attribute is used to distinguish the possible endpoints that may be specified by
584 reference in the <AuthnRequest> message. The isDefault attribute is used to specify the endpoint to
585 use if not specified in a request.

586 The <md:SPDescriptor> element's WantAssertionsSigned attribute MAY be used by a service
587 provider to document a requirement that assertions delivered with this profile be signed. This is in addition
588 to any requirements for signing imposed by the use of a particular binding.

589 If the request or response message is delivered using the HTTP Artifact binding, the artifact issuer MUST
590 provide at least one <md:ArtifactResolutionService> endpoint element in its metadata.

591 The <md:AttributeConsumerDescriptor> element MAY be used to document the service provider's
592 need or desire for SAML attributes to be delivered along with authentication information. The actual
593 inclusion of attributes is of course at the discretion of the identity provider. One or more
594 <md:AttributeConsumingService> elements MAY be included in its metadata, each with an index
595 attribute to distinguish different services that MAY be specified by reference in the <AuthnRequest>
596 message. The isDefault attribute is used to specify a default set of attribute requirements.

597 4.2 Enhanced Client and Proxy (ECP) Profile

598 In the scenario supported by the enhanced client and proxy profile, a user of an enhanced client or proxy
599 either accesses a resource at a service provider, or accesses an identity provider such that the service
600 provider and desired resource are understood or implicit. The user authenticates (or has already
601 authenticated) to the identity provider, which then produces an authentication assertion (possibly with input
602 from the service provider) and the service provider consumes the assertion to establish a security context
603 for the user. During this process, a name identifier might also be established between the providers for the
604 principal, subject to the parameters of the interaction and the consent of the parties.

605 To implement this scenario, a profile of the SAML Authentication Request protocol is used, in conjunction
606 with the Reverse-SOAP binding.

607 It is assumed that the user is using an enhanced client or proxy (see below) and can authenticate to the
608 identity provider by some means outside the scope of SAML.

609 4.2.1 Required Information

610 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp

611 **Contact information:** security-services-comment@lists.oasis-open.org

612 **SAML Confirmation Method Identifiers:** The SAML 2.0 "bearer" confirmation method identifier is used
613 by this profile. The following RECOMMENDED identifier has been assigned to this confirmation method:

614 urn:oasis:names:tc:SAML:2.0:cm:bearer

615 **Description:** Given below.

616 **Updates:** None.

617 4.2.2 Preliminaries

618 The Enhanced Client and Proxy (ECP) profile specifies interactions between enhanced clients and/or
619 proxies, service providers, and identity providers. It is a generalization of the browser profile described in
620 section 4.1, and makes reference to it in a number of respects. If not otherwise specified by this profile

621 (and if not specific to the use of browser-based bindings), the rules specified in section 4.1 MUST be
622 observed.

623 An enhanced client or proxy (ECP) is a client or proxy that:

- 624 1. Has, or knows how to obtain, knowledge about the identity provider that the principal associated
625 with the client wishes to use with the service provider.
 - 626 • This allows a service provider to make an authentication request to such a client without the
627 need to know or discover the appropriate identity provider (effectively bypassing step 2 of the
628 browser profile).
- 629 2. Is able to use a reverse SOAP (PAOS) binding as profiled here for an authentication request and
630 response.
 - 631 • This enables a service provider to obtain an authentication assertion from a client that is not
632 necessarily directly addressable and not necessarily continuously available.
 - 633 • It leverages the benefits of SOAP while using a well-defined exchange pattern and profile to
634 enable interoperability.
 - 635 • The enhanced client may be viewed as a SOAP intermediary between the service provider and
636 the identity provider.

637 The enhanced client may be a browser or some other user agent that supports the functionality described
638 in this profile. An enhanced proxy is an HTTP proxy (typically a WAP gateway) that emulates an enhanced
639 client. Unless stated otherwise, all statements referring to enhanced clients are to be understood as
640 statements about both enhanced clients as well as enhanced client proxies.

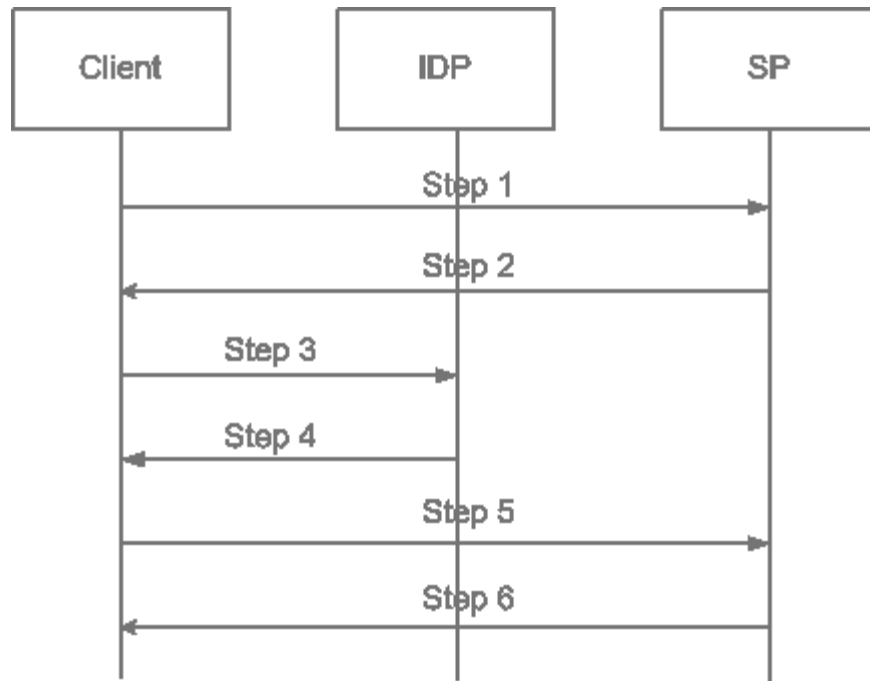
641 Since the enhanced client sends and receives messages in the body of HTTP requests and responses, it
642 has no arbitrary restrictions on the size of the protocol messages.

643 This profile leverages the Reverse SOAP binding [SAMLBind]. Implementers of this profile MUST follow
644 the rules for HTTP indications of PAOS support specified in that binding, in addition to those specified in
645 this profile. This specification profiles a PAOS SOAP header block conveyed between the HTTP
646 responder and the ECP but does not define PAOS. The PAOS specification is normative in case of
647 question regarding PAOS [PAOS].

648 This profile defines SOAP header blocks that accompany the SAML requests and responses. These
649 header blocks may be composed with other SOAP header blocks as necessary, for example with the
650 SOAP Message Security [WSS] header block to add security features if needed, for example encryption of
651 the authentication request.

652 Two sets of request/response SOAP header blocks are used: PAOS header blocks for generic PAOS
653 information and ECP profile-specific header blocks to convey information specific to ECP profile
654 functionality.

655 The following diagram shows the processing flow in the ECP profile:



656 **4.2.3 Step 1: Accessing the Service Provider: ECP>SP**

657 In step 1, the ECP accesses the service provider with an HTTP request. This HTTP request MUST
 658 conform to the PAOS binding, which means it must include the following HTTP header fields:

- 659 1. The HTTP Accept Header field indicating the ability to accept the MIME type
 660 "application/vnd.paos+xml"
- 661 2. The HTTP PAOS Header field specifying the PAOS version with urn:liberty:paos:2003-08 at
 662 minimum.
- 663 3. Furthermore, support for this profile MUST be specified in the HTTP PAOS Header field as a service
 664 value, with the value urn:oasis:names:tc:SAML:2.0:profiles:ecp. This value should
 665 correspond to the service attribute in the PAOS Request SOAP header block

666 To give an example, a user-agent may request a page from the SP as follows:

```

667 GET /index HTTP/1.1
668 Host: identity-service.example.com
669 Accept: text/html; application/vnd.paos+xml
670 PAOS: ver='urn:liberty:paos:2003-08' ; 'urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp'
  
```

671 **4.2.4 Steps 2,3: SOAP Message containing <AuthnRequest>: SP>ECP>IDP**

672 When the service provider requires a security context for the principal before providing a service or data, it
 673 can respond to the HTTP request using the PAOS binding with an <AuthnRequest> message in the
 674 HTTP response. The service provider will issue an HTTP 200 OK response to the ECP containing a single
 675 SOAP envelope.

676 The SOAP envelope MUST contain:

- 677 1. An <AuthnRequest> element in the SOAP body, intended for the ultimate SOAP recipient, the
 678 identity provider.
- 679 2. A PAOS SOAP header block targeted at the ECP using the SOAP actor value of
 680 http://schemas.xmlsoap.org/soap/actor/next/. This header block provides control
 681 information such as the URL to which to send the response in this solicit-response message

682 exchange pattern.

683 3. An ECP profile-specific Request SOAP header block targeted at the ECP using the SOAP actor
684 `http://schemas.xmlsoap.org/soap/actor/next/`. The ECP Request header block defines
685 information related to the authentication request that the ECP may need to process it, such as a list
686 of identity providers acceptable to the service provider, whether the ECP may interact with the
687 principal through the client, and the service provider's human-readable name that may be displayed
688 to the principal.

689 The SOAP envelope MAY contain an ECP RelayState SOAP header block targeted at the ECP using the
690 SOAP actor value of `http://schemas.xmlsoap.org/soap/actor/next/`. The header contains
691 state information to be returned by the ECP along with the SAML response.

692 The ECP will determine which identity provider is appropriate and route the SOAP message appropriately.

693 The ECP MUST remove the PAOS, ECP RelayState, and ECP Request header blocks before passing the
694 `<AuthnRequest>` message on to the identity provider, using the SAML SOAP binding.

695 Note that the `<AuthnRequest>` element may itself be signed by the service provider. In this and other
696 respects, the message rules specified in the browser SSO profile in section 4.1.4.1 MUST be followed.

697 Prior to or subsequent to this step, the identity provider MUST establish the identity of the principal by
698 some means, or it MUST return an error `<Response>` in step 4, described below.

699 **4.2.4.1 PAOS Request Header Block: SP>ECP**

700 The PAOS Request header block signals the use of PAOS processing and includes the following
701 attributes:

702 `responseConsumerURL` [Required]

703 Specifies where the ECP is to send an error response. Also used to verify the correctness of the
704 identity provider's response, by cross checking this location against the
705 `AssertionServiceConsumerURL` in the ECP response header block. This value MUST be the
706 same as the `AssertionServiceConsumerURL` (or the URL referenced in metadata) conveyed in
707 the `<AuthnRequest>`.

708 `service` [Required]

709 Indicates that the PAOS service being used is this SAML authentication profile. The value MUST be
710 `urn:oasis:names:tc:SAML:2.0:profiles:ecp`.

711 `S:mustUnderstand` [Required]

712 The value MUST be 1 (true). A SOAP fault MUST be generated if the PAOS header block is not
713 understood.

714 `S:actor` [Required]

715 The value MUST be `http://schemas.xmlsoap.org/soap/actor/next/`.

716 `messageID` [Optional]

717 Allows optional response correlation. It MAY be used in this profile, but is NOT required, since this
718 functionality is provided by the SAML protocol layer, via the `ID` attribute in the `<AuthnRequest>` and
719 the `InResponseTo` attribute in the `<Response>`.

720 The PAOS Request SOAP header block has no element content.

721 **4.2.4.2 ECP Request Header Block : SP > ECP**

722 The ECP Request SOAP header block is used to convey information needed by the ECP to process the

723 authentication request. It is mandatory and its presence signals the use of this profile. It contains the
724 following elements and attributes:

725 `S:mustUnderstand` [Required]

726 The value MUST be 1 (true). A SOAP fault MUST be generated if the ECP header block is not
727 understood.

728 `S:actor` [Required]

729 The value MUST be <http://schemas.xmlsoap.org/soap/actor/next/>.

730 `ProviderName` [Optional]

731 A human-readable name for the requesting service provider.

732 `IsPassive` [Optional]

733 A boolean value. If `true`, the identity provider and the client itself MUST NOT take control of the user
734 interface from the request issuer and interact with the principal in a noticeable fashion. If a value is not
735 provided, the default is `true`.

736 `<saml:Issuer>` [Required]

737 This element MUST contain the unique identifier of the requesting service provider; the `Format`
738 attribute MUST be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-`
739 `format:entity`.

740 `<samlp:IDPList>` [Optional]

741 Optional list of identity providers that the service provider recognizes and from which the ECP may
742 choose to service the request. See [SAMLCore] for details on the content of this element.

743 See section 4.2.8 for the XML schema that defines this header block.

744 **4.2.4.3 ECP RelayState Header Block : SP > ECP**

745 The ECP RelayState SOAP header block is used to convey state information from the service provider
746 that it will need later when processing the response from the ECP. It is optional, but if used, the ECP
747 MUST include an identical header block in the response in step 5. It contains the following attributes:

748 `S:mustUnderstand` [Required]

749 The value MUST be 1 (true). A SOAP fault MUST be generated if the header block is not understood.

750 `S:actor` [Required]

751 The value MUST be <http://schemas.xmlsoap.org/soap/actor/next/>.

752 The content of the header block element is a string containing state information created by the requester.
753 If provided, the ECP MUST include the same value in a RelayState header block when responding to the
754 service provider in step 5. The string value MUST NOT exceed 80 bytes in length and SHOULD be
755 integrity protected by the requester independent of any other protections that may or may not exist during
756 message transmission.

757 See section 4.2.8 for the XML schema that defines this header block.

758 **4.2.4.4 SP>ECP Request Example**

759 The following is an example of the SOAP authentication request from the service provider to the ECP:

```
760 <S:Envelope  
761     xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
762     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
763     xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
```

```

764 <S:Header>
765   <paos:Request xmlns:paos="urn:liberty:paos:2003-08"
766     responseConsumerURL="http://identity-service.example.com/abc"
767     messageID="6c3a4f8b9c2d" S:actor="next" S:mustUnderstand="1"
768     service="urn:oasis:names:tc:SAML:2.0:profiles:ecp">
769   </paos:Request>
770   <ecp:Request xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
771     S:mustUnderstand="1" S:actor="http://schemas.xmlsoap.org/soap/actor/next/"
772     ProviderName="Service Provider X" IsPassive="0">
773   <saml:Issuer>https://ServiceProvider.example.com</saml:Issuer>
774   <samlp:IDPList>
775     <samlp:IDPEntry ProviderID="https://IdentityProvider.example.com"
776       Name="Identity Provider X"
777       Loc="https://IdentityProvider.example.com/saml2/sso"
778     </samlp:IDPEntry>
779     <samlp:GetComplete>
780       https://ServiceProvider.example.com/idplist?id=604be136-fe91-441e-afb8
781     </samlp:GetComplete>
782   </samlp:IDPList>
783 </ecp:Request>
784 <ecp:RelayState xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
785   S:mustUnderstand="1" S:actor="http://schemas.xmlsoap.org/soap/actor/next/"
786   ...
787 </ecp:RelayState>
788 </S:Header>
789 <S:Body>
790   <samlp:AuthnRequest> ... </samlp:AuthnRequest>
791 </S:Body>
792 </S:Envelope>

```

793 4.2.4.5 ECP>IDP Request Example

794 As noted above, the PAOS and ECP header blocks are removed from the SOAP message by the ECP
795 before the authentication request is forwarded to the identity provider. An example authentication request
796 from the ECP to the identity provider is as follows:

```

797 <S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
798   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
799   <S:Body>
800     <samlp:AuthnRequest> ... </samlp:AuthnRequest>
801   </S:Body>
802 </S:Envelope>

```

803 4.2.5 Steps 4,5: Authentication Response SOAP Message: IDP>ECP>SP

804 The identity provider returns a SAML <Response> message (or SOAP fault) when presented with an
805 authentication request, after having established the identity of the principal. The SAML response is
806 conveyed using the SAML SOAP binding in a SOAP message with a <Response> element in the SOAP
807 body, intended for the service provider as the ultimate SOAP receiver. The rules for the response

808 specified in the browser SSO profile in section 4.1.4.2 MUST be followed.

809 The identity provider's response message MUST contain a profile-specific ECP Response SOAP header
810 block, and MAY contain an ECP RelayState header block, both targeted at the ECP. The ECP removes
811 the header block(s), and MAY add a PAOS Response SOAP header block and an ECP RelayState
812 header block before forwarding the SOAP response to the service provider using the PAOS binding.

813 The <paos:Response> SOAP header block in the response to the service provider is generally used to
814 correlate this response to an earlier request from the service provider. In this profile, the correlation
815 refToMessageID attribute is not required since the SAML <Response> element's InResponseTo
816 attribute may be used for this purpose, but if the <paos:Request> SOAP Header block had a
817 messageID then the <paos:Response> SOAP header block MUST be used.

818 The RelayState header block value is typically provided by the service provider to the ECP with its request,
819 but if the identity provider is producing an unsolicited response (without having received a corresponding
820 SAML request), then it SHOULD include a RelayState header block that indicates, based on mutual
821 agreement with the service provider, how to handle subsequent interactions with the ECP. This MAY be
822 the URL of a resource at the service provider.

823 If the service provider included a RelayState SOAP header block in its request to the ECP, or if the identity
824 provider included a RelayState SOAP header block with its response, then the ECP MUST include an
825 identical header block with the SAML response sent to the service provider. The service provider's value
826 for this header block (if any) MUST take precedence.

827 **4.2.5.1 ECP Response Header Block : IDP > ECP**

828 The ECP response SOAP header block MUST be used on the response from the identity provider to the
829 ECP. It contains the following attributes:

830 S:mustUnderstand [Required]

831 The value MUST be 1 (true). A SOAP fault MUST be generated if the ECP header block is not
832 understood.

833 S:actor [Required]

834 The value MUST be next.

835 AssertionConsumerServiceURL [Required]

836 Set by the identity provider based on the <AuthnRequest> message or the service provider's
837 metadata obtained by the identity provider.

838 The ECP MUST confirm that this value corresponds to the value the ECP obtained in the
839 responseConsumerURL in the PAOS Request SOAP header block it received from the service
840 provider. Since the responseConsumerURL MAY be relative and the
841 AssertionConsumerServiceURL is absolute, some processing/normalization may be required.

842 This mechanism is used for security purposes to confirm the correct response destination. If the
843 values do not match, then the ECP MUST generate a SOAP fault response to the service provider
844 and MUST NOT return the SAML response.

845 The ECP Response SOAP header has no element content.

846 See section 4.2.8 for the XML schema that defines this header block.

847 **4.2.5.2 IDP>ECP Response Example**

```
848 <S:Envelope  
849     xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"  
850     xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
851     xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
```



```

852 <S:Header>
853   <ecp:Response S:mustUnderstand="1" S:actor="next"
854 AssertionConsumerServiceURL="https://ServiceProvider.example.com/ecp_assertion_consum
855 er"/>
856 </S:Header>
857 <S:Body>
858   <samlp:Response> ... </samlp:Response>
859 </S:Body>
860 </S:Envelope>

```

861 4.2.5.3 PAOS Response Header Block : ECP>SP

862 The PAOS Response header block includes the following attributes:

863 `S:mustUnderstand` [Required]

864 The value MUST be 1 (true). A SOAP fault MUST be generated if the PAOS header block is not
865 understood.

866 `S:actor` [Required]

867 The value MUST be `next`.

868 `refToMessageID` [Optional]

869 Allows correlation with the PAOS request. This optional attribute (and the header block as a whole)
870 MUST be added by the ECP if the corresponding PAOS request specified the `messageID` attribute.

871 Note that the equivalent functionality is provided in SAML using `<AuthnRequest>` and `<Response>`
872 correlation.

873 The PAOS Response SOAP header has no element content.

874 4.2.5.4 ECP>SP Response Example

```

875 <S:Envelope
876   xmlns:paos="urn:liberty:paos:2003-08"
877   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
878   xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
879 <S:Header>
880   <paos:Response refToMessageID="6c3a4f8b9c2d" S:actor="next" S:mustUnderstand="1"/>
881   <ecp:RelayState xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
882     S:mustUnderstand="1" S:actor="next">
883     ...
884   </ecp:RelayState>
885 </S:Header>
886 <S:Body>
887   <samlp:Response> ... </samlp:Response>
888 </S:Body>
889 </S:Envelope>

```

890 4.2.6 Step 6: HTTP service response: SP>ECP

891 Once the service provider has received the SAML response in an HTTP request (in a SOAP envelope
892 using PAOS), it may respond with the service data in the HTTP response. In consuming the response, the

893 rules specified in the browser SSO profile in section 4.1.4.3 and 4.1.4.5 MUST be followed. That is, the
894 same processing rules used when receiving the <Response> with the HTTP POST binding apply to the
895 use of PAOS.

896 4.2.7 Security Considerations

- 897 1. The <AuthnRequest> message SHOULD be signed. Per the rules specified by the browser SSO
898 profile, the assertions enclosed in the <Response> MUST be signed. The delivery of the response
899 in the SOAP envelope via PAOS is essentially analogous to the use of the HTTP POST binding and
900 security countermeasures appropriate to that binding are used.
- 901 2. The SOAP headers should be integrity protected, such as with SOAP Message Security or through
902 the use of SSL/TLS over every HTTP exchange with the client.
- 903 3. The service provider should be authenticated to the ECP, for example with server-side TLS
904 authentication.
- 905 4. The ECP should be authenticated to the identity provider, such as by maintaining an authenticated
906 session.

907 4.2.8 ECP Profile XML Schema

908 The following normative XML schema defines the SOAP Request/Response header blocks used by this
909 profile.

```
910 <schema
911   targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
912   xmlns="http://www.w3.org/2001/XMLSchema"
913   xmlns:ecp="urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"
914   xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
915   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
916   xmlns:S="http://schemas.xmlsoap.org/soap/envelope/"
917   elementFormDefault="unqualified"
918   attributeFormDefault="unqualified"
919   blockDefault="substitution"
920   version="2.0">
921   <import namespace="urn:oasis:names:tc:SAML:2.0:protocol"
922     schemaLocation="sstc-saml-schema-protocol-2.0.xsd"/>
923   <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
924     schemaLocation="sstc-saml-schema-assertion-2.0.xsd"/>
925   <import namespace="http://schemas.xmlsoap.org/soap/envelope/"
926     schemaLocation="http://schemas.xmlsoap.org/soap/envelope/" />
927
928   <element name="Request" type="ecp:RequestType"/>
929   <complexType name="RequestType">
930     <sequence>
931       <element ref="saml:Issuer"/>
932       <element ref="samlp:IDPList" minOccurs="0"/>
933     </sequence>
934     <attribute ref="S:mustUnderstand" use="required"/>
935     <attribute ref="S:actor" use="required"/>
936     <attribute name="ProviderName" type="string" use="optional"/>
937     <attribute name="IsPassive" type="boolean" use="optional"/>
938   </complexType>
939
940   <element name="Response" type="ecp:ResponseType"/>
941   <complexType name="ResponseType">
942     <attribute ref="S:mustUnderstand" use="required"/>
943     <attribute ref="S:actor" use="required"/>
944     <attribute name="AssertionConsumerServiceURL" type="anyURI"
945     use="required"/>
946   </complexType>
947
948   <element name="RelayState" type="ecp:RelayStateType"/>
```

```
946     <complexType name="RelayStateType">
947         <simpleContent>
948             <extension base="string">
949                 <attribute ref="S:mustUnderstand" use="required"/>
950                 <attribute ref="S:actor" use="required"/>
951             </extension>
952         </simpleContent>
953     </complexType>
954 </schema>
```

955 **4.3 Identity Provider Discovery Profile**

956 This section defines a profile by which a service provider can discover which identity providers a principal
957 is using with the Web Browser SSO profile. In deployments having more than one identity provider,
958 service providers need a means to discover which identity provider(s) a principal uses. The discovery
959 profile relies on a cookie that is written in a domain that is common between identity providers and service
960 providers in a deployment. The domain that the deployment predetermines is known as the common
961 domain in this profile, and the cookie containing the list of identity providers is known as the common
962 domain cookie.

963 Which entities host web servers in the common domain is a deployment issue and is outside the scope of
964 this profile.

965 **4.3.1 Common Domain Cookie**

966 The name of the cookie MUST be `_saml_idp`. The format of the cookie value MUST be a set of one or
967 more base-64 encoded URI values separated by a single space character. Each URI is the unique
968 identifier of an identity provider, as defined in section 8.3.6 of [SAMLCore]. The final set of values is then
969 URL encoded.

970 The common domain cookie writing service (see below) SHOULD append the identity provider's unique
971 identifier to the list. If the identifier is already present in the list, it MAY remove and append it when
972 authentication of the principal occurs. The intent is that the most recently established identity provider
973 session is the last one in the list.

974 The cookie MUST be set with no Path prefix or a Path prefix of `/`. The Domain MUST be set to
975 `"[common-domain]"` where `[common-domain]` is the common domain established within the deployment
976 for use with this profile. The cookie MUST be marked as secure.

977 Cookie syntax should be in accordance with [RFC2965] or [NetscapeCookie]. The cookie MAY be either
978 session-only or persistent. This choice may be made within a deployment, but should apply uniformly to all
979 identity providers in the deployment.

980 **4.3.2 Setting the Common Domain Cookie**

981 After the identity provider authenticates a principal, it MAY set the common domain cookie. The means by
982 which the identity provider sets the cookie are implementation-specific so long as the cookie is
983 successfully set with the parameters given above. One possible implementation strategy follows and
984 should be considered non-normative. The identity provider may:

- 985 • Have previously established a DNS and IP alias for itself in the common domain.
- 986 • Redirect the user agent to itself using the DNS alias using a URL specifying "https" as the URL
987 scheme. The structure of the URL is private to the implementation and may include session
988 information needed to identify the user-agent.
- 989 • Set the cookie on the redirected user agent using the parameters specified above.
- 990 • Redirect the user agent back to itself, or, if appropriate, to the service provider.

991 4.3.3 Obtaining the Common Domain Cookie

992 When a service provider needs to discover which identity providers a principal uses, it invokes an
993 exchange designed to present the common domain cookie to the service provider after it is read by an
994 HTTP server in the common domain.

995 If the HTTP server in the common domain is operated by the service provider or if other arrangements are
996 in place, the service provider MAY utilize the HTTP server in the common domain to relay its
997 <AuthnRequest> to the identity provider for an optimized single sign-on process.

998 The specific means by which the service provider reads the cookie are implementation-specific so long as
999 it is able to cause the user agent to present cookies that have been set with the parameters given in
1000 section Section 4.3.1. One possible implementation strategy is described as follows and should be
1001 considered non-normative. Additionally, it may be sub-optimal for some applications.

- 1002 • Have previously established a DNS and IP alias for itself in the common domain.
- 1003 • Redirect the user agent to itself using the DNS alias using a URL specifying "https" as the URL
1004 scheme. The structure of the URL is private to the implementation and may include session
1005 information needed to identify the user-agent.
- 1006 • Set the cookie on the redirected user agent using the parameters specified above.
- 1007 • Redirect the user agent back to itself, or, if appropriate, to the identity provider.

1008 4.4 Single Logout Profile

1009 In the scenario supported by the Single Logout profile, a user has an authenticated session at one or more
1010 service providers (the session participants). The identity provider that supplied assertions to the service
1011 providers acts as (or on behalf of) the session authority. The user then wishes to terminate his or her
1012 sessions, or has their sessions administratively terminated (due to timeout, etc.). To implement this
1013 scenario, a profile of the SAML Single Logout protocol is used.

1014 The profile allows the protocol to be combined with a synchronous binding, such as the SOAP binding, or
1015 with asynchronous "front-channel" bindings, such as the HTTP Redirect, POST, or Artifact bindings. A
1016 front-channel binding may be required, for example, in cases in which a principal's session state exists
1017 solely in a user agent in the form of a cookie and a direct interaction between the user agent and the
1018 session participant is required.

1019 4.4.1 Required Information

1020 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:logout

1021 **Contact information:** security-services-comment@lists.oasis-open.org

1022 **Description:** Given below.

1023 **Updates:** None

1024 4.4.2 Profile Overview

1025 The following figure illustrates the basic template for achieving single logout:

1026 <need figure>

1027 The following steps are described by the profile. Within an individual step, there may be one or more
1028 actual message exchanges depending on the binding used for that step and other implementation-
1029 dependent behavior.

1030 1. <LogoutRequest> issued by Service Provider to Identity Provider

1031 In step 1, the service provider initiates single logout and terminates a principal's session(s) by
1032 sending a <LogoutRequest> message to the identity provider from whom it received the
1033 corresponding authentication assertion. The request may be sent directly to the identity provider
1034 or sent indirectly through the user agent.

1035 **2. Identity Provider determines Session Participants**

1036 In step 2, the identity provider uses the contents of the <LogoutRequest> message (or if
1037 initiating logout itself, some other mechanism) to determine the session(s) being terminated. If
1038 there are no other session participants, the profile proceeds with step 5. Otherwise, steps 3 and 4
1039 are repeated for each session participant identified.

1040 **3. <LogoutRequest> issued by Identity Provider to Session Participant/Authority**

1041 In step 3, the identity provider issues a <LogoutRequest> message to a session participant or
1042 session authority related to one or more of the session(s) being terminated. The request may be
1043 sent directly to the entity or sent indirectly through the user agent (if consistent with the form of the
1044 request in step 1).

1045 **4. Session Participant/Authority issues <LogoutResponse> to Identity Provider**

1046 In step 4, a session participant or session authority terminates the principal's session(s) as
1047 directed by the request (if possible) and returns a <LogoutResponse> to the identity provider.
1048 The response may be returned directly to the identity provider or indirectly through the user agent
1049 (if consistent with the form of the request in step 3).

1050 **5. Identity Provider issues <LogoutResponse> to Service Provider**

1051 In step 5, the identity provider issues a <LogoutResponse> message to the original requesting
1052 service provider. ~~be delivered by the user agent to the service provider.~~ The response may be
1053 returned directly to the service provider or indirectly through the user agent (if consistent with the
1054 form of the request in step 1).

1055 Note that an identity provider (acting as session authority) can initiate this profile at step 2 and issue a
1056 <LogoutRequest> to all session participants, also skipping step 5.

1057 **4.4.3 Profile Description**

1058 If the profile is initiated by a service provider, start with section 4.4.3.1. If initiated by the identity provider,
1059 start with section 4.4.3.2. In the descriptions below, the following is referred to:

1060 **Single Logout Service**

1061 This is the single logout protocol endpoint at an identity or service provider to which the
1062 <LogoutRequest> or <LogoutResponse> messages (or an artifact representing them) are
1063 delivered. The same or different endpoints MAY be used for requests and responses.

1064 **4.4.3.1 <LogoutRequest> issued by Service Provider to Identity Provider**

1065 If the logout profile is initiated by a service provider, it examines the authentication assertion(s) it received
1066 pertaining to the local session(s) being terminated, and collects the `SessionIndex` value(s) it received
1067 from the identity provider. If multiple identity providers are involved, then the profile MUST be repeated
1068 independently for each one.

1069 To initiate the profile, the service provider issues a <LogoutRequest> message to the identity provider's
1070 single logout service request endpoint containing one or more applicable <SessionIndex> elements. At
1071 least one element MUST be included. Metadata (as in [SAMLMeta]) MAY be used to determine the
1072 location of this endpoint and the bindings supported by the identity provider.

1073 **Synchronous Bindings (Back-Channel)**

1074 The service provider MAY use a synchronous binding, such as the SOAP binding [SAMLBind], to
1075 send the request directly to the identity provider. The identity provider would then propagate any
1076 required logout messages to additional service providers as required using a synchronous
1077 binding. The requester MUST authenticate itself to the identity provider, either by signing the
1078 <LogoutRequest> or using any other binding-supported mechanism.

1079 **Asynchronous Bindings (Front-Channel)**

1080 Alternatively, the service provider MAY (if the principal's user agent is present) use an
1081 asynchronous binding, such as the HTTP Redirect, POST, or Artifact bindings [SAMLBind] to
1082 send the request to the identity provider through the user agent.

1083 If the HTTP Redirect or POST binding is used, then the <LogoutRequest> message is
1084 delivered to the identity provider in this step. If the HTTP Artifact binding is used, the Artifact
1085 Resolution profile defined in section 5 is used by the identity provider, which makes a callback to
1086 the service provider to retrieve the <LogoutRequest> message, using for example the SOAP
1087 binding.

1088 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0
1089 ([SSL3]) or TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The
1090 <LogoutRequest> message MUST be signed if the HTTP POST or Redirect binding is used.
1091 The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the
1092 request issuer when the artifact is dereferenced.

1093 Each of these bindings provide a RelayState mechanism that the service provider MAY use to
1094 associate the profile exchange with the original request. The service provider SHOULD reveal as
1095 little information as possible in the RelayState value unless the use of the profile does not require
1096 such privacy measures.

1097 Profile-specific rules for the contents of the <LogoutRequest> message are included in section 4.4.4.1.

1098 **4.4.3.2 Identity Provider determines Session Participants**

1099 If the logout profile is initiated by an identity provider, or upon receiving a valid <LogoutRequest>
1100 message, the identity provider processes the request as defined in [SAMLCore]. It MUST examine the
1101 principal identifier and <SessionIndex> elements and determine the set of sessions to be terminated.

1102 The identity provider then follows steps 3 and 4 for each entity participating in the session(s) being
1103 terminated, other than the original requesting service provider (if any), as described in section 3.7.3.2 of
1104 [SAMLCore].

1105 **4.4.3.3 <LogoutRequest> issued by Identity Provider to Session 1106 Participant/Authority**

1107 To propagate the logout, the identity provider issues its own <LogoutRequest> to a session authority or
1108 participant in a session being terminated. The request is sent in the same fashion as described in step 1
1109 using a SAML binding consistent with the capability of the responder and the availability of the user agent
1110 at the identity provider.

1111 Profile-specific rules for the contents of the <LogoutRequest> message are included in section 4.4.4.1.

1112 **4.4.3.4 Session Participant/Authority issues <LogoutResponse> to Identity 1113 Provider**

1114 The session participant/authority MUST process the <LogoutRequest> message as defined in
1115 [SAMLCore]. After processing the message or upon encountering an error, the entity MUST issue a
1116 <LogoutResponse> message containing an appropriate status code to the requesting identity provider
1117 to complete the SAML protocol exchange.

1118 Synchronous Bindings (Back-Channel)

1119 If the identity provider used a synchronous binding, such as the SOAP binding [SAMLBind], the
1120 response is returned directly to complete the synchronous communication. The responder **MUST**
1121 authenticate itself to the requesting identity provider, either by signing the <LogoutResponse> or
1122 using any other binding-supported mechanism.

1123 Asynchronous Bindings (Front-Channel)

1124 If the identity provider used an asynchronous binding, such as the HTTP Redirect, POST, or
1125 Artifact bindings [SAMLBind], then the <LogoutResponse> (or artifact) is returned through the
1126 user agent to the identity provider's single logout service response endpoint. Metadata (as in
1127 [SAMLMeta]) **MAY** be used to determine the location of this endpoint and the bindings supported
1128 by the identity provider.

1129 If the HTTP Redirect or POST binding is used, then the <LogoutResponse> message is
1130 delivered to the identity provider in this step. If the HTTP Artifact binding is used, the Artifact
1131 Resolution profile defined in section 5 is used by the identity provider, which makes a callback to
1132 the responding entity to retrieve the <LogoutResponse> message, using for example the SOAP
1133 binding.

1134 It is **RECOMMENDED** that the HTTP exchanges in this step be made over either SSL 3.0
1135 ([SSL3]) or TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The
1136 <LogoutResponse> message **MUST** be signed if the HTTP POST or Redirect binding is used.
1137 The HTTP Artifact binding, if used, also provides for an alternate means of authenticating the
1138 response issuer when the artifact is dereferenced.

1139 Profile-specific rules for the contents of the <LogoutResponse> message are included in section
1140 4.4.4.2.

1141 4.4.3.5 Identity Provider issues <LogoutResponse> to Service Provider

1142 After processing the original service provider's <LogoutRequest> in step 1, or upon encountering an
1143 error, the identity provider **MUST** respond to the original request with a <LogoutResponse> containing
1144 an appropriate status code to complete the SAML protocol exchange.

1145 The response is sent to the original service provider in the same fashion as described in step 4, using a
1146 SAML binding consistent with the binding used in the request, the capability of the responder, and the
1147 availability of the user agent at the identity provider.

1148 Profile-specific rules for the contents of the <LogoutResponse> message are included in section
1149 4.4.4.2.

1150 4.4.4 Use of Single Logout Protocol

1151 4.4.4.1 <LogoutRequest> Usage

1152 The <Issuer> element **MUST** be present and **MUST** contain the unique identifier of the requesting entity;
1153 the `Format` attribute **MUST** be omitted or have a value of `urn:oasis:names:tc:SAML:2.0:nameid-
1154 format:entity`.

1155 The requester **MUST** authenticate itself to the responder and ensure message integrity, either by signing
1156 the message or using a binding-specific mechanism.

1157 The principal **MUST** be identified in the request using an identifier that **strongly matches** the identifier in
1158 the authentication assertion the requester issued or received regarding the session being terminated, per
1159 the matching rules defined in section 3.3.4 of [SAMLCore].

1160 If the requester is a session participant, it **MUST** include at least one <SessionIndex> element in the

1161 request. If the requester is a session authority (or acting on its behalf), then it MAY omit any such
1162 elements to indicate the termination of all of the principal's applicable sessions.

1163 **4.4.4.2 <LogoutResponse> Usage**

1164 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding
1165 entity; the Format attribute MUST be omitted or have a value of
1166 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.

1167 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1168 the message or using a binding-specific mechanism.

1169 **4.4.5 Use of Metadata**

1170 [SAMLMeta] defines an endpoint element, <md:SingleLogoutService>, to describe supported
1171 bindings and location(s) to which an entity may send requests and responses using this profile.

1172 A requester, if encrypting the principal's identifier, can use the responder's <md:KeyDescriptor>
1173 element with a use attribute of encryption to determine an appropriate encryption algorithm and
1174 settings to use, along with a public key to use in delivering a bulk encryption key.

1175 **4.5 Name Identifier Management Profile**

1176 In the scenario supported by the Name Identifier Management profile, an identity provider has exchanged
1177 some form of persistent identifier for a principal with a service provider, allowing them to share a common
1178 identifier for some length of time. Subsequently, the identity provider may wish to notify the service
1179 provider of a change in the format and/or value that it will use to identify the same principal in the future.
1180 Alternatively the service provider may wish to attach its own "alias" for the principal in order to insure that
1181 the identity provider will include it when communicating with it in the future about the principal. Finally, one
1182 of the providers may wish to inform the other that it will no longer issue or accept messages using a
1183 particular identifier. To implement these scenarios, a profile of the SAML Name Identifier Management
1184 protocol is used.

1185 The profile allows the protocol to be combined with a synchronous binding, such as the SOAP binding, or
1186 with asynchronous "front-channel" bindings, such as the HTTP Redirect, POST, or Artifact bindings. A
1187 front-channel binding may be required, for example, in cases in which direct interaction between the user
1188 agent and the responding provider is required in order to effect the change.

1189 **4.5.1 Required Information**

1190 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:SSO:nameid-mgmt

1191 **Contact information:** security-services-comment@lists.oasis-open.org

1192 **Description:** Given below.

1193 **Updates:** None

1194 **4.5.2 Profile Overview**

1195 The following figure illustrates the basic template for the name identifier management profile.

1196 <need figure>

1197 The following steps are described by the profile. Within an individual step, there may be one or more
1198 actual message exchanges depending on the binding used for that step and other implementation-
1199 dependent behavior.

1200 | **1. <ManageNameIDRequest> issued by Requesting Identity/Service Provider**

1201 | In step 1, an identity or service provider initiates the profile by sending a
1202 | <ManageNameIDRequest> message to another provider that it wishes to inform of a change.
1203 | The request may be sent directly to the responding provider or sent indirectly through the user
1204 | agent.

1205 | **2. <ManageNameIDResponse> issued by Responding Identity/Service Provider**

1206 | In step 2, the responding provider (after processing the request) issues a
1207 | <ManageNameIDResponse> message to the original requesting provider. The response may be
1208 | returned directly to the requesting provider or indirectly through the user agent (if consistent with
1209 | the form of the request in step 1).

1210 | **4.5.3 Profile Description**

1211 | In the descriptions below, the following is referred to:

1212 | **Name Identifier Management Service**

1213 | This is the name identifier management protocol endpoint at an identity or service provider to
1214 | which the <ManageNameIDRequest> or <ManageNameIDResponse> messages (or an artifact
1215 | representing them) are delivered. The same or different endpoints MAY be used for requests and
1216 | responses.

1217 | **4.5.3.1 <ManageNameIDRequest> issued by Requesting Identity/Service Provider**

1218 | To initiate the profile, the requesting provider issues a <ManageNameIDRequest> message to another
1219 | provider's name identifier management service request endpoint. Metadata (as in [SAMLMeta]) MAY be
1220 | used to determine the location of this endpoint and the bindings supported by the responding provider.

1221 | **Synchronous Bindings (Back-Channel)**

1222 | The requesting provider MAY use a synchronous binding, such as the SOAP binding [SAMLBind],
1223 | to send the request directly to the other provider. The requester MUST authenticate itself to the
1224 | other provider, either by signing the <ManageNameIDRequest> or using any other binding-
1225 | supported mechanism.

1226 | **Asynchronous Bindings (Front-Channel)**

1227 | Alternatively, the requesting provider MAY (if the principal's user agent is present) use an
1228 | asynchronous binding, such as the HTTP Redirect, POST, or Artifact bindings [SAMLBind] to
1229 | send the request to the other provider through the user agent.

1230 | If the HTTP Redirect or POST binding is used, then the <ManageNameIDRequest> message is
1231 | delivered to the other provider in this step. If the HTTP Artifact binding is used, the Artifact
1232 | Resolution profile defined in section 5 is used by the other provider, which makes a callback to the
1233 | requesting provider to retrieve the <ManageNameIDRequest> message, using for example the
1234 | SOAP binding.

1235 | It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0
1236 | ([SSL3]) or TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The
1237 | <ManageNameIDRequest> message MUST be signed if the HTTP POST or Redirect binding is
1238 | used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating
1239 | the request issuer when the artifact is dereferenced.

1240 | Each of these bindings provide a RelayState mechanism that the requesting provider MAY use to
1241 | associate the profile exchange with the original request. The requesting provider SHOULD reveal
1242 | as little information as possible in the RelayState value unless the use of the profile does not
1243 | require such privacy measures.

1244 Profile-specific rules for the contents of the <ManageNameIDRequest> message are included in section
1245 4.4.4.1.

1246 **4.5.3.2 <ManageNameIDResponse> issued by Responding Identity/Service** 1247 **Provider**

1248 The recipient MUST process the <ManageNameIDRequest> message as defined in [SAMLCore]. After
1249 processing the message or upon encountering an error, the recipient MUST issue a
1250 <ManageNameIDResponse> message containing an appropriate status code to the requesting provider
1251 to complete the SAML protocol exchange.

1252 **Synchronous Bindings (Back-Channel)**

1253 If the requesting provider used a synchronous binding, such as the SOAP binding [SAMLBind],
1254 the response is returned directly to complete the synchronous communication. The responder
1255 MUST authenticate itself to the requesting provider, either by signing the
1256 <ManageNameIDResponse> or using any other binding-supported mechanism.

1257 **Asynchronous Bindings (Front-Channel)**

1258 If the requesting provider used an asynchronous binding, such as the HTTP Redirect, POST, or
1259 Artifact bindings [SAMLBind], then the <ManageNameIDResponse> (or artifact) is returned
1260 through the user agent to the requesting provider's name identifier management service response
1261 endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the location of this endpoint
1262 and the bindings supported by the requesting provider.

1263 If the HTTP Redirect or POST binding is used, then the <ManageNameIDResponse> message is
1264 delivered to the requesting provider in this step. If the HTTP Artifact binding is used, the Artifact
1265 Resolution profile defined in section 5 is used by the requesting provider, which makes a callback
1266 to the responding provider to retrieve the <ManageNameIDResponse> message, using for
1267 example the SOAP binding.

1268 It is RECOMMENDED that the HTTP exchanges in this step be made over either SSL 3.0
1269 ([SSL3]) or TLS 1.0 ([RFC2246]) to maintain confidentiality and message integrity. The
1270 <ManageNameIDResponse> message MUST be signed if the HTTP POST or Redirect binding is
1271 used. The HTTP Artifact binding, if used, also provides for an alternate means of authenticating
1272 the response issuer when the artifact is dereferenced.

1273 Profile-specific rules for the contents of the <ManageNameIDResponse> message are included in
1274 section 4.4.4.2.

1275 **4.5.4 Use of Name Identifier Management Protocol**

1276 **4.5.4.1 <ManageNameIDRequest> Usage**

1277 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting entity;
1278 the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-
1279 format:entity.

1280 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing
1281 the message or using a binding-specific mechanism.

1282 **4.5.4.2 <ManageNameIDResponse> Usage**

1283 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding
1284 entity; the Format attribute MUST be omitted or have a value of
1285 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.

1286 | The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1287 | the message or using a binding-specific mechanism.

1288 | **4.5.5 Use of Metadata**

1289 | [SAMLMeta] defines an endpoint element, <md:ManageNameIDService>, to describe supported
1290 | bindings and location(s) to which an entity may send requests and responses using this profile.

1291 | A requester, if encrypting the principal's identifier, can use the responder's <md:KeyDescriptor>
1292 | element with a use attribute of encryption to determine an appropriate encryption algorithm and
1293 | settings to use, along with a public key to use in delivering a bulk encryption key.

5 Artifact Resolution Profile

1294

1295 [SAMLCore] defines an Artifact Resolution protocol for dereferencing a SAML artifact into a corresponding
1296 protocol message. The HTTP Artifact binding in [SAMLBind] leverages this mechanism to pass SAML
1297 protocol messages by reference. This profile describes the use of this protocol with a synchronous
1298 binding, such as the SOAP binding defined in [SAMLBind].

5.1 Required Information

1299

1300 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:artifact

1301 **Contact information:** security-services-comment@lists.oasis-open.org

1302 **Description:** Given below.

1303 **Updates:** None

5.2 Profile Overview

1304

1305 The message exchange and basic processing rules that govern this profile are largely defined by section
1306 3.5 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to
1307 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to
1308 SOAP 1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1309 The following figure illustrates the basic template for the artifact resolution profile.

1310 <need figure>

1311 The following steps are described by the profile.

1. <ArtifactResolve> issued by Requesting Entity

1312

1313 In step 1, a requester initiates the profile by sending an <ArtifactResolve> message to an
1314 artifact issuer.

2. <ArtifactResponse> issued by Responding Entity

1315

1316 In step 2, the responder (after processing the request) issues an <ArtifactResponse>
1317 message to the requester.

5.3 Profile Description

1318

1319 In the descriptions below, the following is referred to:

Artifact Resolution Service

1320

1321 This is the artifact resolution protocol endpoint at an artifact issuer to which
1322 <ArtifactResolve> messages are delivered.

5.3.1 <ArtifactResolve> issued by Requesting Entity

1323

1324 To initiate the profile, a requester, having received an artifact and determined the issuer using the
1325 SourceID, sends an <ArtifactResolve> message containing the artifact to an artifact issuer's artifact
1326 resolution service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the location of this
1327 endpoint and the bindings supported by the artifact issuer

1328 The requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send the
1329 request directly to the artifact issuer. The requester SHOULD authenticate itself to the identity provider,
1330 either by signing the <ArtifactResolve> message or using any other binding-supported mechanism.
1331 Specific profiles that use the HTTP Artifact binding MAY impose additional requirements such that
1332 authentication is mandatory.

1333 Profile-specific rules for the contents of the <ArtifactResolve> message are included in section 5.4.1.

1334 **5.3.2 <ArtifactResponse> issued by Responding Entity**

1335 The artifact issuer MUST process the <ArtifactResolve> message as defined in [SAMLCore]. After
1336 processing the message or upon encountering an error, the artifact issuer MUST return an
1337 <ArtifactResponse> message containing an appropriate status code to the requester to complete the
1338 SAML protocol exchange. If successful, the dereferenced SAML protocol message corresponding to the
1339 artifact will also be included.

1340 The responder MUST authenticate itself to the requester, either by signing the <ArtifactResponse> or
1341 using any other binding-supported mechanism.

1342 Profile-specific rules for the contents of the <ArtifactResponse> message are included in section
1343 5.4.2.

1344 **5.4 Use of Artifact Resolution Protocol**

1345 **5.4.1 <ArtifactResolve> Usage**

1346 The <Issuer> element MUST be present and MUST contain the unique identifier of the requesting entity;
1347 the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-
1348 format:entity.

1349 The requester SHOULD authenticate itself to the responder and ensure message integrity, either by
1350 signing the message or using a binding-specific mechanism. Specific profiles that use the HTTP Artifact
1351 binding MAY impose additional requirements such that authentication is mandatory.

1352 **5.4.2 <ArtifactResponse> Usage**

1353 The <Issuer> element MUST be present and MUST contain the unique identifier of the artifact issuer;
1354 the Format attribute MUST be omitted or have a value of urn:oasis:names:tc:SAML:2.0:nameid-
1355 format:entity.

1356 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1357 the message or using a binding-specific mechanism.

1358 **5.5 Use of Metadata**

1359 [SAMLMeta] defines an indexed endpoint element, <md:ArtifactResolutionService>, to describe
1360 supported bindings and location(s) to which a requester may send requests using this profile. The index
1361 attribute is used to distinguish the possible endpoints that may be specified by reference in the artifact's
1362 EndpointIndex field.

1363 6 Assertion Request/Query/Request -Profile

1364 [SAMLCore] defines a protocol for requesting existing assertions by reference or by querying on the basis
1365 of a subject and additional statement-specific criteria. This profile describes the use of this protocol with a
1366 synchronous binding, such as the SOAP binding defined in [SAMLBind].

1367 6.1 Required Information

1368 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:query

1369 **Contact information:** security-services-comment@lists.oasis-open.org

1370 **Description:** Given below.

1371 **Updates:** None.

1372 6.2 Profile Overview

1373 The message exchange and basic processing rules that govern this profile are largely defined by section
1374 3.3 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to
1375 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to
1376 SOAP 1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1377 The following figure illustrates the basic template for the query/request profile.

1378 <need figure>

1379 The following steps are described by the profile.

1380 1. Query/Request issued by Requesting Entity

1381 In step 1, a requester initiates the profile by sending an <AssertionIDRequest>.
1382 <SubjectQuery>, <AuthnQuery>, <AttributeQuery>, or <AuthzDecisionQuery>
1383 message to a SAML authority.

1384 2. <Response> issued by SAML Authority

1385 In step 2, the responding SAML authority (after processing the query or request) issues a
1386 <Response> message to the requester.

1387 6.3 Profile Description

1388 In the descriptions below, the following are referred to:

1389 Query/Request Service

1390 This is the query/request protocol endpoint at a SAML authority to which query or
1391 <AssertionIDRequest> messages are delivered.

1392 6.3.1 Query/Request issued by Requesting Entity

1393 To initiate the profile, a requester issues an <AssertionIDRequest>, <SubjectQuery>.
1394 <AuthnQuery>, <AttributeQuery>, or <AuthzDecisionQuery> message to a SAML authority's
1395 query/request service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the location of
1396 this endpoint and the bindings supported by the SAML authority.

1397 | The requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send the
1398 | request directly to the identity provider. The requester SHOULD authenticate itself to the SAML authority
1399 | either by signing the message or using any other binding-supported mechanism.

1400 | Profile-specific rules for the contents of the various messages are included in section 6.4.1.

1401 | **6.3.2 <Response> issued by SAML Authority**

1402 | The SAML authority MUST process the query or request message as defined in [SAMLCore]. After
1403 | processing the message or upon encountering an error, the SAML authority MUST return a <Response>
1404 | message containing an appropriate status code to the requester to complete the SAML protocol
1405 | exchange. If the request is successful in locating one or more matching assertions, they will also be
1406 | included in the response.

1407 | The responder SHOULD authenticate itself to the requester, either by signing the <Response> or using
1408 | any other binding-supported mechanism.

1409 | Profile-specific rules for the contents of the <Response> message are included in section 6.4.2.

1410 | **6.4 Use of Query/Request Protocol**

1411 | **6.4.1 Query/Request Usage**

1412 | The <Issuer> element MUST be present.

1413 | The requester SHOULD authenticate itself to the responder and ensure message integrity, either by
1414 | signing the message or using a binding-specific mechanism.

1415 | **6.4.2 <Response> Usage**

1416 | The <Issuer> element MUST be present and MUST contain the unique identifier of the responding
1417 | SAML authority; the Format attribute MUST be omitted or have a value of
1418 | urn:oasis:names:tc:SAML:2.0:nameid-format:entity. Note that this need not necessarily
1419 | match the <Issuer> element in the returned assertion(s).

1420 | The responder SHOULD authenticate itself to the requester and ensure message integrity, either by
1421 | signing the message or using a binding-specific mechanism.

1422 | **6.5 Use of Metadata**

1423 | [SAMLMeta] defines several endpoint elements, <md:AssertionIDRequestService>,
1424 | <md:AuthnQueryService>, <md:AttributeService>, and <md:AuthzService>, to describe
1425 | supported bindings and location(s) to which a requester may send requests or queries using this profile.

1426 | The SAML authority, if encrypting the resulting assertions or assertion contents for a particular entity, can
1427 | use that entity's <md:KeyDescriptor> element with a use attribute of encryption to determine an
1428 | appropriate encryption algorithm and settings to use, along with a public key to use in delivering a bulk
1429 | encryption key.

1430 7 Name Identifier Mapping Profile

1431 [SAMLCore] defines a Name Identifier Mapping protocol for mapping a principal's name identifier into a
1432 different name identifier for the same principal. This profile describes the use of this protocol with a
1433 synchronous binding, such as the SOAP binding defined in [SAMLBind], and additional guidelines for
1434 protecting the privacy of the principal with encryption and limiting the use of the mapped identifier.

1435 7.1 Required Information

1436 **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:nameidmapping

1437 **Contact information:** security-services-comment@lists.oasis-open.org

1438 **Description:** Given below.

1439 **Updates:** None.

1440 7.2 Profile Overview

1441 The message exchange and basic processing rules that govern this profile are largely defined by section
1442 3.89 of [SAMLCore] that defines the messages to be exchanged, in combination with the binding used to
1443 exchange the messages. Section 3.2 of [SAMLBind] defines the binding of the message exchange to
1444 SOAP 1.1. Unless specifically noted here, all requirements defined in those specifications apply.

1445 The following figure illustrates the basic template for the name identifier mapping profile.

1446 <need figure>

1447 The following steps are described by the profile.

1448 1. <NameIDMappingRequest> issued by Requesting Entity

1449 In step 1, a requester initiates the profile by sending a <NameIDMappingRequest> message to
1450 an identity provider.

1451 2. <NameIDMappingResponse> issued by Identity Provider

1452 In step 2, the responding identity provider (after processing the request) issues a
1453 <NameIDMappingResponse> message to the requester.

1454 7.3 Profile Description

1455 In the descriptions below, the following is referred to:

1456 Name Identifier Mapping Service

1457 This is the name identifier mapping protocol endpoint at an identity provider to which
1458 <NameIDMappingRequest> messages are delivered.

1459 7.3.1 <NameIDMappingRequest> issued by Requesting Entity

1460 To initiate the profile, a requester issues a <NameIDMappingRequest> message to an identity provider's
1461 name identifier mapping service endpoint. Metadata (as in [SAMLMeta]) MAY be used to determine the
1462 location of this endpoint and the bindings supported by the identity provider.

1463 The requester MUST use a synchronous binding, such as the SOAP binding [SAMLBind], to send the

1464 request directly to the identity provider. The requester MUST authenticate itself to the identity provider,
1465 either by signing the <NameIDMappingRequest> or using any other binding-supported mechanism.

1466 Profile-specific rules for the contents of the <NameIDMappingRequest> message are included in
1467 section 7.4.1.

1468 **7.3.2 <NameIDMappingResponse> issued by Identity Provider**

1469 The identity provider MUST process the <ManageNameIDRequest> message as defined in [SAMLCore].
1470 After processing the message or upon encountering an error, the identity provider MUST return a
1471 <NameIDMappingResponse> message containing an appropriate status code to the requester to
1472 complete the SAML protocol exchange.

1473 The responder MUST authenticate itself to the requester, either by signing the
1474 <NameIDMappingResponse> or using any other binding-supported mechanism.

1475 Profile-specific rules for the contents of the <NameIDMappingResponse> message are included in
1476 section 7.4.2. In this profile, a requester uses a synchronous binding to send a
1477 <NameIDMappingRequest> message directly to an identity provider containing a name identifier for a
1478 principal that is shared between them. Note that this identifier need not itself be persistent, and MAY be
1479 encrypted (perhaps obtained from the previous use of this profile).

1480 The requester MUST authenticate to the identity provider, either using a mechanism permitted by the
1481 binding, or by signing the <NameIDMappingRequest> message. The <Issuer> element MUST be
1482 present in the request.

1483 If the identity provider receiving the request recognizes the principal, can support the requester's
1484 <NameIDPolicy> for that principal, and is willing to fulfill the request based on authentication of the
1485 requester and any applicable policies, then it responds with a successful <NameIDMappingResponse>
1486 in the binding-specific response containing the requested name identifier. The resulting identifier MAY be
1487 encrypted and time-limited, as described below. Otherwise the response will contain an error status code.

1488 The responding identity provider MUST authenticate to the requester, either using a mechanism permitted
1489 by the binding, or by signing the <NameIDMappingResponse> message.

1490 The <Issuer> element MUST be present in the response and MUST contain the unique identifier of the
1491 responding identity provider; the Format attribute MUST be omitted or have a value of
1492 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.

1493 **7.4 Use of Name Identifier Mapping Protocol**

1494 **7.4.1 <NameIDMappingRequest> Usage**

1495 The <Issuer> element MUST be present.

1496 The requester MUST authenticate itself to the responder and ensure message integrity, either by signing
1497 the message or using a binding-specific mechanism.

1498 **7.4.2 <NameIDMappingResponse> Usage**

1499 The <Issuer> element MUST be present and MUST contain the unique identifier of the responding
1500 identity provider; the Format attribute MUST be omitted or have a value of
1501 urn:oasis:names:tc:SAML:2.0:nameid-format:entity.

1502 The responder MUST authenticate itself to the requester and ensure message integrity, either by signing
1503 the message or using a binding-specific mechanism.

1504 | **Use of Encryption**

1505 | Section 2.3.3 of [SAMLCore] defines the use of encryption to apply confidentiality to a name identifier. In
1506 | most cases, the identity provider SHOULD encrypt the mapped name identifier it returns to the requester
1507 | to protect the privacy of the principal. The requester **may** extract the <EncryptedID> element and
1508 | place it in subsequent protocol messages or assertions.

1509 | **7.4.2.1 Limiting Use of Mapped Identifier**

1510 | Additional limits on the use of the resulting identifier MAY be applied by the identity provider by returning
1511 | the mapped name identifier in the form of an <Assertion> containing the identifier in its <Subject> but
1512 | without any statements. The assertion is then encrypted and the result used as the <EncryptedData>
1513 | element in the <EncryptedID> returned to the requester. The assertion MAY include a <Conditions>
1514 | element to limit use, as defined by [SAMLCore], such as time-based constraints or use by specific relying
1515 | parties, and MUST be signed for integrity protection.

1516 | **7.5 Use of Metadata**

1517 | [SAMLMeta] defines an endpoint element, <md:NameIDMappingService>, to describe **supported**
1518 | **bindings** ~~thead~~ location(s) to which a requester may send requests using this profile.

1519 | The identity provider, if encrypting the resulting identifier for a particular entity, can use that entity's
1520 | <md:KeyDescriptor> element with a use attribute of encryption to determine an appropriate
1521 | encryption algorithm and settings to use, along with a public key to use in delivering a bulk encryption key.

1522 8 **SAML Attribute Profiles**

1523 8.1 **Guidelines**

1524 ~~This section provides a checklist of items that MUST be addressed by each attribute profile.~~

1525 6. ~~A human-readable string name for the profile.~~

1526 7. ~~Unique URI to be used for the NameFormat attribute of the <saml:AttributeDescriptor>~~
1527 ~~element.~~

1528 8. ~~Syntax and restrictions on class of strings acceptable as the value of the name attribute of the~~
1529 ~~<saml:AttributeDescriptor> element when the selected NameFormat attribute value is present.~~

1530 9. ~~Additional attributes (together with required namespaces) defined by the profile that may be used with~~
1531 ~~the <saml:AttributeDescriptor> element.~~

1532 10. ~~Rules for determining the equality of <saml:AttributeDescriptor> elements as defined by the~~
1533 ~~profile.~~

1534 11. ~~Syntax and restrictions on values acceptable in a <saml:AttributeValue> element, when the~~
1535 ~~selected NameFormat attribute value is present and whether the xsi:type attribute must be present.~~

1536 12. ~~Additional XML attributes (together with required namespaces) defined by the profile that may be used~~
1537 ~~with the <saml:Attribute> element.~~

1538 8.2 **Basic Attribute Profile**

1539 ~~The Basic attribute profile specifies simplified, but non-unique, naming of SAML attributes together with~~
1540 ~~attribute values based on the built-in XML Schema data types, eliminating the need for extension schemas~~
1541 ~~to validate syntax.~~

1542 8.2.1 **Required Information**

1543 ~~**Identification:** <urn:oasis:names:tc:SAML:2.0:profiles:attribute:basic>~~

1544 ~~**Contact information:** security-services-comment@lists.oasis-open.org~~

1545 ~~**Description:** Given below.~~

1546 ~~**Updates:** None.~~

1547 8.2.2 **NameFormat Value**

1548 ~~URI: <urn:oasis:names:tc:SAML:2.0:attribute-name:basic>~~

1549 8.2.3 **SAML Attribute Naming**

1550 ~~The NameFormat XML attribute in <AttributeDescriptor> and <Attribute> elements MUST be~~
1551 ~~<urn:oasis:names:tc:SAML:2.0:attrname-format:basic>.~~

1552 ~~The class of strings acceptable as the value of the Name attribute of the~~
1553 ~~<saml:AttributeDescriptor> and <saml:Attribute> elements MUST be drawn from the set of~~

1554 | ~~values belonging to the primitive type **Name** as defined in Section 3.3.6 of [XML-Schema-Part2]. The~~
1555 | ~~Name XML attribute MUST adhere to the rules specified for that format, as defined by [SAMLCore].~~

1556 | ~~No additional XML attributes are defined for the <saml:Attribute> or~~
1557 | ~~<saml:AttributeDesignator> elements.~~

1558 | **8.2.4 Profile-Specific XML Attributes**

1559 | ~~No additional XML attributes are defined for use with the <AttributeDesignator> or <Attribute>~~
1560 | ~~elements.~~

1561 | **8.2.5 <AttributeDesignator> Comparison**

1562 | ~~Two <AttributeDesignator> elements are equal if and only if the values of their Name XML attributes~~
1563 | ~~are equal in the sense of Section 3.3.6 of [XML-Schema-Part2].~~

1564 | **8.2.6 SAML Attribute Values**

1565 | The schema type of the contents of the <saml:AttributeValue> element MUST be drawn from one of
1566 | the ~~values of the~~ types defined in Section 3.3 of [XML-Schema-Part2][XML-Schema-Part2]. The
1567 | xsi:type attribute MUST be present and be given the appropriate value.

1568 | **8.2.7 AttributeDesignator Comparison**

1569 | ~~Two <saml:AttributeDesignator> elements are equal if and only if the values of the Name XML~~
1570 | ~~attributes are equal in the sense of Section 3.3.6 of [XML-Schema-Part2].~~

1571 | **8.2.8 Example**

1572 | TBD

1573 | **8.3 X.500/LDAP Attribute Profile**

1574 | There is a substantial body of work describing standard syntaxes for X.500/LDAP attributes. This includes
1575 | RFC2256 [RFC2256], which describes an overview of the attribute types and object classes defined by
1576 | the ISO and ITU-T committees in the X.500 documents, in particular those intended for use by directory
1577 | clients. Several authors have built upon these approaches to develop additional attribute types and some
1578 | of these have been widely implemented. For example, the **inetOrgPerson** object class defined in
1579 | RFC2798 [RFC2798] has received wide implementation amongst LDAP vendors. Other efforts include the
1580 | definition of the eduPerson object class by the EDUCAUSE/Internet2 task force [eduPersonSchema].

1581 | The X.500/LDAP attribute profile standardizes the naming and representation of such attributes when
1582 | expressed as SAML attributes, providing unique naming consistent with the OID mechanism used natively
1583 | by such specifications.

1584 | **8.3.1 Required Information**

1585 | **Identification:** urn:oasis:names:tc:SAML:2.0:profiles:attribute:LDAP

1586 | **Contact information:** security-services-comment@lists.oasis-open.org

1587 | **Description:** Given below.

1588 | **Updates:** None.

1589 **8.3.2 SAML NameFormat ValueAttribute Naming**

1590 ~~The NameFormat XML attribute in <AttributeDesignator> and <Attribute> elements MUST be~~
1591 ~~urn:oasis:names:tc:SAML:2.0:attrname-format:uri, URI:~~
1592 ~~urn:oasis:names:tc:SAML:2.0:attribute-name:X500-LDAP~~

1593 **8.3.3 Attribute Names**

1594 ~~To construct attribute names, Following [Morgan], we adopt~~ the URN `oid` namespace described in
1595 ~~[RFC3061] is used.~~ In this approach the ~~Attribute-nName XML attribute~~ is based on the OID assigned to
1596 the X.500/LDAP attribute type.

1597 Example:

```
1598 urn:oid:1.3.6.1.4.1.299
```

1599 X.500 conventions require that every object-class be identified with a unique OID. This ensures ~~that~~
1600 attribute names are unambiguous.-

1601 For purposes of human readability, there ~~is may~~ also ~~be~~ a requirement ~~for some applications~~ to carry an
1602 optional string name together with the OID ~~URN~~. This ~~modeled by the~~ optional XML attribute
1603 `FriendlyName` ~~with namespace urn:oasis:names:tc:SAML:2.0:attribute-name:X500-LDAP. The~~
1604 ~~FriendlyName attribute is drawn from [RFCXXXX]. (defined in [SAMLCore])~~ MAY be used for this
1605 ~~purpose.~~

1606 **8.3.4 Profile-Specific XML Attributes**

1607 ~~No additional XML attributes are defined for use with the <AttributeDesignator> or <Attribute>~~
1608 ~~elements.~~

1609 **8.3.5 <AttributeDesignator> Comparison**

1610 ~~Two <AttributeDesignator> elements are equal if and only if their Name XML attribute values are~~
1611 ~~equal in the sense of [RFC3061]. The FriendlyName attribute plays no role in the comparison.~~

1612 **8.3.6 SAML Attribute Values**

1613 ~~We need to define a convention for carrying different attribute syntaxes within XML. [RFC2252] explains~~
1614 ~~that octet strings are the canonical representation for X.500/LDAP attribute syntaxes. The primitive type~~
1615 ~~hexbinary (Section 3.2.15 of [XML Schema Part2] allows octet strings to be represented by their~~
1616 ~~hexadecimal representation. The hexbinary type MUST be used when <saml:AttributeValue> contains an~~
1617 ~~X.500/LDAP attribute value. The canonical representation for X.500/LDAP attribute syntaxes is an octet~~
1618 ~~string. However, to simplify the job of the attribute consumer, any X.500/LDAP attribute syntax that can~~
1619 ~~easily be expressed in string form SHOULD be passed as a string within the <AttributeValue>~~
1620 ~~element, with no additional whitespace. In such cases, the xsi:type XML attribute MUST be set to~~
1621 ~~xsd:string.~~

1622 ~~Examples of such attribute syntaxes are those with string, numeric, or date/time values. Date/time values~~
1623 ~~MUST be expressed in UTC form.~~

1624 ~~Any attribute value (particularly those without a reasonable string form, such as binary data) MAY be~~
1625 ~~passed by base64-encoding the octet string and specifying an xsi:type XML attribute of~~
1626 ~~xsd:base64Binary. The xsi:type XML attribute MUST be present in such cases.~~

1627 ~~As additional standards in the expression of X.500/LDAP attribute syntaxes in XML form develop, this~~
1628 ~~profile may evolve to incorporate such approaches.~~

1629 **8.3.7 AttributeDesignator Comparison**

1630 ~~<saml:AttributeDesignator> elements are equal if and only if the Name attribute values are equal in~~
1631 ~~the sense of [RFC3061]. The FriendlyName attribute plays no role in the comparison.~~

1632 **8.3.8 Example**

1633 TBD

1634 **8.4 UUID Attribute Profile**

1635 The UUID attribute profile standardizes the expression of UUID values as SAML attribute names and
1636 values. It is applicable when the attribute's source system is one that identifies an attribute with a UUID.
1637 The value of the attribute may also be a UUID, but need not be.

1638 **8.4.1 Required Information**

1639 **Identification:** <urn:oasis:names:tc:SAML:2.0:profiles:attribute:UUID>

1640 **Contact information:** security-services-comment@lists.oasis-open.org

1641 **Description:** [Given below.](#)

1642 **Updates:** [None.](#)

1643 **8.4.2 UUIDs and GUIDs Background**

1644 UUIDs (Universally Unique Identifiers), also known as GUIDs (Globally Unique Identifiers), are used to
1645 define objects and subjects such that they are guaranteed uniqueness across space and time. -UUIDs
1646 were originally used in the Network Computing System (NCS), and then used in the Open Software
1647 Foundation's (OSF) Distributed Computing Environment (DCE). -Recently GUIDs have been used in
1648 Microsoft's [COM and](#) Active Directory/Windows 2000/2003 platform.

1649 A UUID is a 128 bit number, generated such that it should never be duplicated within the domain of
1650 interest. -UUIDs are used to represent a wide range of objects including, but not limited to, subjects/users,
1651 groups of users and node names. -A UUID, represented as a hexadecimal string, is as follows:

1652 `f81d4fae-7dec-11d0-a765-00a0c91e6bf6`

1653 In DCE and Microsoft Windows, the UUID is usually presented to the administrator in the form of a
1654 "friendly name". For instance the above UUID could represent the user john.hughes@entegrity.com.

8.4.3

1655 **8.4.4 SAML Attribute Naming**

1656 The NameFormat XML attribute in <AttributeDesignator> and <Attribute> elements MUST be
1657 <urn:oasis:names:tc:SAML:2.0:attrname-format:uri>.

1658 To construct attribute names, the URN `uuid` namespace described in [[http://www.ietf.org/internet-](http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-03.txt)
1659 [drafts/draft-mealling-uuid-urn-03.txt](http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-03.txt)] is used. In this approach the Name XML attribute is based on the
1660 URN form of the underlying UUID that identifies the attribute.

1661 **Example:**

1662 `urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6`

1663 For purposes of human readability, there may also be a requirement for some applications to carry an
1664 optional string name together with the OID URN. The optional XML attribute `FriendlyName` (defined in
1665 [[SAMLCore](#)]) MAY be used for this purpose.

1666 | **8.4.5 Profile-Specific XML Attributes**

1667 | No additional XML attributes are defined for use with the <AttributeDesignator> or <Attribute>
1668 | elements.

1669 | **8.4.6 <AttributeDesignator> Comparison**

1670 | Two <AttributeDesignator> elements are equal if and only if their Name XML attribute values are
1671 | equal in the sense of [http://www.ietf.org/internet-drafts/draft-mealling-uuid-urn-03.txt]. The
1672 | FriendlyName attribute plays no role in the comparison.

1673 | **8.4.7 SAML Attribute Values**

1674 | In cases in which the attribute's value is also a UUID, the same URN syntax described above MUST be
1675 | used to express the value within the <AttributeValue> element. The xsi:type XML attribute MUST
1676 | be set to xsd:anyURI.

1677 | If the attribute's value is not a UUID, then there are no restrictions on the use of the <AttributeValue>
1678 | element.

1679 | **8.4.8 Standard DCE Attribute Names**

1680 | DCE is able to transport a wide range of authorization data within its Privilege Attribute Certificate (PAC).
1681 | Several useful attributes carried within the PAC structure receive special mention. Each is named by a
1682 | well-defined UUID value, as described below:

1683 | **8.4.8.1**

1683 | ~~**The DCE PAC entry types that are supported in SAML 2.0 are provided in**~~
1684 | ~~**the below table. Principal**~~

1685 | This single-valued attribute represents the SAML subject's DCE principal identity, in UUID form.

1686 | **Name:** urn:uuid:TBD

1687 | **<AttributeValue>:** a UUID URN containing the UUID of the DCE principal

1688 | **8.4.8.2 Primary Group**

1689 | This single-valued attribute represents the SAML subject's primary DCE group membership, in UUID
1690 | form.

1691 | **Name:** urn:uuid:TBD

1692 | **<AttributeValue>:** a UUID URN containing the UUID of the DCE principal's primary DCE group

1693 | **8.4.8.3 Groups**

1694 | This multi-valued attribute represents the SAML subject's DCE local group memberships, in UUID form.

1695 | **Name:** urn:uuid:TBD

1696 | **<AttributeValue>:** a UUID URN containing the UUID of a group in which the DCE principal is a
1697 | member

8.4.9

1698 The attribute name space used for DCE attributes is:
1699 **URI:** <urn:oasis:names:tc:SAML:2.0:attribute-namespace#dce>

8.4.10 Attribute Values

1701 Whilst the UUID is guaranteed to be unique in across space and time, the friendly name is not. Hence
1702 each attribute defined in the previous sections will carry the UUID as well as a friendly name
1703 TBD—on how to define.

1704 Example

1705 TBD

8.5 XACML Attribute Profile

1707 SAML attribute assertions may be used as input to authorization decisions made according to the OASIS
1708 eXtensible Access Control Markup Language (XACML) standard specification [XACML]. Since the SAML
1709 attribute format differs from the XACML attribute format, there is a mapping that must be performed. The
1710 XACML attribute profile facilitates this mapping by standardizing naming, value syntax, and additional
1711 attribute metadata. SAML attributes generated in conformance with this profile can be mapped
1712 automatically into XACML attributes and used as input to XACML authorization decisions.

8.5.1 Required Information

1714 **Identification:** <urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML>

1715 **Contact information:** security-services-comment@lists.oasis-open.org

1716 **Description:** Given below.

1717 **Updates:** None.

8.5.2 SAML Attribute Naming

1719 The NameFormat XML attribute in <AttributeDesignator> and <Attribute> elements MUST be
1720 <urn:oasis:names:tc:SAML:2.0:attrname-format:uri>.

1721 The Name XML attribute MUST adhere to the rules specified for that format, as defined by [SAMLCore].

1722 For purposes of human readability, there may also be a requirement for some applications to carry an
1723 optional string name together with the OID URN. The optional XML attribute FriendlyName (defined in
1724 [SAMLCore]) MAY be used for this purpose, but is not translatable into the XACML attribute equivalent.

8.5.3 Profile-Specific XML Attributes

1726 XACML requires each attribute to carry an explicit data type. To supply this data type value, a new URI-
1727 valued XML attribute called DataType is defined in the XML namespace
1728 <urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML>.

1729 SAML <Attribute> elements conforming to this profile MUST include the namespace-qualified
1730 DataType attribute, or the value is presumed to be <http://www.w3.org/2001/XMLSchema#string>.

1731 While in principle any URI reference can be used as a data type, the standard values to be used are
1732 specified in Appendix A of the XACML 2.0 Specification [XACML]. If non-standard values are used, then
1733 each XACML PDP that will be consuming mapped SAML attributes with non-standard DataType values
1734 must be extended to support the new data types.

1735 | **8.5.4 <AttributeDesignator> Comparison**

1736 | Two <AttributeDesignator> elements are equal if and only if their Name XML attribute values are
1737 | equal in a binary comparison. The FriendlyName attribute plays no role in the comparison.

1738 | **8.5.5 SAML Attribute Values**

1739 | The syntax of the <AttributeValue> element's content MUST correspond to the data type expressed
1740 | in the profile-specific DataType XML attribute appearing in the parent <Attribute> element. For data
1741 | types corresponding to the types defined in section 3.3 of [XML-Schema-Part2], the xsi:type XML
1742 | attribute SHOULD also be used.

1743 | **8.5.6 Profile-Specific Schema**

1744 | The following schema defines the profile-specific DataType XML attribute:

```
1745 | <schema targetNamespace="urn:oasis:names:tc:SAML:2.0:profiles:attribute:XACML"  
1746 | xmlns="http://www.w3.org/2001/XMLSchema"  
1747 | version="2.0">  
1748 | <attribute name="DataType" type="anyURI"/>  
1749 | </schema>
```

1750 | **8.5.7 Example**

1751 | TBD

9 References

- 1753 **[AES]** FIPS-197, Advanced Encryption Standard (AES), available from <http://www.nist.gov/>.
- 1754 **[Anders]** A suggestion on how to implement SAML browser bindings without using “Artifacts”,
1755 <http://www.x-obi.com/OBI400/andersr-browser-artifact.ppt>.
- 1756 **[CoreAssnEx]** Core Assertions Architecture, Examples and Explanations, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-core-phill-07.pdf)
1757 [open.org/committees/security/docs/draft-sstc-core-phill-07.pdf](http://www.oasis-open.org/committees/security/docs/draft-sstc-core-phill-07.pdf).
- 1758 **[HTML401]** HTML 4.01 Specification, W3C Recommendation 24 December 1999,
1759 <http://www.w3.org/TR/html4>.
- 1760 **[Liberty]** The Liberty Alliance Project, <http://www.projectliberty.org>.
- 1761 **[MSURL]** Microsoft technical support article,
1762 <http://support.microsoft.com/support/kb/articles/Q208/4/27.ASP>.
- 1763 **[PAOS]** Aarts, R., “Liberty Reverse HTTP Binding for SOAP Specification”, Version: 1.0,
1764 <https://www.projectliberty.org/specs/liberty-paos-v1.0.pdf>
- 1765 **[Rescorla-Sec]** E. Rescorla et al., Guidelines for Writing RFC Text on Security Considerations,
1766 <http://www.ietf.org/internet-drafts/draft-iab-sec-cons-03.txt>.
- 1767 **[RFC1738]** Uniform Resource Locators (URL), <http://www.ietf.org/rfc/rfc1738.txt>
- 1768 **[RFC1750]** Randomness Recommendations for Security. <http://www.ietf.org/rfc/rfc1750.txt>
- 1769 **[RFC1945]** Hypertext Transfer Protocol -- HTTP/1.0, <http://www.ietf.org/rfc/rfc1945.txt>.
- 1770 **[RFC2045]** Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message
1771 Bodies, <http://www.ietf.org/rfc/rfc2045.txt>
- 1772 **[RFC2119]** S. Bradner, Key words for use in RFCs to Indicate Requirement Levels, IETF RFC 2119,
1773 March 1997, <http://www.ietf.org/rfc/rfc2119.txt>.
- 1774 **[RFC2246]** The TLS Protocol Version 1.0, <http://www.ietf.org/rfc/rfc2246.txt>.
- 1775 **[RFC2279]** UTF-8, a transformation format of ISO 10646, <http://www.ietf.org/rfc/rfc2279.txt>.
- 1776 **[RFC2616]** Hypertext Transfer Protocol -- HTTP/1.1, <http://www.ietf.org/rfc/rfc2616.txt>.
- 1777 **[RFC2617]** HTTP Authentication: Basic and Digest Access Authentication, IETF RFC 2617,
1778 <http://www.ietf.org/rfc/rfc2617.txt>.
- 1779 **[SAMLBind]** Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0, DRAFT
- 1780 **[SAMLCore]** E. Maler et al. Assertions and Protocol for the OASIS Security Assertion Markup
1781 Language (SAML). OASIS, September 2003. Document ID oasis-sstc-saml-core-1.1.
1782 <http://www.oasis-open.org/committees/security/>.
- 1783 **[SAMLMeta]** Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0, DRAFT
- 1784 **[SAMLGloss]** E. Maler et al. Glossary for the OASIS Security Assertion Markup Language (SAML).
1785 OASIS, September 2003. Document ID oasis-sstc-saml-glossary-1.1. [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1786 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 1787 **[SAMLSec]** E. Maler et al. Security Considerations for the OASIS Security Assertion Markup
1788 Language (SAML), OASIS, September 2003, Document ID oasis-sstc-saml-sec-consider-
1789 1.1. <http://www.oasis-open.org/committees/security/>.
- 1790 **[SAMLReqs]** Darren Platt et al., SAML Requirements and Use Cases, OASIS, April 2002,
1791 <http://www.oasis-open.org/committees/security/>.
- 1792 **[SAMLWeb]** OASIS Security Services Technical Committee website, [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1793 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 1794 **[SESSION]** RL “Bob” Morgan, Support of target web server sessions in Shibboleth,
1795 <http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-session-00.txt>
- 1796 **[ShibMarlena]** Marlena Erdos, Shibboleth Architecture DRAFT v1.1,

1797 <http://shibboleth.internet2.edu/draft-internet2-shibboleth-arch-v05.html> .

1798 [SOAP1.1] D. Box et al., Simple Object Access Protocol (SOAP) 1.1, World Wide Web Consortium
1799 Note, May 2000, <http://www.w3.org/TR/SOAP>.

1800 [SSL3] A. Frier et al., The SSL 3.0 Protocol, Netscape Communications Corp, November 1996.

1801 [SSTC-Liberty1.1]“Liberty v1.1 specification set submittal letter”, 11 Apr 2003, [http://lists.oasis-](http://lists.oasis-open.org/archives/security-services/200304/msg00072.html)
1802 [open.org/archives/security-services/200304/msg00072.html](http://lists.oasis-open.org/archives/security-services/200304/msg00072.html)

1803 [SSTC-Liberty1.2] “Liberty ID-FF 1.2 Contribution to SSTC”, 13 Nov 2003, [http://www.oasis-](http://www.oasis-open.org/archives/security-services/200311/msg00060.html)
1804 [open.org/archives/security-services/200311/msg00060.html](http://www.oasis-open.org/archives/security-services/200311/msg00060.html)

1805 [WEBSSO] RL “Bob” Morgan, Interactions between Shibboleth and local-site web sign-on services,
1806 <http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-websso-00.txt>

1807 [WSDL1_1] “Web Services Description Language (WSDL) 1.1”, W3C Note 15 March 2001,
1808 <http://www.w3.org/TR/2001/NOTE-wsdl-20010315>

1809 [WSS] A Nadalin, C Kaler, P Hallam-Baker, R Monzillo, “Web Services Security: SOAP Message
1810 Security .0 (WS-Security 2004)”, OASIS, March 2004, [http://www.oasis-](http://www.oasis-open.org/committees/wss)
1811 [open.org/committees/wss](http://www.oasis-open.org/committees/wss).

1812 [WSS-SAML] P. Hallam-Baker et al., Web Services Security: SAML Token Profile, OASIS, March 2003,
1813 <http://www.oasis-open.org/committees/wss>.

1814 [XMLSig] D. Eastlake et al., XML-Signature Syntax and Processing, World Wide Web Consortium,
1815 <http://www.w3.org/TR/xmlsig-core/>.

1816 [RFC2256] M. Wahl, RFC 2256 - A Summary of the X.500(96) User Schema for use with LDAPv3,
1817 December 1997

1818 [Morgan] R. L. Morgan, Conventions for Use of X.500/LDAP Attribute Types in SAML, Draft 00, 5
1819 November 2003, available from SAML Document repository as draft-morgan-saml-attr-
1820 x500-00.pdf

1821 [RFC2798] W. Smith, Definition of the inetOrgPerson LDAP Object class, April 2000

1822 [eduPersonSchema] eduPerson.ldif, Available from <http://www.educase.edu/eduperson>

1823 [Mealling] P Leach et al, A UUID URN Namespace. Internet-Draft, draft-mealling-uuid-urn-03.
1824 January 2004

1825 [RFC3061] M. Mealling, RFC3061 – A URN Namespace of Object Identifiers, Feb 2001

1826 [XML-Schema-Part2] Paul V. Biron, Ashok Malhotra, XML Schema Part 2: Datatypes, W3C
1827 Recommendation 02 May 2001, <http://www.w3.org/TR/xmlschema-2/>

1828 [XACML] [T. Moses, ed., OASIS eXtensible Access Control Markup Language \(XACML\) Versions](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml)
1829 [1.0, 1.1, and 2.0. Available on the OASIS XACML TC web page at http://www.oasis-](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml)
1830 [open.org/committees/tc_home.php?wg_abbrev=xacml.](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml)

1831 **A. Acknowledgments**

1832 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
1833 Committee, whose voting members at the time of publication were:

- 1834 • TBD

B. Revision History

Rev	Date	By Whom	What
00	02/16/04	Frederick Hirsch	Split new profiles document from bindings and profiles, removed bindings section. Added ECP profile, added and formatted references.
2	03/02/04	Frederick Hirsch	Removed URL Size restriction section – this is located in the bindings document. Minor cleanup in section 2.1
3	03/27/04	Frederick Hirsch	Changes to reflect core 8, review comments, corrections.
4	03/30/04	Frederick Hirsch	Additional review comments, corrections.
6	04/16/04	Scott Cantor	Replaced 1.1 SSO profiles with new proposal, added discovery profile, revised confirmation method descriptions, removed binding-related duplications, added placeholders for additional profiles.
7	05/09/04	Scott Cantor	Added NameIdentifierMapping profile
8	05/14/04	Frederick Hirsch	Changes based on 5/11/04 SSTC conference call – replace Identifier with ID in elements, in elements and attributes replace Authentication with Authn . Specifically, changed <AuthenticationStatement>, <NameIdentifierMappingRequest>, <NameIdentifierMappingResponse>, <EncryptedIdentifier>, <NameIdentifierMappingService>
9	05/30/04	Scott Cantor	Sync'd confirmation data sections to new schema in core-14, relaxed NameIDMapping profile requirement for SOAP binding, started clean-up of ECP, adjusted SSO profile to reflect bindings-12, added back sender-vouches.
10	06/07/04	Prateek Mishra	Added attribute profiles materials from hughes-mishra-baseline-attributes-04 with John Hughes updates
11	06/13/04	Scott Cantor	Added metadata considerations to profiles, minor editorial cleanups, new section headers for profiles
12	07/06/04	Scott Cantor	Final SSO cleanup, formalized ECP schema, fixed examples, intro section.
13	07/08/04	Scott Cantor	Added RelayState ECP header, more ECP cleanup, added SingleLogout profile, fixes to discovery profile
14	07/08/04	Scott Cantor	Filled in remaining profiles, re-edited attribute profiles

1836

C. Notices

1837 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
1838 might be claimed to pertain to the implementation or use of the technology described in this document or
1839 the extent to which any license under such rights might or might not be available; neither does it represent
1840 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
1841 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
1842 available for publication and any assurances of licenses to be made available, or the result of an attempt
1843 made to obtain a general license or permission for the use of such proprietary rights by implementors or
1844 users of this specification, can be obtained from the OASIS Executive Director.

1845 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
1846 other proprietary rights which may cover technology that may be required to implement this specification.
1847 Please address the information to the OASIS Executive Director.

1848 **Copyright © OASIS Open 2003-2004. All Rights Reserved.**

1849 This document and translations of it may be copied and furnished to others, and derivative works that
1850 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
1851 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
1852 this paragraph are included on all such copies and derivative works. However, this document itself may
1853 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
1854 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
1855 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
1856 into languages other than English.

1857 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
1858 or assigns.

1859 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1860 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
1861 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
1862 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

1863 JavaScript is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and
1864 other countries.