
1 Shibboleth Architecture

2 Technical Overview

3 Working Draft 02, 8 June 2005

4 Document identifier:

5 draft-mace-shibboleth-tech-overview-02

6 Location:

7 <http://shibboleth.internet2.edu/shibboleth-documents.html>

8 Editors:

9 Tom Scavo (trscavo@ncsa.uiuc.edu), NCSA

10 Scott Cantor (cantor.2@osu.edu), The Ohio State University

11 Contributors:

12 Nathan Dors (dors@cac.washington.edu), University of Washington

13 Abstract:

14 This non-normative document gives a technical overview of Shibboleth.

15 This is a **working draft** and the text may change before completion. Please submit
16 comments to the shibboleth-dev mailing list (see <http://shibboleth.internet2.edu/> for
17 subscription details).

18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58

Table of Contents

1	Introduction	3
1.1	Background	3
1.2	Notation	3
1.3	Outline	4
2	Shibboleth Components	5
2.1	Identity Provider	5
2.1.1	Authentication Authority	5
2.1.2	Single Sign-On Service	5
2.1.3	Inter-Site Transfer Service	5
2.1.4	Artifact Resolution Service	5
2.1.5	Attribute Authority	6
2.2	Service Provider	6
2.2.1	Assertion Consumer Service	6
2.2.2	Attribute Requester	6
2.3	WAYF Service	6
3	SAML Assertions	7
3.1	Authentication Assertions	7
3.2	Attribute Assertions	9
3.3	Multiple Assertions	9
3.4	Signed Assertions	11
4	Shibboleth SSO Profiles	13
4.1	Authentication Request Profile	13
4.2	Browser/POST Profile	13
4.2.1	Browser/POST Profile with WAYF Service	15
4.3	Browser/Artifact Profile	17
4.3.1	Browser/Artifact Profile with WAYF Service	20
5	Attributes	21
5.1	Browser/POST Profile with Attribute Exchange	21
5.2	Browser/Artifact Profile with Attribute Exchange	23
5.3	Directory Schema	24
6	Metadata	26
6.1	Identity Provider Metadata	26
6.1.1	SSO Service Metadata	27
6.1.2	Attribute Authority Metadata	28
6.2	Service Provider Metadata	29
6.2.1	Assertion Consumer Service Metadata	29
7	References	31
7.1	Normative References	31
7.2	Non-Normative References	31

59 1 Introduction

60 The *Shibboleth Architecture* [ShibProt] extends the SAML 1.1 single sign-on and attribute
61 exchange mechanisms by specifying service-provider-first SSO profiles and enhanced
62 features for user privacy. This document provides a technical overview of Shibboleth and as
63 such is an extension of [SAMLTech].

64 1.1 Background

65 The *Shibboleth Architecture* is built upon the following standards:

- 66 • Hypertext Transfer Protocol (HTTP)
- 67 • Extensible Markup Language (XML)
- 68 • XML Schema
- 69 • XML Signature
- 70 • SOAP¹
- 71 • Security Assertion Markup Language (SAML)

72 Definitive references for each of these standards are found in section 7 of this Overview.

73 1.2 Notation

74 Conventional XML namespace prefixes are used throughout the listings in this Overview,
75 whether or not a namespace declaration is present in the listing:

- 76 • The prefix `saml:` stands for the SAML 1.1 assertion namespace:
77 `urn:oasis:names:tc:SAML:1.0:assertion`
- 78 • The prefix `samlp:` stands for the SAML 1.1 request-response protocol namespace:
79 `urn:oasis:names:tc:SAML:1.0:protocol`
- 80 • The prefix `md:` stands for the SAML 2.0 metadata namespace:
81 `urn:oasis:names:tc:SAML:2.0:metadata`
- 82 • The prefix `ds:` stands for the W3C XML Signature namespace:
83 `http://www.w3.org/2000/09/xmldsig#`
- 84 • The prefix `xsd:` stands for the W3C XML Schema namespace:
85 `http://www.w3.org/2001/XMLSchema`
- 86 • The prefix `xsi:` stands for the W3C XML Schema namespace for schema-related
87 markup that appears in XML instances:
88 `http://www.w3.org/2001/XMLSchema-instance`

89 The following typographical conventions for *displayed text* are used:

90 HTTP requests appear like this.
91 HTTP responses appear like this.
92 HTML code listings appear like this.
93 JavaScript code listings appear like this.
94 XML code listings appear like this.
95 SAML code listings appear like this.

¹ Originally, SOAP was an acronym for "Simple Object Access Protocol", but this turned out to be a misnomer since SOAP has nothing to do with "object access" in the object-oriented programming sense of the phrase. So today we no longer think of "SOAP" as an acronym although the capitalization persists.

96 URNs appear like this.

97 URLs appear like this.

98 Likewise the following typographical conventions for *in-line text* are used: <XMLelement>,
99 <HTMLelement>, XMLattribute, HTMLattribute, LDAPattribute, HTTPparameter,
100 OtherCode. Finally, variable placeholders in code fragments (both displayed and in-line)
101 are italicized (e.g., *artifact*).

102 **1.3 Outline**

103 Immediately following this introduction, in section 2 we describe the components comprising
104 the *Shibboleth System*. Section 3 is a general introduction to *Security Assertion Markup*
105 *Language* (SAML) with special emphasis on those SAML constructs used by Shibboleth. In
106 section 4, the Shibboleth *browser profiles* are described in detail, while the Shibboleth
107 *attribute profiles* are outlined in section 5. Also in section 5, we emphasize the importance of
108 *directory schema* using *eduPerson* as the canonical example of a directory in higher
109 education. An introduction to SAML *metadata* is given in section 6, and finally in section 7
110 we provide a list of definitive *references* for further study.

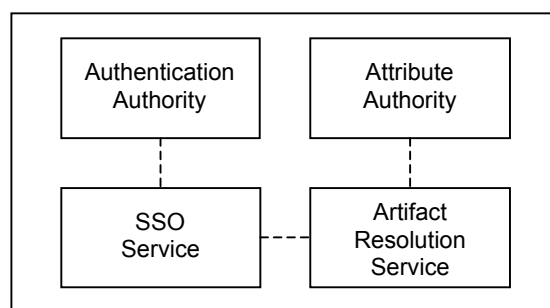
111 2 Shibboleth Components

112 The functional components of a conforming Shibboleth implementation include an identity
113 provider, a service provider, an optional “Where are you from?” (WAYF) service, and various
114 interacting subcomponents.

115 2.1 Identity Provider

116 An *identity provider* (formerly called an *origin*) maintains user credentials and attributes.
117 Upon request the identity provider (IdP) will assert authentication statements or attribute
118 statements to relying parties, specifically service providers. IdP subcomponents are depicted
119 in Figure 1 and discussed in the following subsections.

Identity Provider



120

121

Figure 1: Shibboleth Identity Provider

122 2.1.1 Authentication Authority

123 The *authentication authority* issues authentication statements to other components. The
124 authentication authority is integrated with the IdP's authentication service (the details of
125 which are out of scope).

126 2.1.2 Single Sign-On Service

127 A *single sign-on (SSO) service* is the first point of contact at the IdP. The SSO service
128 initiates the authentication process at the IdP and ultimately redirects the client to the inter-
129 site transfer service (unless the function of the SSO service and inter-site transfer service are
130 combined, which is encouraged). Note: The SSO service is not defined by SAML 1.1, which
131 specifies only identity-provider-first SSO profiles.

132 2.1.3 Inter-Site Transfer Service

133 The *inter-site transfer service* issues HTTP responses conforming to the Browser/POST and
134 Browser/Artifact profiles. The inter-site transfer service interacts with the authentication
135 authority behind the scenes to produce the required authentication assertion. Note: The
136 concept of an inter-site transfer service has been removed in SAML 2.0, so we assume that
137 the functions of the SSO service and the inter-site transfer service are combined, and
138 therefore ignore the latter in all that follows.

139 2.1.4 Artifact Resolution Service

140 If the Browser/Artifact profile is used, the IdP sends an artifact to the SP instead of the actual
141 assertion. (An *artifact* is a reference to an authentication assertion.) The SP then sends the

142 artifact to the *artifact resolution service* at the IdP via a back-channel exchange. In return, the
143 IdP sends the required authentication assertion to the SP.

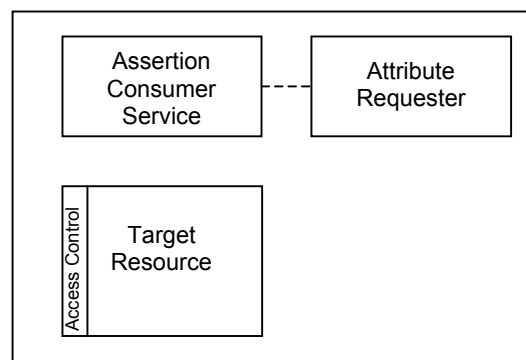
144 **2.1.5 Attribute Authority**

145 The *attribute authority* processes attribute requests; that is, it issues attribute assertions. The
146 attribute authority authenticates and authorizes any requests it receives.

147 **2.2 Service Provider**

148 A *service provider* (formerly called a *target*) manages secured resources. User access to
149 resources is based on assertions received by the service provider (SP) from an identity
150 provider. SP subcomponents are shown in Figure 2. Note that the service provider has
151 access control in place that prevents access by clients without a security context.

Service Provider



152

153

Figure 2: Shibboleth Service Provider

154 **2.2.1 Assertion Consumer Service**

155 The *assertion consumer service* (formerly called a *SHIRE*) is the service provider endpoint of
156 the SSO exchange. It processes the authentication assertion returned by the SSO service (or
157 artifact resolution service, depending on the profile used), initiates an optional attribute
158 request (see section 5), establishes a security context at the SP, and redirects the client to
159 the desired target resource.

160 **2.2.2 Attribute Requester**

161 An *attribute requester* (formerly called a *SHAR*) at the SP and the attribute authority at the
162 IdP may conduct a back-channel attribute exchange once a security context has been
163 established at the SP. That is, the SP and IdP interact directly, bypassing the browser.

164 **2.3 WAYF Service**

165 An optional *WAYF service* is operated independent of the SP and IdP. The WAYF can be
166 used by the SP to determine the user's preferred IdP, with or without user interaction. The
167 WAYF is essentially a proxy for the authentication request passed from the SP to the SSO
168 service at the IdP.

169 3 SAML Assertions

170 This section provides some background material on SAML assertions that sets the stage for
171 the profiles to follow. In Shibboleth, *SAML assertions* are transferred from identity providers
172 to service providers. Assertions contain statements that service providers can use to make
173 access control decisions. Three types of statements are specified by SAML:

- 174 1. Authentication statements
- 175 2. Attribute statements
- 176 3. Authorization decision statements

177 *Authentication statements* assert to the service provider that the principal did indeed
178 authenticate with the identity provider at a particular time using a particular method of
179 authentication. The examples in section 3.1, for instance, are assertions containing
180 authentication statements typically used in the Browser/POST and Browser/Artifact profiles
181 (section 4), respectively.

182 *Attribute statements* provide additional information about a principal so that service providers
183 can make access control decisions. For example, the attribute assertion given in section 3.2
184 is the result of the attribute exchange described in section 5.1.

185 Authentication statements and attribute statements may be combined in a single assertion.
186 Alternatively, multiple assertions may be issued by the IdP, each containing its own
187 statement. A variation of the Browser/Artifact profile (section 5.2), for instance, relies on a
188 pair of assertions, both obtained as the result of a single message exchange. The example
189 in section 3.3 illustrates the corresponding pair of assertions.

190 In some situations, it may be preferable for the application to delegate an access control
191 decision to another component or service. In this case, the service provider indicates to the
192 service the resource being accessed where after the service asserts an *authorization*
193 *decision statement* that dictates whether or not the principal is allowed access to the
194 resource. In Shibboleth, such statements are out of scope, but the interested reader is
195 referred to [GSISecurity] for a relevant example.

196 Finally, SAML assertions may be digitally signed. An XML element such as that given in
197 section 3.4 is inserted into those assertions where additional message integrity is required.

198 3.1 Authentication Assertions

199 The following *authentication assertion* (sometimes called an *SSO assertion*) contains a
200 `<saml:AuthenticationStatement>` element (but no attributes):

```
201 <saml:Assertion  
202   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"  
203   MajorVersion="1" MinorVersion="1"  
204   AssertionID="a75adf55-01d7-40cc-929f-dbd8372ebdfc"  
205   IssueInstant="2004-12-05T09:22:02Z"  
206   Issuer="https://idp.example.org/shibboleth">  
207   <saml:Conditions  
208     NotBefore="2004-12-05T09:17:02Z"  
209     NotOnOrAfter="2004-12-05T09:27:02Z">  
210     <saml:AudienceRestrictionCondition>  
211       <saml:Audience>http://sp.example.org/shibboleth</saml:Audience>  
212     </saml:AudienceRestrictionCondition>  
213   </saml:Conditions>  
214   <saml:AuthenticationStatement  
215     AuthenticationInstant="2004-12-05T09:22:00Z"
```

```

216 AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
217 <saml:Subject>
218   <saml:NameIdentifier
219     Format="urn:mace:shibboleth:1.0:nameIdentifier"
220     NameQualifier="https://idp.example.org/shibboleth">
221     3f7b3dcf-1674-4ecd-92c8-1544f346baf8
222   </saml:NameIdentifier>
223   <saml:SubjectConfirmation>
224     <saml:ConfirmationMethod>
225       urn:oasis:names:tc:SAML:1.0:cm:bearer
226     </saml:ConfirmationMethod>
227   </saml:SubjectConfirmation>
228 </saml:Subject>
229 </saml:AuthenticationStatement>
230 </saml:Assertion>

```

231 The value of the `<saml:NameIdentifier>` element in the preceding example is called a
232 Shibboleth *handle*: an opaque, transient identifier associated with the authenticated user.
233 Further messages regarding this user (such as the one in the next section) explicitly refer to
234 this handle instead of the actual identity of the user.

235 Note that the above assertion (called a *bearer assertion*) assigns the following value to the
236 `<saml:ConfirmationMethod>` element:

```
237 urn:oasis:names:tc:SAML:1.0:cm:bearer
```

238 Whereas the preceding assertion is used in the Browser/POST profile (section 4.2), the
239 following assertion is used in the Browser/Artifact profile (section 4.3):

```

240 <saml:Assertion
241   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
242   MajorVersion="1" MinorVersion="1"
243   AssertionID="003c6cc1-9ff8-10f9-990f-004005b13a2b"
244   IssueInstant="2004-12-05T09:22:05Z"
245   Issuer="https://idp.example.org/shibboleth">
246   <saml:Conditions
247     NotBefore="2004-12-05T09:17:05Z"
248     NotOnOrAfter="2004-12-05T09:27:05Z">
249     <saml:AudienceRestrictionCondition>
250       <saml:Audience>http://sp.example.org/shibboleth</saml:Audience>
251     </saml:AudienceRestrictionCondition>
252   </saml:Conditions>
253   <saml:AuthenticationStatement
254     AuthenticationInstant="2004-12-05T09:22:00Z"
255     AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
256     <saml:Subject>
257       <saml:NameIdentifier
258         Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress"
259         NameQualifier="https://idp.example.org/shibboleth">
260         user@idp.example.org
261       </saml:NameIdentifier>
262       <saml:SubjectConfirmation>
263         <saml:ConfirmationMethod>
264           urn:oasis:names:tc:SAML:1.0:cm:artifact
265         </saml:ConfirmationMethod>
266       </saml:SubjectConfirmation>
267     </saml:Subject>
268   </saml:AuthenticationStatement>
269 </saml:Assertion>

```


270 In this case, the value of the `<saml:ConfirmationMethod>` element is
271 `urn:oasis:names:tc:SAML:1.0:cm:artifact`
272 which indicates the Browser/Artifact profile was used to retrieve the assertion.
273 Some use cases require more than an opaque identifier, so for the sake of illustration the
274 value of the `<saml:NameIdentifier>` element in the preceding example is an email
275 address. Presumably this uniquely identifies the authenticated user to the SP.
276 Note that an authentication assertion may include additional information about the subject,
277 such as attributes.

278 **3.2 Attribute Assertions**

279 The following *attribute assertion* contains a `<saml:AttributeStatement>` element only:

```
280 <saml:Assertion  
281   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"  
282   MajorVersion="1" MinorVersion="1"  
283   AssertionID="a144e8f3-adad-594a-9649-924517abe933"  
284   IssueInstant="2004-12-05T09:22:05Z"  
285   Issuer="https://idp.example.org/shibboleth">  
286   <saml:Conditions  
287     NotBefore="2004-12-05T09:17:05Z"  
288     NotOnOrAfter="2004-12-05T09:52:05Z">  
289     <saml:AudienceRestrictionCondition>  
290       <saml:Audience>http://sp.example.org/shibboleth</saml:Audience>  
291     </saml:AudienceRestrictionCondition>  
292   </saml:Conditions>  
293   <saml:AttributeStatement>  
294     <saml:Subject>  
295       <saml:NameIdentifier  
296         Format="urn:mace:shibboleth:1.0:nameIdentifier"  
297         NameQualifier="https://idp.example.org/shibboleth">  
298         3f7b3dcf-1674-4ecd-92c8-1544f346baf8  
299       </saml:NameIdentifier>  
300     </saml:Subject>  
301     <saml:Attribute  
302       AttributeName="urn:mace:dir:attribute-def:eduPersonPrincipalName"  
303       AttributeNamespace="urn:mace:shibboleth:1.0:attributeNamespace:uri">  
304       <saml:AttributeValue Scope="example.org">  
305         userid  
306       </saml:AttributeValue>  
307     </saml:Attribute>  
308   </saml:AttributeStatement>  
309 </saml:Assertion>
```

310 Note that the value of the `<saml:NameIdentifier>` element is the same as that in the first
311 example in section 3.1, which implies that the attribute statement is obtained subsequent to
312 and as a consequence of the previous authentication statement.

313 **3.3 Multiple Assertions**

314 The following pair of assertions contain `<saml:AuthenticationStatement>` and
315 `<saml:AttributeStatement>` elements, respectively:

```
316 <saml:Assertion  
317   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"  
318   MajorVersion="1" MinorVersion="1"
```

```

319 AssertionID="00305ed1-a1bd-10f9-a2d0-004005b13a2b"
320 IssueInstant="2004-12-05T09:22:05Z"
321 Issuer="https://idp.example.org/shibboleth">
322 <saml:Conditions
323   NotBefore="2004-12-05T09:17:05Z"
324   NotOnOrAfter="2004-12-05T09:27:05Z">
325   <saml:AudienceRestrictionCondition>
326     <saml:Audience>http://sp.example.org/shibboleth</saml:Audience>
327   </saml:AudienceRestrictionCondition>
328 </saml:Conditions>
329 <saml:AuthenticationStatement
330   AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
331   AuthenticationInstant="2004-12-05T09:22:00Z">
332   <saml:Subject>
333     <saml:NameIdentifier
334       Format="urn:mace:shibboleth:1.0:nameIdentifier"
335       NameQualifier="https://idp.example.org/shibboleth"
336       082dd87d-f380-4fd6-8726-694ef2bb71e9
337     </saml:NameIdentifier>
338     <saml:SubjectConfirmation>
339       <saml:ConfirmationMethod>
340         urn:oasis:names:tc:SAML:1.0:cm:artifact
341       </saml:ConfirmationMethod>
342     </saml:SubjectConfirmation>
343   </saml:Subject>
344 </saml:AuthenticationStatement>
345 </saml:Assertion>
346 <saml:Assertion
347   xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
348   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
349   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
350   MajorVersion="1" MinorVersion="1"
351   AssertionID="008c6181-a288-10f9-b6d6-004005b13a2b"
352   IssueInstant="2004-12-05T09:22:05Z"
353   Issuer="https://idp.example.org/shibboleth">
354   <saml:Conditions
355     NotBefore="2004-12-05T09:17:05Z"
356     NotOnOrAfter="2004-12-05T09:52:05Z">
357     <saml:AudienceRestrictionCondition>
358       <saml:Audience>http://sp.example.org/shibboleth</saml:Audience>
359     </saml:AudienceRestrictionCondition>
360   </saml:Conditions>
361   <saml:AttributeStatement>
362     <saml:Subject>
363       <saml:NameIdentifier
364         Format="urn:mace:shibboleth:1.0:nameIdentifier"
365         NameQualifier="https://idp.example.org/shibboleth"
366         082dd87d-f380-4fd6-8726-694ef2bb71e9
367       </saml:NameIdentifier>
368     </saml:Subject>
369     <saml:Attribute
370       AttributeName="urn:mace:dir:attribute-def:eduPersonScopedAffiliation"
371       AttributeNamespace="urn:mace:shibboleth:1.0:attributeNamespace:uri">
372       <saml:AttributeValue Scope="example.org">
373         member
374       </saml:AttributeValue>
375     <saml:AttributeValue Scope="example.org">

```

```

376     faculty
377     </saml:AttributeValue>
378 </saml:Attribute>
379 <saml:Attribute
380     AttributeName="urn:mace:dir:attribute-def:eduPersonEntitlement"
381     AttributeNamespace="urn:mace:shibboleth:1.0:attributeNamespace:uri">
382     <saml:AttributeValue xsi:type="xsd:anyURI">
383         urn:mace:oclc.org:100277910
384     </saml:AttributeValue>
385     <saml:AttributeValue xsi:type="xsd:anyURI">
386         https://sp.example.org/entitlements/123456789
387     </saml:AttributeValue>
388     <saml:AttributeValue xsi:type="xsd:anyURI">
389         urn:mace:incommon:entitlement:common:1
390     </saml:AttributeValue>
391 </saml:Attribute>
392 </saml:AttributeStatement>
393 </saml:Assertion>

```

394 **3.4 Signed Assertions**

395 Many SAML assertions, especially assertions that pass through a browser, are digitally
396 signed (see section 5 of [SAMLCore]). Here we give an example of a `<ds:Signature>`
397 element (with visual formatting, which invalidates the signature) used to sign the
398 authentication assertion obtained via the Browser/POST profile in section 4.2:

```

399 <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
400     <ds:SignedInfo>
401         <ds:CanonicalizationMethod
402             Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
403         <ds:SignatureMethod
404             Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
405         <ds:Reference URI="#c7055387-af61-4fce-8b98-e2927324b306">
406             <ds:Transforms>
407                 <ds:Transform
408                     Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
409 signature" />
410                 <ds:Transform
411                     Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
412                     <InclusiveNamespaces
413                         PrefixList="#default saml samlp ds xsd xsi"
414                         xmlns="http://www.w3.org/2001/10/xml-exc-c14n#" />
415                     </ds:Transform>
416                 </ds:Transforms>
417                 <ds:DigestMethod
418                     Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
419                 <ds:DigestValue>TCDVSuG6grhyHbzhQFWFzGrxIPE=</ds:DigestValue>
420             </ds:Reference>
421         </ds:SignedInfo>
422         <ds:SignatureValue>
423             x/GyPbzmFEe85pGD3c1aXG4VspB9V9jGCjwcRCKrtwPS6vdVNCcY5rHaFPYWkf+5
424             EIYcPzx+pX1h43SmwviCqXRjRtMANWbHLhWAptaK1ywS7gFgsD01qjyen3CP+m3D
425             w6vKhaqledl0BYyrIzb4KkHO4ahNyBVXbJwqv5pUaE4=
426         </ds:SignatureValue>
427         <ds:KeyInfo>
428             <ds:X509Data>
429                 <ds:X509Certificate>

```

4 Shibboleth SSO Profiles

Shibboleth specifies two browser-based single sign-on (SSO) profiles:

1. Browser/POST Profile
2. Browser/Artifact Profile

These profiles extend the SAML 1.1 profiles of the same name [SAMLBind]. While the SAML 1.1 profiles begin with a request to the IdP, the Shibboleth profiles are service-provider-first and therefore more complex.

4.1 Authentication Request Profile

To accommodate SP-first profiles, Shibboleth introduces the notion of an *authentication request*. A Shibboleth authentication request is a URL-encoded message sent to an SSO service endpoint at the IdP. The following parameters are included in the query string:

- `providerId`

The `providerId` parameter is the unique identifier (usually a URI) of the SP. The IdP may use the `providerId` to perform special handling or processing of the authentication request.

- `shire`

The `shire` parameter (so named for historical reasons) is the location of the assertion consumer service endpoint at the SP. When using the Browser/POST profile, this URL becomes the value of the `<form>` element's `action` attribute. For the Browser/Artifact profile, the value of the `shire` parameter is a prefix of the redirect URL used by the SSO service.

- `target`

The `target` parameter is a state maintenance parameter, possibly the location of the target resource. Regardless of the profile used, the `target` parameter must be preserved by the IdP and included in the response to the SP.

- `time`

The (optional) `time` parameter is the current time in seconds past the epoch. It is used to assist the IdP in detecting stale requests from the client. It is important that the `time` parameter *not* be used as any kind of security measure.

Here is an example of a Shibboleth authentication request (without URL encoding for clarity):

```
https://idp.example.org/shibboleth/SSO?  
target=https://sp.example.org/myresource&  
shire=https://sp.example.org/shibboleth/SSO&  
providerId=https://sp.example.org/shibboleth&  
time=1102260120
```

Detailed handling of this request is outlined in the subsections below.

4.2 Browser/POST Profile

The *Shibboleth Browser/POST profile* is a combination of two profiles, the Shibboleth Authentication Request profile [ShibProt] and the SAML 1.1 Browser/POST profile [SAMLBind]. The message flow associated with the Shibboleth Browser/POST profile is depicted in Figure 3. As with both Shibboleth SSO profiles, the message flow begins with a request for a secured resource at the SP.

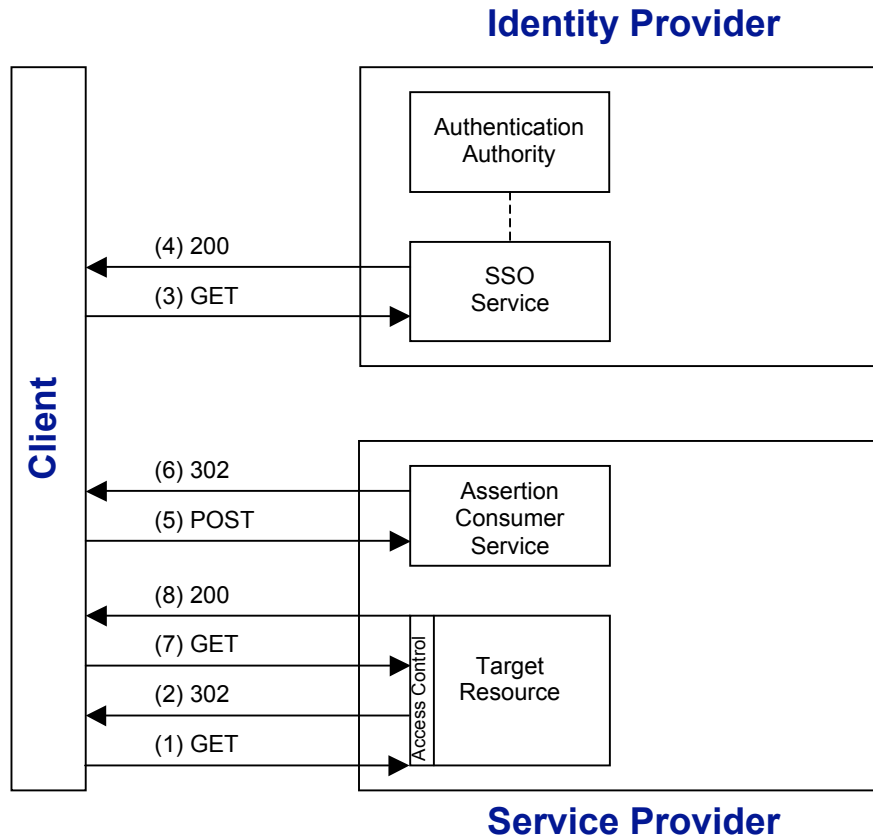


Figure 3: Shibboleth Browser/POST Profile

493

494

495 (1) The client requests a target resource at the SP:

496 `https://sp.example.org/myresource`

497 The SP performs a security check on behalf of the target resource. If a valid security context
498 at the SP already exists, skip steps 2–7.

499 (2) The SP redirects the client to the single sign-on (SSO) service at the IdP. Three
500 parameters are appended to the redirect URL (the optional `time` parameter is omitted in this
501 example).

502 (3) The client requests the SSO service at the IdP:

503 `https://idp.example.org/shibboleth/SSO?`
504 `target=https://sp.example.org/myresource&`
505 `shire=https://sp.example.org/shibboleth/SSO/POST&`
506 `providerId=https://sp.example.org/shibboleth`

507 The SSO service processes the authentication request and performs a security check. If the
508 user does not have a valid security context, the IdP identifies the principal (details omitted).
509 Once the principal has been identified, the SSO service obtains an authentication statement
510 from the authentication authority.

511 (4) The SSO service responds with a document containing an HTML form:

```
512 <form method="post"
513       action="https://sp.example.org/shibboleth/SSO/POST" ...>
514   <input name="TARGET" type="hidden"
515         value="https://sp.example.org/myresource" />
516   <input name="SAMLResponse" value="response" type="hidden" />
```

```
517 ...
518 <input type="submit" value="Submit" />
519 </form>
```

520 The value of the `TARGET` parameter has been preserved from step 3. The value of the
521 `SAMLResponse` parameter is the base64 encoding of a digitally signed
522 `<samlp:Response>` element such as the one below:

```
523 <samlp:Response
524   xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
525   MajorVersion="1" MinorVersion="1"
526   IssueInstant="2004-12-05T09:22:02Z"
527   Recipient="https://sp.example.org/shibboleth/SSO/POST"
528   ResponseID="c7055387-af61-4fce-8b98-e2927324b306">
529   <!-- insert ds:Signature element here (section 3.4) -->
530   <samlp:Status><samlp:StatusCode Value="samlp:Success"/></samlp:Status>
531   <!-- insert SAML assertion here (section 3.1) -->
532 </samlp:Response>
```

533 (5) The client issues a POST request to the assertion consumer service at the SP. To
534 automate the submission of the form, the following line of JavaScript (or its equivalent) may
535 appear in the HTML document containing the form in step 4:

```
536 window.onload = function() { document.forms[0].submit(); }
```

537 This assumes that the page contains a single HTML `<form>` element.

538 (6) The assertion consumer service processes the authentication response, creates a
539 security context at the SP and redirects the client to the target resource.

540 (7) The client requests the target resource at the SP (again):

```
541 https://sp.example.org/myresource
```

542 (8) Since a security context exists, the SP returns the resource to the client.

543 **4.2.1 Browser/POST Profile with WAYF Service**

544 In general, the SP does not know the user's preferred IdP at step 2 of the Browser/POST
545 profile. The process whereby the SP determines the appropriate IdP is called *identity*
546 *provider discovery* (generally, a very difficult problem). Shibboleth specifies an (optional)
547 WAYF service to aid the SP in IdP discovery.

548 A WAYF ("Where are you from?") service is an intermediary between the SP and the IdP.
549 The SP sends its authentication request to the WAYF, which somehow determines the user's
550 preferred IdP (through unspecified means) and then subsequently redirects the client to the
551 desired IdP. In the process, the WAYF preserves the values of all parameters except the
552 `time` parameter, which is updated.

553 In practice, the first time the client accesses an SP, the WAYF might present the user with a
554 form containing a list of all available IdPs. The user selects an IdP from the list and submits
555 the form, after which the WAYF redirects the client to the selected IdP. At the same time, the
556 client stores a reference to the IdP in a client-side cookie such that subsequent visits to the
557 same SP are redirected straight through to the corresponding IdP by the WAYF.

558 As a hypothetical example, suppose that a WAYF service resides at domain `example.org`
559 and that the SP redirects the client to this WAYF as follows:

```
560 https://wayf.example.org/?
561   target=https://sp.example.org/myresource&
562   shire=https://sp.example.org/shibboleth/SSO/POST&
563   providerId=https://sp.example.org/shibboleth
```

564 In response, the WAYF returns a form to the client with the following elements:

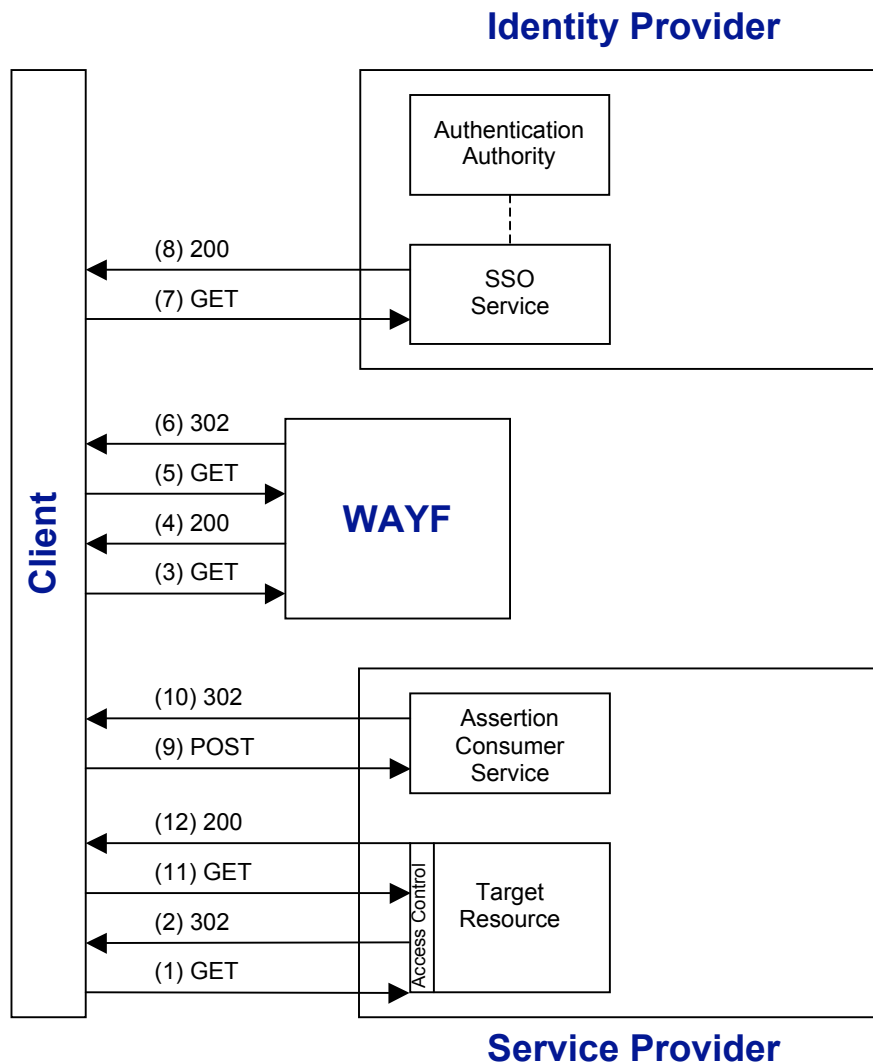
```

565 <form method="get" action="https://wayf.example.org/" ...>
566   <input name="target" type="hidden"
567     value="https://sp.example.org/myresource" />
568   <input name="shire" type="hidden"
569     value="https://sp.example.org/shibboleth/SSO/POST" />
570   <input name="providerId" type="hidden"
571     value="https://sp.example.org/shibboleth" />
572   ...
573   <input type="submit" value="Submit" />
574 </form>

```

575 Presumably the form contains additional controls that allow the user to select an IdP, after
576 which the form is submitted to the WAYF. The WAYF processes the form, sets the cookie
577 and redirects the client to the selected IdP.

578 The message flow of the Shibboleth Browser/POST profile with WAYF service is outlined in
579 Figure 4. (The flow of the corresponding Browser/Artifact profile is similar.) In step 4 of that
580 flow, the WAYF returns an HTML form to the user who chooses an IdP from a list. After the
581 form is submitted at step 5, the client is redirected to the SSO service of the chosen IdP
582 where after the flow is identical to the ordinary Browser/POST profile.



583

584

Figure 4: Shibboleth Browser/POST Profile with WAYF Service

585 (1) The client requests a target resource at the SP:

```
586 https://sp.example.org/myresource
```

587 The SP performs a security check on behalf of the target resource. If a valid security context
588 at the SP already exists, skip steps 2–11.

589 (2) The SP redirects the client to the WAYF. Three parameters are appended to the redirect
590 URL (the optional `time` parameter is omitted in this example).

591 (3) The client requests the WAYF service:

```
592 https://wayf.example.org/?  
593   target=https://sp.example.org/myresource&  
594   shire=https://sp.example.org/shibboleth/SSO/POST&  
595   providerId=https://sp.example.org/shibboleth
```

596 The WAYF service processes the authentication request and performs a cookie check. If the
597 user has the required cookie, skip steps 4 and 5. Otherwise an HTML form is returned to the
598 client.

599 (4) The WAYF returns an HTML form to the client. The parameters of the authentication
600 request (shown in the previous step) are encoded as hidden fields in the form as illustrated
601 above.

602 (5) The user selects an IdP from the list and submits the form, which causes the browser to
603 issue a GET request to the WAYF. (Depending on the particular WAYF implementation,
604 multiple HTTP transactions may occur at this step.)

605 (6) The WAYF updates the cookie with the user's preferred IdP and redirects the client to the
606 SSO service. Three parameters are appended to the redirect URL.

607 (7) The client requests the SSO service at the IdP:

```
608 https://idp.example.org/shibboleth/SSO?  
609   target=https://sp.example.org/myresource&  
610   shire=https://sp.example.org/shibboleth/SSO/POST&  
611   providerId=https://sp.example.org/shibboleth
```

612 The SSO service processes the authentication request and performs a security check. If the
613 user does not have a valid security context, the IdP identifies the principal (details omitted).
614 Once the principal has been identified, the SSO service obtains an authentication statement
615 from the authentication authority.

616 (8) The SSO service responds with an HTML form as in step 4 of the ordinary Browser/POST
617 profile.

618 (9) The client issues a POST request to the assertion consumer service at the service
619 provider as in step 5 of the ordinary Browser/POST profile.

620 (10) The assertion consumer service processes the authentication response, creates a
621 security context at the SP and redirects the client to the target resource.

622 (11) The client requests the target resource at the SP (again):

```
623 https://sp.example.org/myresource
```

624 (12) Since a security context exists, the SP returns the resource to the client.

625 **4.3 Browser/Artifact Profile**

626 The *Shibboleth Browser/Artifact profile* is a combination of the Shibboleth Authentication
627 Request [ShibProt] profile and the SAML 1.1 Browser/Artifact profile [SAMLBind]. The
628 message flow associated with the Shibboleth Browser/Artifact profile is shown in Figure 5.

629 Note the back-channel exchange of the SAML artifact in steps 6 and 7. This is the
 630 distinguishing characteristic between the Browser/Artifact and Browser/POST profiles.

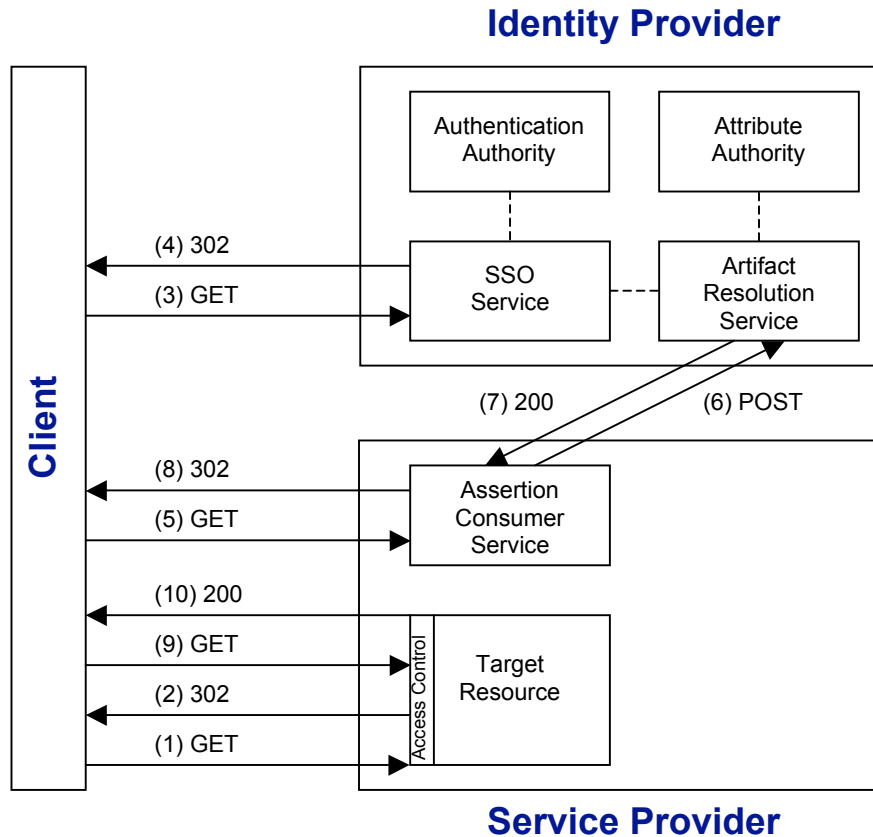


Figure 5: Shibboleth Browser/Artifact Profile

631
 632
 633 (1) The client requests a target resource at the SP:

```
634 https://sp.example.org/myresource
```

635 The SP performs a security check on behalf of the target resource. If a valid security context
 636 at the SP already exists, skip steps 2–9.

637 (2) The SP redirects the client to the single sign-on (SSO) service at the IdP. Three
 638 parameters are appended to the redirect URL (the optional `time` parameter is omitted in this
 639 example).

640 (3) The client requests the SSO service at the IdP:

```
641 https://idp.example.org/shibboleth/SSO?  

  642   target=https://sp.example.org/myresource&  

  643   shire=https://sp.example.org/shibboleth/SSO/Artifact&  

  644   providerId=https://sp.example.org/shibboleth
```

645 The SSO service processes the authentication request and performs a security check. If the
 646 user does not have a valid security context, the IdP identifies the principal (details omitted).
 647 Once the principal has been identified, the SSO service obtains an authentication statement
 648 from the authentication authority.

649 (4) The SSO service redirects the client to the assertion consumer service at the SP. A
 650 `TARGET` parameter and a `SAMLart` parameter are appended to the redirect URL prefix given
 651 by the `shire` parameter in step 3.

652 (5) The client requests the assertion consumer service at the SP:

```
653 https://sp.example.org/shibboleth/SSO/Artifact?
654 TARGET=https://sp.example.org/myresource&
655 SAMLart=AAEwGDwd3Z7Fr1GPbM82Fk2CZbpNBldxD%2Bt2Prp%2BTDtqxVA78iMf3F23
```

656 where the value of the TARGET parameter is preserved from step 3 and the value of the
657 SAMLart parameter is a SAML artifact of type 0x0001 defined in [SAMLBind].

658 (6) The assertion consumer service validates the request and dereferences the artifact by
659 sending a SAML Request to the artifact resolution service at the IdP:

```
660 POST /shibboleth/Artifact HTTP/1.1
661 Host: idp.example.org
662 Content-Type: text/xml
663 Content-Length: nnn
664 SOAPAction: http://www.oasis-open.org/committees/security
665
666 <?xml version="1.1" encoding="ISO-8859-1"?>
667 <SOAP-ENV:Envelope
668   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
669   <SOAP-ENV:Header/>
670   <SOAP-ENV:Body>
671     <samlp:Request
672       xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
673       MajorVersion="1" MinorVersion="1"
674       RequestID="f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
675       IssueInstant="2004-12-05T09:22:04Z">
676       <samlp:AssertionArtifact>
677         AAEwGDwd3Z7Fr1GPbM82Fk2CZbpNBldxD+t2Prp+TDtqxVA78iMf3F23
678       </samlp:AssertionArtifact>
679     </samlp:Request>
680   </SOAP-ENV:Body>
681 </SOAP-ENV:Envelope>
```

682 where the value of the <samlp:AssertionArtifact> element is the SAML artifact at
683 step 5.

684 (7) The artifact resolution service at the IdP returns a <samlp:Response> element
685 (containing an authentication statement) to the assertion consumer service at the SP:

```
686 HTTP/1.1 200 OK
687 Content-Type: text/xml
688 Content-Length: nnnn
689
690 <?xml version="1.1" encoding="ISO-8859-1"?>
691 <SOAP-ENV:Envelope
692   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
693   <SOAP-ENV:Header/>
694   <SOAP-ENV:Body>
695     <samlp:Response
696       xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
697       MajorVersion="1" MinorVersion="1"
698       Recipient="https://sp.example.org/shibboleth/SSO/Artifact"
699       ResponseID="00099cf1-a355-10f9-9e95-004005b13a2b"
700       InResponseTo="f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
701       IssueInstant="2004-12-05T09:22:05Z">
702     <samlp:Status>
703       <samlp:StatusCode Value="samlp:Success"/>
704     </samlp:Status>
```

```
705     <!-- insert SAML assertion here (see section 3.1) -->
706     </samlp:Response>
707 </SOAP-ENV:Body>
708 </SOAP-ENV:Envelope>
```

709 (8) The assertion consumer service processes the authentication response, creates a
710 security context at the SP and redirects the client to the target resource.

711 (9) The client requests the target resource at the SP (again):

```
712 https://sp.example.org/myresource
```

713 (10) Since a security context exists, the SP returns the resource to the client.

714 **4.3.1 Browser/Artifact Profile with WAYF Service**

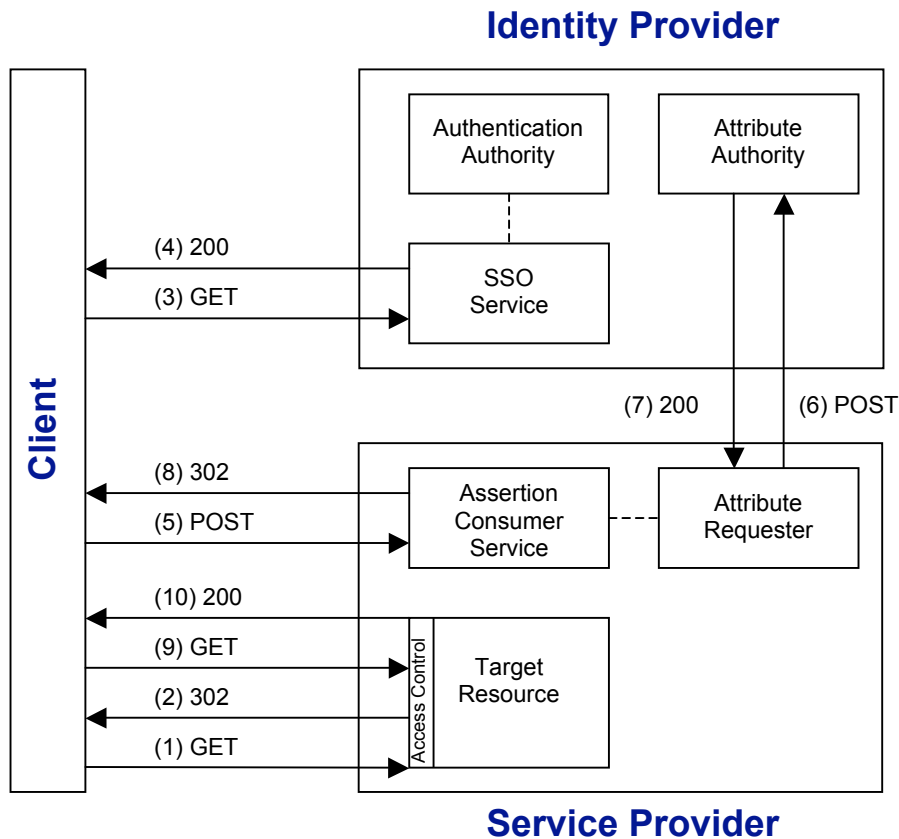
715 Like the Browser/POST profile, an SP that supports the Browser/Artifact profile may also
716 utilize the services of a WAYF. The resulting Browser/Artifact profile with WAYF service is
717 analogous to the Browser/POST case (section 4.2.1).

718 **5 Attributes**

719 Shibboleth specifies the use of a standard SAML *attribute request protocol* to facilitate
720 attribute sharing among IdPs and SPs. In this section we augment the SSO profiles of
721 sections 4.2 and 4.3 with an attribute exchange that permits the SP to make an informed
722 access control decision. Such an attribute exchange is optional since the SP may choose to
723 provide a security context based solely on an authentication assertion. In practice, however,
724 the SP may need to know more about the authenticated user before access to a resource
725 may be granted.

726 **5.1 Browser/POST Profile with Attribute Exchange**

727 Although attributes may be embedded in the authentication assertion transferred from IdP to
728 SP, Shibboleth specifies an optional back-channel exchange using a SAML protocol binding
729 such as the SOAP 1.1 binding. The attribute exchange is a simple two-step process initiated
730 towards the end of the ordinary Browser/POST profile (see Figure 6).



731

732

Figure 6: Shibboleth Browser/POST Profile with Attribute Exchange

733 The first five steps of the ordinary Browser/POST profile proceed as before. Before access
734 to the resource is granted, however, an attribute exchange is required.

735 (1–5) Same as the ordinary Browser/POST profile.

736 (6) The assertion consumer service parses the POST request, validates the signature on the
737 <samlp:Response> element, creates a security context at the SP and passes control to the

738 attribute requester to initiate the attribute exchange. The attribute requester POSTs a SAML
739 SOAP message to the attribute authority (AA) at the IdP:

```
740 POST /shibboleth/AA/SOAP HTTP/1.1
741 Host: idp.example.org
742 Content-Type: text/xml
743 Content-Length: nnn
744 SOAPAction: http://www.oasis-open.org/committees/security
745
746 <?xml version="1.1" encoding="ISO-8859-1"?>
747 <SOAP-ENV:Envelope
748   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
749   <SOAP-ENV:Header/>
750   <SOAP-ENV:Body>
751     <samlp:Request
752       xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
753       xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
754       MajorVersion="1" MinorVersion="1"
755       IssueInstant="2004-12-05T09:22:04Z"
756       RequestID="aaf23196-1773-2113-474a-fe114412ab72">
757     <samlp:AttributeQuery
758       Resource="https://sp.example.org/shibboleth">
759       <saml:Subject>
760         <saml:NameIdentifier
761           Format="urn:mace:shibboleth:1.0:nameIdentifier"
762           NameQualifier="https://idp.example.org/shibboleth">
763           3f7b3dcf-1674-4ecd-92c8-1544f346baf8
764         </saml:NameIdentifier>
765       </saml:Subject>
766       <saml:AttributeDesignator
767         AttributeName="urn:mace:dir:attribute-def:eduPersonPrincipalName"
768         AttributeNamespace="urn:mace:shibboleth:1.0:attributeNamespace:uri"/>
769     </samlp:AttributeQuery>
770   </samlp:Request>
771 </SOAP-ENV:Body>
772 </SOAP-ENV:Envelope>
```

773 (7) The AA at the IdP processes the request and returns the required attributes to the
774 attribute requester:

```
775 HTTP/1.1 200 OK
776 Content-Type: text/xml
777 Content-Length: nnnn
778
779 <?xml version="1.1" encoding="ISO-8859-1"?>
780 <SOAP-ENV:Envelope
781   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
782   <SOAP-ENV:Header/>
783   <SOAP-ENV:Body>
784     <samlp:Response
785       xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
786       InResponseTo="aaf23196-1773-2113-474a-fe114412ab72"
787       IssueInstant="2004-12-05T09:22:05Z"
788       MajorVersion="1" MinorVersion="1"
789       ResponseID="b07b804c-7c29-ea16-7300-4f3d6f7928ac">
790     <samlp:Status>
791       <samlp:StatusCode Value="samlp:Success"/>
792     </samlp:Status>
793     <!-- insert SAML assertion from section 3.2 here -->
```

```
794     </samlp:Response>
795 </SOAP-ENV:Body>
796 </SOAP-ENV:Envelope>
797 (8) The assertion consumer service filters the attributes, updates the security context and
798 redirects the client to the target resource.
799 (9) The client requests the target resource at the SP (again):
800 https://sp.example.org/myresource
801 (10) Since a security context exists, the SP returns the resource to the client.
```

802 **5.2 Browser/Artifact Profile with Attribute Exchange**

803 Recall that the Browser/Artifact profile of section 4.3 incorporates its own back-channel
804 exchange to dereference the artifact. To obtain attributes, an additional back-channel
805 exchange might be used (as in section 5.1). Alternatively, attributes could be retrieved at the
806 same time the artifact is dereferenced, which we illustrate in this section. The resulting flow
807 is identical to the previous flow associated with the Browser/Artifact profile (see Figure 5).

808 As observed in section 3, an attribute statement may be combined with an authentication
809 statement in one of two ways: both statements may be wrapped in a single assertion or
810 separate assertions may contain individual statements. The latter approach is illustrated
811 here.

812 Two assertions require two artifacts, which is the primary difference between this example
813 and the example of section 4.3. Steps 4–7 of the latter example are modified for two artifacts
814 as follows:

815 (1–3) Same as the ordinary Browser/Artifact profile.

816 (4) The SSO service redirects the client to the assertion consumer service at the SP. A
817 TARGET parameter and *two* SAMLart parameters are appended to the redirect URL.

818 (5) The client requests the assertion consumer service at the SP:

```
819 https://sp.example.org/shibboleth/SSO/Artifact?
820   TARGET=https://sp.example.org/myresource&
821   SAMLart=AAEwGDwd3Z7Fr1GPbM82Fk2CZbpNB7YuJ8gk%2BvmCjh9Y4Wsq6H5%2BKU4C&
822   SAMLart=AAEwGDwd3Z7Fr1GPbM82Fk2CZbpNB8tb%2By6YOO40XGf14QfLKq9xZ8UW
```

823 where one of the artifacts references an authentication assertion while the other artifact
824 corresponds to an attribute assertion (order does not matter).

825 (6) The assertion consumer service validates the request and dereferences both artifacts by
826 sending a single SAML Request to the artifact resolution service at the IdP:

```
827 POST /shibboleth/Artifact HTTP/1.1
828 Host: idp.example.org
829 Content-Type: text/xml
830 Content-Length: nnn
831 SOAPAction: http://www.oasis-open.org/committees/security
832
833 <?xml version="1.1" encoding="ISO-8859-1"?>
834 <SOAP-ENV:Envelope
835   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
836   <SOAP-ENV:Header/>
837   <SOAP-ENV:Body>
838     <samlp:Request
839       xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
840       MajorVersion="1" MinorVersion="1"
841       RequestID="f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
842       IssueInstant="2004-12-05T09:22:04Z">
```

```

843     <samlp:AssertionArtifact>
844         AAEwGDwd3Z7Fr1GPbM82Fk2CZbpNB7YuJ8gk+vmCjh9Y4Wsq6H5+KU4C
845     </samlp:AssertionArtifact>
846     <samlp:AssertionArtifact>
847         AAEwGDwd3Z7Fr1GPbM82Fk2CZbpNB8tb+y6Y0040XGfl4QfLKq9xz8UW
848     </samlp:AssertionArtifact>
849 </samlp:Request>
850 </SOAP-ENV:Body>
851 </SOAP-ENV:Envelope>

```

852 where the values of the `<samlp:AssertionArtifact>` elements are the SAML artifacts
853 at step 5.

854 (7) The artifact resolution service at the IdP returns a `<samlp:Response>` element
855 (containing two assertions) to the assertion consumer service at the SP:

```

856 HTTP/1.1 200 OK
857 Content-Type: text/xml
858 Content-Length: nnnn
859
860 <?xml version="1.1" encoding="ISO-8859-1"?>
861 <SOAP-ENV:Envelope
862     xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
863     <SOAP-ENV:Header/>
864     <SOAP-ENV:Body>
865         <samlp:Response
866             xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
867             MajorVersion="1" MinorVersion="1"
868             Recipient="https://sp.example.org/shibboleth/SSO/Artifact"
869             ResponseID="00099cf1-a355-10f9-9e95-004005b13a2b"
870             InResponseTo="f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
871             IssueInstant="2004-12-05T09:22:05Z">
872             <samlp:Status>
873                 <samlp:StatusCode Value="samlp:Success"/>
874             </samlp:Status>
875             <!-- insert SAML assertions from section 3.3 here -->
876         </samlp:Response>
877     </SOAP-ENV:Body>
878 </SOAP-ENV:Envelope>

```

879 (8–10) Same as the ordinary Browser/Artifact profile.

880 **5.3 Directory Schema**

881 Shibboleth, by virtue of SAML, has a well-defined attribute exchange mechanism, but neither
882 specification defines any attributes per se. Instead it is left to individual implementations to
883 define their own attributes. In a cross-domain federated environment, in particular, a
884 standard approach to user attributes is crucial. In the absence of such standards, federation
885 on a large scale will be difficult if not impossible.

886 To address this issue within the higher education community, Internet2 and EDUCAUSE
887 have jointly developed a set of attributes and associated bindings called *eduPerson*
888 [*eduPerson*]. The LDAP binding of *eduPerson* is derived from the standard LDAP object
889 class called *inetOrgPerson* [RFC 2798] which itself is based on other standard LDAP
890 classes (see e.g., [RFC 2256]).

891 Of all the attributes defined by this hierarchy of object classes, approximately 40 attributes
892 have been defined by InCommon as *common identity attributes* [*InCommonAttrs*]:

Common Attributes

Highly recommended	6
Suggested	10
Optional	25
	41

893 For example, here are InCommon's six "highly recommended" attributes:

Attribute Name	Attribute Value
givenName	Mary
sn (surname)	Smith
cn (common name)	Mary Smith
eduPersonScopedAffiliation	student@idp.example.org
eduPersonPrincipalName	mary.smith@idp.example.org
eduPersonTargetedID	—

894 The attribute `eduPersonTargetedID` does not have a precise value syntax. See the
895 *EduPerson Specification* [`eduPersonSpec`] for more information about
896 `eduPersonTargetedID`.

897 6 Metadata

898 Both IdPs and SPs publish information about themselves in special XML files called
899 *metadata* files. Since metadata precisely identify the provider and the services it offers,
900 metadata files are digitally signed for integrity protection.

901 A SAML metadata specification was first published as part of SAML 2.0 [SAML2Meta]. Since
902 that time, the OASIS Security Services Technical Committee has considered adopting a
903 minimal subset of SAML 2.0 metadata for use with SAML 1.x as specified in [SAML1Meta].
904 Shibboleth further constrains this profile in [ShibProt].

905 Only four metadata descriptors are defined for use within Shibboleth:

- 906 1. <md:IDPSSODescriptor>
- 907 2. <md:SPSSODescriptor>
- 908 3. <md:AuthnAuthorityDescriptor>
- 909 4. <md:AttributeAuthorityDescriptor>

910 These metadata elements correspond to the SSO service (IdP), the assertion consumer
911 service (SP), the authentication authority (IdP), and the attribute authority (IdP), respectively.
912 We will omit the <md:AuthnAuthorityDescriptor> element in what follows since the
913 authentication authority operates behind the scenes in a typical Shibboleth deployment, in
914 which case it need not publish metadata about itself.

915 Multiple <md:EntityDescriptor> elements may be grouped together in a single
916 <md:EntitiesDescriptor> element:

```
917 <md:EntitiesDescriptor  
918   xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"  
919   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
920   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
921   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
922   xsi:schemaLocation="urn:oasis:names:tc:SAML:2.0:metadata ..."  
923   Name="urn:mace:shibboleth:examples"  
924   validUntil="2010-01-01T00:00:00Z">  
925   <!-- insert ds:Signature element (section 3.4) -->  
926   <!-- insert md:EntityDescriptor elements here -->  
927 </md:EntitiesDescriptor>
```

928 Note that an <md:EntitiesDescriptor> element may be signed (see section 3.4).
929 Alternatively, the individual <md:EntityDescriptor> elements may be signed (see the
930 examples in the following subsections).

931 6.1 Identity Provider Metadata

932 An identity provider publishes data about itself in an <md:EntityDescriptor> element:

```
933 <md:EntityDescriptor  
934   xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"  
935   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"  
936   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"  
937   entityID="https://idp.example.org/shibboleth">  
938   <!-- insert ds:Signature element (section 3.4) -->  
939   <!-- insert md:IDPSSODescriptor element (section 6.1.1) -->  
940   <!-- insert md:AttributeAuthorityDescriptor element (section 6.1.2) -->  
941   <md:Organization>  
942     <md:OrganizationName xml:lang="en">  
943       Shibboleth Identity Provider
```

```

944 </md:OrganizationName>
945 <md:OrganizationDisplayName xml:lang="en">
946   Shibboleth Identity Provider @ Some Location
947 </md:OrganizationDisplayName>
948 <md:OrganizationURL xml:lang="en">
949   http://www.idp.example.org/
950 </md:OrganizationURL>
951 </md:Organization>
952 <md:ContactPerson contactType="technical">
953   <md:SurName>Shibboleth IdP Support</md:SurName>
954   <md:EmailAddress>shib-support@idp.example.org</md:EmailAddress>
955 </md:ContactPerson>
956 </md:EntityDescriptor>

```

957 The `entityID` attribute is the unique providerId of the IdP. Note that the details of the
958 digital signature (in the `<ds:Signature>` element) have been omitted from this example.
959 See section 3.4 for relevant discussion.

960 The IdP manages an SSO service and an attribute authority, each having its own descriptor.
961 These are detailed in the following subsections.

962 6.1.1 SSO Service Metadata

963 The SSO service at the IdP is encapsulated in an `<md:IDPSSODescriptor>` element:

```

964 <md:IDPSSODescriptor
965   protocolSupportEnumeration="urn:oasis:names:tc:SAML:1.1:protocol
966   urn:mace:shibboleth:1.0">
967   <md:KeyDescriptor use="signing">
968     <ds:KeyInfo>
969       <ds:KeyName>IdP SSO Key</ds:KeyName>
970     </ds:KeyInfo>
971   </md:KeyDescriptor>
972   <md:ArtifactResolutionService isDefault="true" index="0"
973     Binding="urn:oasis:names:tc:SAML:1.0:bindings:SOAP-binding"
974     Location="https://idp.example.org/shibboleth/Artifact"/>
975   <md:NameIDFormat>
976     urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress
977   </md:NameIDFormat>
978   <md:NameIDFormat>
979     urn:mace:shibboleth:1.0:nameIdentifier
980   </md:NameIDFormat>
981   <md:SingleSignOnService
982     Binding="urn:mace:shibboleth:1.0:profiles:AuthnRequest"
983     Location="https://idp.example.org/shibboleth/SSO"/>
984 </md:IDPSSODescriptor>

```

985 The previous metadata element describes the SSO service at the IdP of section 4. Note the
986 following details about this element:

- 987 • Since Shibboleth supports an SSO service (whereas SAML1 does not), protocol
988 support includes the URI
989 `urn:mace:shibboleth:1.0`
990 which is a unique identifier specified by [ShibProt].
- 991 • Key information has been omitted for brevity.
- 992 • The `Binding` attribute of the `<md:ArtifactResolutionService>` element
993 indicates that the SAML SOAP binding should be used (see [SAMLBind]).

- 994 • The `Location` attribute of the `<md:ArtifactResolutionService>` element is
995 used in step 6 of the Browser/Artifact profile of section 4.3.
- 996 • The `index` attribute of the `<md:ArtifactResolutionService>` element is
997 ignored.
- 998 • The `<md:NameIDFormat>` elements indicate what SAML *NameIdentifier formats*
999 [SAMLCore] the SSO service supports. One of these is a standard Shibboleth *handle*
1000 indicated by URI
1001 `urn:mace:shibboleth:1.0:nameIdentifier`
- 1002 • The `Binding` attribute of the `<md:SingleSignOnService>` element is a unique
1003 identifier specified by [ShibProt].
- 1004 • The `Location` attribute of the `<md:SingleSignOnService>` element is used by
1005 SPs and WAYFs to redirect the client to the SSO service at the IdP.

1006 6.1.2 Attribute Authority Metadata

1007 The attribute authority at the IdP is given by an `<md:AttributeAuthorityDescriptor>`
1008 element:

```
1009 <md:AttributeAuthorityDescriptor
1010   protocolSupportEnumeration="urn:oasis:names:tc:SAML:1.1:protocol">
1011   <md:KeyDescriptor use="signing">
1012     <ds:KeyInfo>
1013       <ds:KeyName>IdP AA Key</ds:KeyName>
1014     </ds:KeyInfo>
1015   </md:KeyDescriptor>
1016   <md:AttributeService
1017     Binding="urn:oasis:names:tc:SAML:1.0:bindings:SOAP-binding"
1018     Location="https://idp.example.org/shibboleth/AA/SOAP"/>
1019   <saml:Attribute
1020     Name="urn:mace:dir:attribute-def:eduPersonPrincipalName"
1021     NameFormat="urn:mace:shibboleth:1.0:attributeNamespace:uri"/>
1022   <saml:Attribute
1023     Name="urn:mace:dir:attribute-def:eduPersonScopedAffiliation"
1024     NameFormat="urn:mace:shibboleth:1.0:attributeNamespace:uri">
1025     <saml:AttributeValue>member</saml:AttributeValue>
1026     <saml:AttributeValue>student</saml:AttributeValue>
1027     <saml:AttributeValue>faculty</saml:AttributeValue>
1028     <saml:AttributeValue>employee</saml:AttributeValue>
1029     <saml:AttributeValue>staff</saml:AttributeValue>
1030   </saml:Attribute>
1031   <saml:Attribute
1032     Name="urn:mace:dir:attribute-def:eduPersonEntitlement"
1033     NameFormat="urn:mace:shibboleth:1.0:attributeNamespace:uri"/>
1034   <md:NameIDFormat>
1035     urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress
1036   </md:NameIDFormat>
1037   <md:NameIDFormat>
1038     urn:mace:shibboleth:1.0:nameIdentifier
1039   </md:NameIDFormat>
1040 </md:AttributeAuthorityDescriptor>
```

1041 This element is not needed for the SSO profiles of section 4, but it is required for the attribute
1042 exchanges outlined in sections 5.1 and 5.2.

1043 **6.2 Service Provider Metadata**

1044 A service provider also publishes data about itself in an `<md:EntityDescriptor>`
1045 element:

```
1046 <md:EntityDescriptor
1047   xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
1048   xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
1049   xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
1050   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1051   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1052   entityID="https://sp.example.org/shibboleth">
1053   <!-- insert ds:Signature element (section 3.4) -->
1054   <!-- insert md:SPSSODescriptor element (section 6.2.1) -->
1055   <md:Organization>
1056     <md:OrganizationName xml:lang="en">
1057       Shibboleth Service Provider
1058     </md:OrganizationName>
1059     <md:OrganizationDisplayName xml:lang="en">
1060       Shibboleth Service Provider @ Some Location
1061     </md:OrganizationDisplayName>
1062     <md:OrganizationURL xml:lang="en">
1063       http://www.sp.example.org/
1064     </md:OrganizationURL>
1065   </md:Organization>
1066   <md:ContactPerson contactType="technical">
1067     <md:SurName>Shibboleth SP Support</md:SurName>
1068     <md:EmailAddress>shib-support@sp.example.org</md:EmailAddress>
1069   </md:ContactPerson>
1070 </md:EntityDescriptor>
```

1071 The primary component managed by the SP is the assertion consumer service, which is
1072 discussed below.

1073 **6.2.1 Assertion Consumer Service Metadata**

1074 The assertion consumer service is represented by an `<md:SPSSODescriptor>` element:

```
1075 <md:SPSSODescriptor
1076   protocolSupportEnumeration="urn:oasis:names:tc:SAML:1.1:protocol">
1077   <md:KeyDescriptor use="signing">
1078     <ds:KeyInfo>
1079       <ds:KeyName>SP SSO Key</ds:KeyName>
1080     </ds:KeyInfo>
1081   </md:KeyDescriptor>
1082   <md:NameIDFormat>
1083     urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress
1084   </md:NameIDFormat>
1085   <md:NameIDFormat>
1086     urn:mace:shibboleth:1.0:nameIdentifier
1087   </md:NameIDFormat>
1088   <md:AssertionConsumerService isDefault="true" index="0"
1089     Binding="urn:oasis:names:tc:SAML:1.0:profiles:browser-post"
1090     Location="https://sp.example.org/shibboleth/SSO/POST"/>
1091   <md:AssertionConsumerService index="1"
1092     Binding="urn:oasis:names:tc:SAML:1.0:profiles:artifact-01"
1093     Location="https://sp.example.org/shibboleth/SSO/Artifact"/>
1094   <md:AttributeConsumingService isDefault="true" index="0">
1095     <md:ServiceName xml:lang="en">
```

```
1096     Service Provider Portal
1097 </md:ServiceName>
1098 <md:RequestedAttribute
1099     Name="urn:mace:dir:attribute-def:eduPersonScopedAffiliation"
1100     NameFormat="urn:mace:shibboleth:1.0:attributeNamespace:uri">
1101 </md:RequestedAttribute>
1102 <md:RequestedAttribute
1103     Name="urn:mace:dir:attribute-def:eduPersonEntitlement"
1104     NameFormat="urn:mace:shibboleth:1.0:attributeNamespace:uri">
1105     <saml:AttributeValue xsi:type="xsd:anyURI">
1106         https://sp.example.org/entitlements/123456789
1107     </saml:AttributeValue>
1108 </md:RequestedAttribute>
1109 </md:AttributeConsumingService>
1110 </md:SPSSODescriptor>
```

1111 Note the following details about the `<md:SPSSODescriptor>` metadata element:

- 1112 • Key information has been omitted for brevity.
- 1113 • The `Binding` attributes of the `<md:AssertionConsumerService>` elements are
1114 standard URIs specified by [SAMLMeta].
- 1115 • The `Location` attribute of the first `<md:AssertionConsumerService>` element
1116 (`index="0"`) is used in step 3 of the Browser/POST profile in section 4.2.
- 1117 • The `Location` attribute of the second `<md:AssertionConsumerService>`
1118 element (`index="1"`) is used in step 3 of the Browser/Artifact profile in section 4.3.
- 1119 • The `<md:AttributeConsumingService>` element is used by the IdP to formulate
1120 an attribute assertion that is *pushed* to the SP in step 7 of the Browser/Artifact profile
1121 outlined in section 5.2.
- 1122 • The `index` attributes of the `<md:AssertionConsumerService>` and
1123 `<md:AttributeConsumingService>` elements are ignored.

1124 As noted above, the values of the `Location` attribute are the same as the values of the
1125 `shire` parameter in the corresponding profile. Upon receiving the authentication request,
1126 the IdP checks the value of the `shire` parameter against the locations in SP metadata,
1127 which minimizes the possibility of a rogue SP orchestrating a man-in-the-middle attack.

1128 7 References

1129 7.1 Normative References

- 1130 [eduPersonSpec] *EduPerson Specification* (200312), <http://www.nmi->
1131 [edit.org/eduPerson/internet2-mace-dir-eduperson-200312.html](http://www.nmi-edit.org/eduPerson/internet2-mace-dir-eduperson-200312.html)
- 1132 [RFC 2256] M. Wahl, *A Summary of the X.500(96) User Schema for use with LDAPv3*,
1133 December 1997. <http://www.ietf.org/rfc/rfc2256.txt>
- 1134 [RFC 2798] M. Smith, *Definition of the inetOrgPerson LDAP Object Class*, April 2000.
1135 <http://www.ietf.org/rfc/rfc2798.txt>
- 1136 [SAML2Meta] S. Cantor et al., *Metadata for the OASIS Security Assertion Markup Language*
1137 *(SAML) V2.0*. OASIS SSTC, 15 March 2005. Document ID saml-metadata-2.0-os
1138 <http://www.oasis-open.org/committees/security/>
- 1139 [SAMLBind] E. Maler et al., *Bindings and Profiles for the OASIS Security Assertion Markup*
1140 *Language (SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-bindings-profiles-
1141 1.1 <http://www.oasis-open.org/committees/security/>
- 1142 [SAMLCore] E. Maler et al., *Assertions and Protocols for the OASIS Security Assertion*
1143 *Markup Language (SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-core-1.1
1144 <http://www.oasis-open.org/committees/security/>
- 1145 [SAMLMeta] G. Whitehead and S. Cantor, *SAML 1.x Metadata Profile*. OASIS SSTC,
1146 February 2005. Document ID draft-saml1x-metadata-04 <http://www.oasis->
1147 [open.org/committees/security/](http://www.oasis-open.org/committees/security/)
- 1148 [ShibProt] S. Cantor et al., *Shibboleth Architecture: Protocols and Profiles*. Internet2-MACE,
1149 February 2005. Document ID draft-mace-shibboleth-arch-protocols-09
1150 <http://shibboleth.internet2.edu/>
- 1151 [SOAP] D. Box et al., *Simple Object Access Protocol (SOAP) 1.1*. W3C Note 08 May 2000.
1152 <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>

1153 7.2 Non-Normative References

- 1154 [eduPerson] *eduPerson Object Class*. <http://www.educause.edu/eduperson/>
- 1155 [HTTP] *HTTP - Hypertext Transfer Protocol*. World Wide Web Consortium (W3C).
1156 <http://www.w3.org/Protocols/>
- 1157 [InCommonAttribs] *InCommon Federation: Common Identity Attributes*.
1158 <http://www.incommonfederation.org/docs/policies/federatedattributes.html>
- 1159 [GSIsecurity] K. Bhatia et al., *Engineering an End-to-End GSI-based Security Infrastructure*.
1160 <http://users.sdsc.edu/~chandras/Papers/ccgrid-submission.pdf>
- 1161 [SAMLTech] J. Hughes et al. *Technical Overview of the OASIS Security Assertion Markup*
1162 *Language (SAML) V1.1*. OASIS, May 2004. Document ID sstc-saml-tech-overview-1.1-cd
1163 <http://www.oasis-open.org/committees/security/>
- 1164 [XML] *XML Core Working Group Public Page*. World Wide Web Consortium (W3C).
1165 <http://www.w3.org/XML/Core/>
- 1166 [XMLSchema] *XML Schema*. World Wide Web Consortium (W3C).
1167 <http://www.w3.org/XML/Schema>
- 1168 [XMLSig] *XML Signature WG*. World Wide Web Consortium (W3C).
1169 <http://www.w3.org/Signature/>