



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

Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML)

Document identifier: draft-sstc-bindings-model-13

Location: <http://www.oasis-open.org/committees/security/docs>

Publication date: 29 March 2002

Maturity Level: Committee working draft

Send comments to: security-services-comment@lists.oasis-open.org *unless* you are subscribed to the security-services list for committee members -- send comments there if so. Note: Before sending messages to the security-services-comment list, you must first subscribe. To subscribe, send an email message to security-services-comment-request@lists.oasis-open.org with the word "subscribe" as the body of the message.

Editor:

Prateek Mishra, Netegrity, pmishra@netegrity.com

Contributors:

- Bob Blakley, Tivoli
- Scott Cantor, Ohio State University
- Marlena Erdos, Tivoli
- Chris Ferris, Sun Microsystems
- Simon Godik, Crosslogix
- Jeff Hodges, Oblix
- Eve Maler, Sun Microsystems
- RL "Bob" Morgan, University of Washington
- Tim Moses, Entrust
- Evan Prodromou, Securant
- Irving Reid, Baltimore
- Krishna Sankar, Cisco Systems

Rev	Date	By Whom	What
05	18 August 2001	Prateek Mishra	Bindings model draft
0.6	8 November 2001	Prateek Mishra	Removed SAML HTTP binding, removed artifact PUSH case, updated SOAP profile based on Blakley note
0.7	3 December 2001	Prateek Mishra	Re-structured based on F2F#5 comments; separated discussion and normative language
0.8	24 December 2001	Eve Maler, Prateek Mishra	Edited for public consumption; Incorporates comments from reviewers (Tim, Simon, Irving) and all f2f#5 changes; Developmental edit on the back half of the draft, plus random small edits to the whole document
0.9	9 January 2002	Prateek Mishra	Includes “required information” for each binding and profile; includes Tim’s alternative artifact format
10	10 February 2002	Prateek Mishra	Removed SOAP Profile; added note on obsolete XML schema namespace in SOAP binding.
11	15 February 2002	Prateek Mishra	Fixed typographical errors, binding and profile URIs
12	8 March 2002	Prateek Mishra	200203/msg00030.html 200203/msg00003.html 200202/msg00207.html 200202/msg00181.html 200203/msg00034.html
<u>13</u>	<u>25 March 2002</u>	<u>Prateek Mishra</u>	<u>200203/msg00118.html</u> <u>200203/msg00152.html</u> <u>200203/msg00042.html (ELM-7)</u> <u>200201/msg00225.html</u>

38

39

39	<u>Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML)</u>	<u>14</u>
40	<u>1 Introduction</u>	<u>55</u>
41	<u>1.1 Protocol Binding and Profile Concepts</u>	<u>55</u>
42	<u>1.2 Notation</u>	<u>55</u>
43	<u>2 Specification of Additional Protocol Bindings and Profiles</u>	<u>66</u>
44	<u>2.1 Guidelines for Specifying Protocol Bindings and Profiles</u>	<u>66</u>
45	<u>2.2 Process Framework for Describing and Registering Protocol Bindings and Profiles</u>	<u>77</u>
46	<u>3 Protocol Bindings</u>	<u>77</u>
47	<u>3.1 SOAP Binding for SAML</u>	<u>77</u>
48	<u>3.1.1 Required Information</u>	<u>88</u>
49	<u>3.1.2 Protocol-Independent Aspects of the SAML SOAP Binding</u>	<u>88</u>
50	<u>3.1.2.1 Basic Operation</u>	<u>88</u>
51	<u>3.1.2.2 SOAP Headers</u>	<u>99</u>
52	<u>3.1.2.3 Authentication</u>	<u>99</u>
53	<u>3.1.2.4 Message Integrity</u>	<u>99</u>
54	<u>3.1.2.5 Confidentiality</u>	<u>99</u>
55	<u>3.1.3 Use of SOAP over HTTP</u>	<u>1040</u>
56	<u>3.1.3.1 HTTP Headers</u>	<u>1040</u>
57	<u>3.1.3.2 Authentication</u>	<u>1040</u>
58	<u>3.1.3.3 Message Integrity</u>	<u>1040</u>
59	<u>3.1.3.4 Message Confidentiality</u>	<u>1144</u>
60	<u>3.1.3.5 Security Considerations</u>	<u>1144</u>
61	<u>3.1.3.6 Error Reporting</u>	<u>1144</u>
62	<u>3.1.3.7 Example SAML Message Exchange Using SOAP over HTTP</u>	<u>1144</u>
63	<u>4 Profiles</u>	<u>1242</u>
64	<u>4.1 Web Browser SSO Profiles of SAML</u>	<u>1242</u>
65	<u>4.1.1 Browser/Artifact Profile of SAML</u>	<u>1414</u>
66	<u>4.1.1.1 Required Information</u>	<u>1414</u>
67	<u>4.1.1.2 Preliminaries</u>	<u>1414</u>
68	<u>4.1.1.3 Step 1: Accessing the Inter-Site Transfer Service</u>	<u>1545</u>
69	<u>4.1.1.4 Step 2: Redirecting to the Destination Site</u>	<u>1616</u>
70	<u>4.1.1.5 Step 3: Accessing the Assertion Consumer Service</u>	<u>1616</u>
71	<u>4.1.1.6 Steps 4 and 5: Acquiring the Corresponding Assertions</u>	<u>1747</u>

72	<u>4.1.1.7 Step 6: Responding to the User’s Request for a Resource</u>	<u>1717</u>
73	<u>4.1.1.8 Artifact Format</u>	<u>1818</u>
74	<u>4.1.1.9 Threat Model and Countermeasures</u>	<u>1919</u>
75	<u>4.1.2 Browser/POST Profile of SAML</u>	<u>2121</u>
76	<u>4.1.2.1 Required Information</u>	<u>2121</u>
77	<u>4.1.2.2 Preliminaries</u>	<u>2121</u>
78	<u>4.1.2.3 Step 1: Accessing the Inter-Site Transfer Service</u>	<u>2121</u>
79	<u>4.1.2.4 Step 2: Generating and Supplying the Response</u>	<u>2222</u>
80	<u>4.1.2.5 Step 3: Posting the Form Containing the Response</u>	<u>2222</u>
81	<u>4.1.2.6 Step 4: Responding to the User’s Request for a Resource</u>	<u>2424</u>
82	<u>4.1.2.7 Threat Model and Countermeasures</u>	<u>2424</u>
83	<u>5 Use of SSL 3.0 or TLS 1.0</u>	<u>2525</u>
84	<u>5.1 SAML SOAP Binding</u>	<u>2525</u>
85	<u>5.2 Web Browser Profiles of SAML</u>	<u>2525</u>
86	<u>6 References</u>	<u>2626</u>
87	<u>7 URL Size Restriction (Non-Normative)</u>	<u>2727</u>
88	<u>8 Alternative SAML Artifact Format</u>	<u>2828</u>
89	<u>8.1 Required Information</u>	<u>2828</u>
90	<u>8.2 Format Details</u>	<u>2929</u>
91	<u>Appendix A. Notices</u>	<u>3030</u>
92		
93		
94		
95		
96		
97		
98		
99		
100		
101		
102		

1 Introduction

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

A separate specification [SAMLCore] defines the SAML assertions and request-response messages themselves.

1.1 Protocol Binding and Profile Concepts

Mappings from SAML request-response message exchanges into standard messaging or communication protocols are called SAML *protocol bindings* (or just *bindings*). An instance of mapping SAML request-response message exchanges into a specific protocol <FOO> is termed a <FOO> *binding for SAML* or a *SAML <FOO> binding*.

For example, an HTTP binding for SAML describes how SAML request and response message exchanges are mapped into HTTP message exchanges. A SAML SOAP binding describes how SAML request and response message exchanges are mapped into SOAP message exchanges.

Sets of rules describing how to embed and extract SAML assertions into a framework or protocol are called *profiles of SAML*. A profile describes how SAML assertions are embedded in or combined with other objects (for example, files of various types, or protocol data units of communication protocols) by an originating party, communicated from the originating site to a destination, and subsequently processed at the destination. A particular set of rules for embedding SAML assertions into and extracting them from a specific class of <FOO> objects is termed a <FOO> *profile of SAML*.

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

The intent of this specification is to specify a selected set of bindings and profiles in sufficient detail to ensure that independently implemented products will interoperate.

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

1.2 Notation

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

Listings of productions or other normative code appear like this.

Example code listings appear like this.

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

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

- 141 • The prefix `saml`: stands for the SAML assertion namespace [SAMLCore].
- 142 • The prefix `samlp`: stands for the SAML request-response protocol namespace
143 [SAMLCore].
- 144 • The prefix `ds`: stands for the W3C XML Signature namespace,
145 `http://www.w3.org/2000/09/xmldsig#` [XMLSig].
- 146 • The prefix `SOAP-ENV`: stands for the SOAP 1.1 namespace,
147 `http://schemas.xmlsoap.org/soap/envelope` [SOAP1.1].

148 This specification uses the following typographical conventions in text: `<SAMLElement>`,
149 `<ns:ForeignElement>`, `Attribute`, `OtherCode`. In some cases, angle brackets are used to
150 indicate nonterminals, rather than XML elements; the intent will be clear from the context.

151 **2 Specification of Additional Protocol** 152 **Bindings and Profiles**

153 This specification defines a selected set of protocol bindings and profiles, but others will need to
154 be developed. It is not possible for the OASIS SAML Technical Committee to standardize all of
155 these additional bindings and profiles for two reasons: it has limited resources and it does not
156 own the standardization process for all of the technologies used. The following sections offer
157 guidelines for specifying bindings and profiles and a process framework for describing and
158 registering them.

159 **2.1 Guidelines for Specifying Protocol Bindings and Profiles**

160 This section provides a checklist of issues that **MUST** be addressed by each protocol binding and
161 profile.

- 162 1. Describe the set of interactions between parties involved in the binding or profile. Any
163 restriction on applications used by each party and the protocols involved in each
164 interaction must be explicitly called out.
- 165 2. Identify the parties involved in each interaction, including: how many parties are
166 involved, and whether intermediaries may be involved.
- 167 3. Specify the method of authentication of parties involved in each interaction, including
168 whether authentication is required and acceptable authentication types.
- 169 4. Identify the level of support for message integrity. What mechanisms are used to ensure
170 message integrity?

- 171 5. Identify the level of support for confidentiality, including whether a third party may view
172 the contents of SAML messages and assertions, whether the binding or profile requires
173 confidentiality and the mechanisms recommended for achieving confidentiality.
- 174 6. Identify the error states, including the error states at each participant, especially those that
175 receive and process SAML assertions or messages.
- 176 7. Identify security considerations, including analysis of threats and description of
177 countermeasures.

178 **2.2 Process Framework for Describing and Registering** 179 **Protocol Bindings and Profiles**

180 For any new protocol binding or profile to be interoperable, it needs to be openly specified. The
181 OASIS SAML Technical Committee will maintain a registry and repository of submitted
182 bindings and profiles titled “Additional Bindings and Profiles” at the SAML website
183 (<http://www.oasis-open.org/committees/security/>) in order to keep the SAML community
184 informed. The Committee will also provide instructions for submission of bindings and profiles
185 by OASIS members.

186 When a profile or protocol binding is registered, the following information MUST be supplied:

- 187 1. Identification: Specify a URI that uniquely identifies this protocol binding or profile.
- 188 2. Contact information: Specify the postal or electronic contact information for the author of
189 the protocol binding or profile.
- 190 3. Description: Provide a text description of the protocol binding or profile. The description
191 SHOULD follow the guidelines in Section [2.10](#).
- 192 4. Updates: Provide references to previously registered protocol bindings or profiles that the
193 current entry improves or obsoletes.

194 **3 Protocol Bindings**

195 The following sections define SAML protocol bindings sanctioned by the OASIS SAML
196 Committee. Only one binding, the SAML SOAP binding, is defined.

197 **3.1 SOAP Binding for SAML**

198

199 SOAP (Simple Object Access Protocol) 1.1 [**SOAP1.1**] is a specification for RPC-like
200 interactions and message communications using XML and HTTP. It has three main parts. One is
201 a message format that uses an envelope and body metaphor to wrap XML data for transmission
202 between parties. The second is a restricted definition of XML data for making strict RPC-like
203 calls through SOAP, without using a predefined XML schema. Finally, it provides a binding for
204 SOAP messages to HTTP and extended HTTP.

205 The SAML SOAP binding defines how to use SOAP to send and receive SAML requests and
206 responses.

207 Like SAML, SOAP can be used over multiple underlying transports. This binding has protocol-
208 independent aspects, but also calls out the use of SOAP over HTTP as REQUIRED (mandatory
209 to implement).

210 **3.1.1 Required Information**

211 Identification:

212 <http://www.oasis-open.org/security/>
213 [urn:oasis:names:tc:SAML:1.0:draft-sstc-bindings-model-132:/bindings:/SOAP-](http://www.oasis-open.org/security/urn:oasis:names:tc:SAML:1.0:draft-sstc-bindings-model-132:/bindings:/SOAP-)
214 binding

215 Contact information:

216 security-services-comment@lists.oasis-open.org

217 Description: Given below.

218 Updates: None.

219 **3.1.2 Protocol-Independent Aspects of the SAML SOAP Binding**

220 The following sections define aspects of the SAML SOAP binding that are independent of the
221 underlying protocol, such as HTTP, on which the SOAP messages are transported.

222 **3.1.2.1 Basic Operation**

223 SOAP messages consist of three elements: an envelope, header data, and a message body. SAML
224 request-response protocol elements MUST be enclosed within the SOAP message body.

225 SOAP 1.1 also defines an optional data encoding system. This system is not used within the
226 SAML SOAP binding. This means that SAML messages can be transported using SOAP without
227 re-encoding from the "standard" SAML schema to one based on the SOAP encoding.

228 The system model used for SAML conversations over SOAP is a simple request-response model.

- 229 1. A system entity acting as a SAML requester transmits a SAML <Request> element
230 within the body of a SOAP message to a system entity acting as a SAML responder. The
231 SAML requester MUST NOT include more than one SAML request per SOAP message
232 or include any additional XML elements in the SOAP body.
- 233 2. The SAML responder MUST return either a <Response> element within the body of
234 another SOAP message or a SOAP fault code. The SAML responder MUST NOT
235 include more than one SAML response per SOAP message or include any additional
236 XML elements in the SOAP body. If a SAML responder cannot, for some reason, process
237 a SAML request, it MUST return a SOAP fault code. SOAP fault codes MUST NOT be
238 sent for errors within the SAML problem domain, for example, inability to find an
239 extension schema or as a signal that the subject is not authorized to access a resource in

240 an authorization query. (SOAP 1.1 faults and fault codes are discussed in [SOAP1.1]
241 §4.1.)

242
243 On receiving a SAML response in a SOAP message, the SAML requester MUST NOT send a
244 fault code or other error messages to the SAML responder. Because the format for the message
245 interchange is a simple request-response pattern, adding additional items such as error conditions
246 would needlessly complicate the protocol.

247 [SOAP1.1] references an early draft of the XML Schema specification including an obsolete
248 namespace. SAML requesters SHOULD generate SOAP documents referencing only the final
249 XML schema namespace. SAML responders MUST be able to process both the XML schema
250 namespace used in [SOAP1.1] as well as the final XML schema namespace.

251 **3.1.2.2 SOAP Headers**

252 A SAML requester in a SAML conversation over SOAP MAY add arbitrary headers to the
253 SOAP message. This binding does not define any additional SOAP headers.

254 **Note:** The reason other headers need to be allowed is that some SOAP
255 software and libraries might add headers to a SOAP message that are out of
256 the control of the SAML-aware process. Also, some headers might be needed
257 for underlying protocols that require routing of messages.

258 A SAML responder MUST NOT require any headers for the SOAP message.

259 **Note:** The rationale is that requiring extra headers will cause fragmentation
260 of the SAML standard and will hurt interoperability.

261 **3.1.2.3 Authentication**

262 Authentication of both the SAML requester and responder is OPTIONAL and depends on the
263 environment of use. Authentication protocols available from the underlying substrate protocol
264 MAY be utilized to provide authentication. Section 3.1.2.2 describes authentication in the SOAP
265 over HTTP environment.

266 **3.1.2.4 Message Integrity**

267 Message integrity of both SAML request and response is OPTIONAL and depends on the
268 environment of use. The security layer in the underlying substrate protocol MAY be used to
269 ensure message integrity. Section 3.1.2.3 describes support for message integrity in the SOAP
270 over HTTP environment.

271 **3.1.2.5 Confidentiality**

272 Confidentiality of both SAML request and response is OPTIONAL and depends on the
273 environment of use. The security layer in the underlying substrate protocol MAY be used to

274 ensure message confidentiality. Section 3.1.2.4 describes support for confidentiality in the SOAP
275 over HTTP environment.

276 **3.1.3 Use of SOAP over HTTP**

277 A SAML processor that claims conformance to the SAML SOAP binding MUST implement
278 SAML over SOAP over HTTP. This section describes certain specifics of using SOAP over
279 HTTP, including HTTP headers, error reporting, authentication, message integrity and
280 confidentiality.

281 The HTTP binding for SOAP is described in [SOAP1.1] §6.0. It requires the use of a
282 SOAPAction header as part of a SOAP HTTP request. A SAML responder MUST NOT depend
283 on the value of this header. A SAML requester MAY set the value of SOAPAction header as
284 follows:

285 <http://www.oasis-open.org/committees/security>

286 **3.1.3.1 HTTP Headers**

287 HTTP proxies MUST NOT cache responses carrying SAML assertions.

288 Both of the following conditions apply when using HTTP 1.1:

- 289 • If the value of the Cache-Control header field is **not** set to no-store, then the SAML
290 responder MUST NOT include the Cache-Control header field in the response.
- 291 • If the Expires response header field is **not** disabled by a Cache-Control header field
292 with a value of no-store, then the Expires field SHOULD NOT be included.

293 There are no other restrictions on HTTP headers.

294 **3.1.3.2 Authentication**

295 The SAML requester and responder MUST implement the following authentication methods:

- 296 1. No client or server authentication.
- 297 2. HTTP basic client authentication [RFC2617] with and without SSL 3.0 or TLS 1.0.
- 298 3. HTTP over SSL 3.0 or TLS 1.0 (see Section 50) server authentication with a server-side
299 certificate.

300 4. HTTP over SSL 3.0 or TLS 1.0 client authentication with a client-side certificate.

301 If a SAML responder uses SSL 3.0 or TLS 1.0, it MUST use a server-side certificate.

302 **3.1.3.3 Message Integrity**

303 When message integrity needs to be guaranteed, SAML responders MUST use HTTP over SSL
304 3.0 or TLS1.0 (see Section 50) with a server-side certificate.

305 **3.1.3.4 Message Confidentiality**

306 When message confidentiality is required, SAML responders MUST use HTTP over SSL 3.0 or
307 TLS 1.0 (see Section [50](#)) with a server-side certificate.

308 **3.1.3.5 Security Considerations**

309 Before deployment, each combination of authentication, message integrity and confidentiality
310 mechanisms SHOULD be analyzed for vulnerability in the context of the deployment
311 environment. See the SAML security considerations document [[SAMLSec](#)] for a detailed
312 discussion.

313 RFC 2617 [[RFC2617](#)] describes possible attacks in the HTTP environment when basic or
314 message-digest authentication schemes are used.

315 **3.1.3.6 Error Reporting**

316 A SAML responder that refuses to perform a message exchange with the SAML requester
317 SHOULD return a "403 Forbidden" response. In this case, the content of the HTTP body is not
318 significant.

319 As described in [[SOAP1.1](#)] § 6.2, in the case of a SOAP error while processing a SOAP request,
320 the SOAP HTTP server MUST return a "500 Internal Server Error" response and include a
321 SOAP message in the response with a SOAP fault element. This type of error SHOULD be
322 returned for SOAP-related errors detected before control is passed to the SAML processor, or
323 when the SOAP processor reports an internal error (for example, the SOAP XML namespace is
324 incorrect, the SAML schema cannot be located, the SAML processor throws an exception, and
325 so on).

326 In the case of a SAML processing error, the SOAP HTTP server MUST respond with "200 OK"
327 and include a SAML-specified error description as the only child of the <SOAP-ENV:Body>
328 element. For more information about SAML error codes, see the SAML assertion and protocol
329 specification [[SAMLCore](#)].

330 **3.1.3.7 Example SAML Message Exchange Using SOAP over HTTP**

331 Following is an example of a request that asks for an assertion containing an authentication
332 statement from a SAML authentication authority.

```
333 POST /SamlService HTTP/1.1
334 Host: www.example.com
335 Content-Type: text/xml
336 Content-Length: nnn
337 SOAPAction: http://www.oasis-open.org/committees/security
338 <SOAP-ENV:Envelope
339   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" >
340   <SOAP-ENV:Body>
341     <samlp:Request xmlns:samlp="..." xmlns:saml="..." xmlns:ds="..." >
342       <ds:Signature> ... </ds:Signature>
343       <samlp:AuthenticationQuery>
344         ...
345       </samlp:AuthenticationQuery>
346     </samlp:Request>
```

```
347     </SOAP-ENV:Body>
348 </SOAP-ENV:Envelope>
```

349 Following is an example of the corresponding response, which supplies an assertion containing
350 authentication statement as requested.

```
351 HTTP/1.1 200 OK
352 Content-Type: text/xml
353 Content-Length: nnnn
354
355 <SOAP-ENV:Envelope
356   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" >
357   <SOAP-ENV:Body>
358     <samlp:Response xmlns:samlp="..." xmlns:saml="..." xmlns:ds="..." >
359       <Status>
360         <StatusCode value="samlp:Success" />
361       </Status>
362       <ds:Signature> ... </ds:Signature>
363       <saml:Assertion>
364         <saml:AuthenticationStatement>
365           ...
366         </saml:AuthenticationStatement>
367       </saml:Assertion>
368     </samlp:Response>
369   </SOAP-Env:Body>
370 </SOAP-ENV:Envelope>
```

371 **4 Profiles**

372 The following sections define profiles of SAML that are sanctioned by the OASIS SAML
373 Committee.

374 Two web browser-based profiles that are designed to support single sign-on (SSO), supporting
375 Scenario 1-1 of the SAML requirements document [SAMLReqs]:

- 376 ○ The browser/artifact profile of SAML
- 377 ○ The browser/POST profile of SAML

378

379 For each type of profile, a section describing the threat model and relevant countermeasures is
380 also included.

381 **4.1 Web Browser SSO Profiles of SAML**

382 In the scenario supported by the web browser SSO profiles, a web user authenticates herself to a
383 *source site*. The web user then uses a secured resource at a destination site, without directly
384 authenticating to the *destination site*.

385 The following assumptions are made about this scenario for the purposes of these profiles:

- 386 ● The user is using a standard commercial browser and has authenticated to a source site by
387 some means outside the scope of SAML.
- 388 ● The source site has some form of security engine in place that can track locally
389 authenticated users [WEBSSO]. Typically, this takes the form of a session that might be

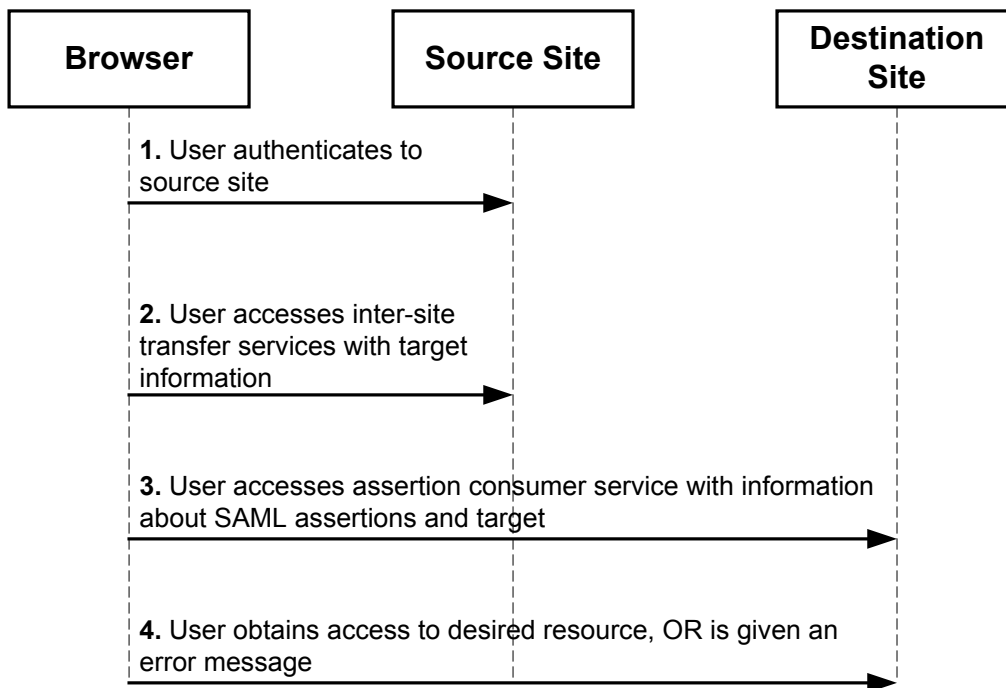
390 represented by an encrypted cookie or an encoded URL or by the use of some other
391 technology [SESSION]. This is a substantial requirement but one that is met by a large
392 class of security engines.

393 At some point, the user attempts to access a *target* resource available from the destination site,
394 and subsequently, through one or more steps (for example, redirection), arrives at an *inter-site*
395 *transfer service* (which may be associated with one or more URIs) at the source site. Starting
396 from this point, the web browser SSO profiles describe a canonical sequence of HTTP exchanges
397 that transfer the user browser to an *assertion consumer service* at the destination site.

398 Information about the SAML assertions provided by the source site and associated with the user,
399 and the desired target, is conveyed from the source to the destination site by the protocol
400 exchange.

401 The assertion consumer service at the destination site can examine both the assertions and the
402 target information and determine whether to allow access to the target resource, thereby
403 achieving web SSO for authenticated users originating from a source site. Often, the destination
404 site also utilizes a security engine that will create and maintain a session, possibly utilizing
405 information contained in the source site assertions, for the user at the destination site.

406 The following figure illustrates this basic template for achieving SSO.



407
408 Two HTTP-based techniques are used in the web browser SSO profiles for conveying
409 information from one site to another via a standard commercial browser.

- 410 • **SAML artifact:** A SAML artifact of “small” bounded size is carried as part of a URL query
411 string such that, when the artifact is conveyed to the source site, the artifact unambiguously
412 references an assertion. The artifact is conveyed via redirection to the destination site, which
413 then acquires the referenced assertion by some further steps. Typically, this involves the use
414 of a registered SAML protocol binding. This technique is used in the browser/artifact profile
415 of SAML.

416 • **Form POST:** SAML assertions are uploaded to the browser within an HTML form and
417 conveyed to the destination site as part of an HTTP POST payload when the user submits the
418 form. This technique is used in the browser/POST profile of SAML.

419 Cookies are not employed in any profile, as cookies impose the limitation that both the source
420 and destination site belong to the same "cookie domain."

421 In the discussion of the web browser SSO profiles, the term *SSO assertion* will be used to refer
422 to an assertion that has (1) a <saml:Conditions> element with `NotBefore` and `NotOnOrAfter`
423 attributes present, and (2) contains one or more authentication statements.

424 **4.1.1 Browser/Artifact Profile of SAML**

425 **4.1.1.1 Required Information**

426 Identification:

427 <urn:oasis:names:tc:SAML:1.0:draft-sstc-bindings-model-132:/profiles:/artifact-01>

428 Contact information:

429 security-services-comment@lists.oasis-open.org

430 Description: Given below.

431 Updates: None.

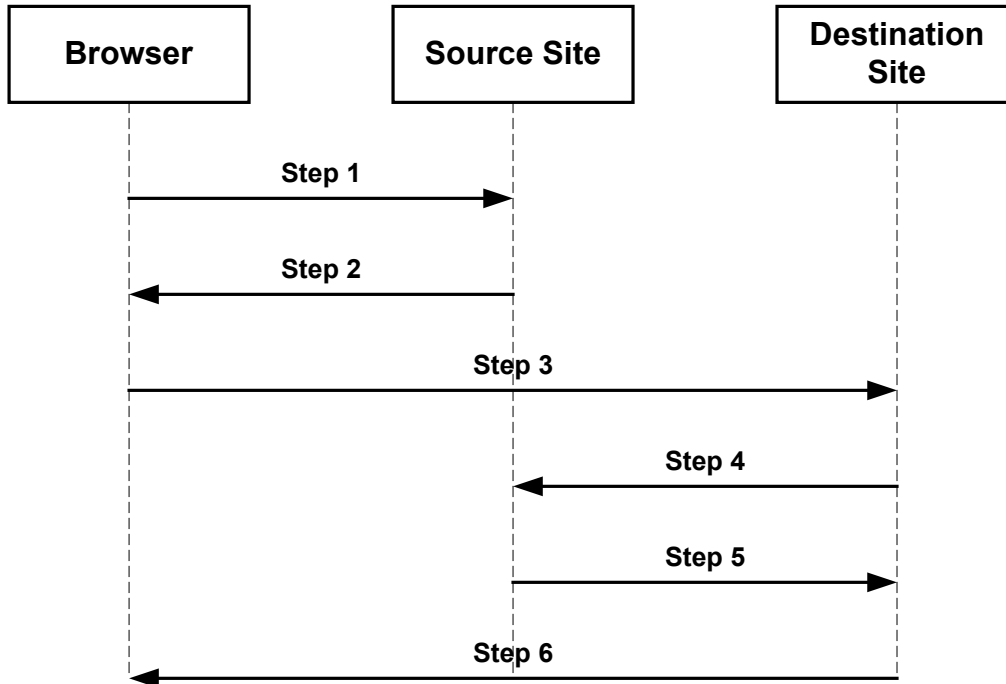
432 **4.1.1.2 Preliminaries**

433 The browser/artifact profile of SAML relies on a reference to the needed assertion traveling in a
434 SAML artifact, which the destination site must dereference from the source site in order to
435 determine whether the user is authenticated.

436 **Note:** The need for a “small” SAML artifact is motivated by restrictions on
437 URL size imposed by commercial web browsers. While RFC 2616
438 [RFC2616] does not specify any restrictions on URL length, in practice
439 commercial web browsers and application servers impose size constraints on
440 URLs, for a maximum size of approximately 2000 characters (see Section
441 79). Further, as developers will need to estimate and set aside URL “real
442 estate” for the artifact, it is important that the artifact have a bounded size,
443 that is, with predefined maximum size. These measures ensure that the
444 artifact can be reliably carried as part of the URL query string and thereby
445 transferred successfully from source to destination site.

446 The browser/artifact profile consists of a single interaction among three parties (a user equipped
447 with a browser, a source site, and a destination site), with a nested sub-interaction between two
448 parties (the source site and the destination site). The interaction sequence is shown in the
449 following figure, with the following sections elucidating each step.

450



451

452 Terminology from RFC 1738 [RFC1738] is used to describe components of a URL. An HTTP
 453 URL has the following form:

454 `http://<HOST>:<port>/<path>?<searchpart>`

455 The following sections specify certain portions of the <searchpart> component of the URL.
 456 Ellipses will be used to indicate additional but unspecified portions of the <searchpart>
 457 component.

458 HTTP requests and responses MUST be drawn from either HTTP 1.1 [RFC2616] or HTTP 1.0
 459 [RFC1945]. Distinctions between the two are drawn only when necessary.

460 **4.1.1.3 Step 1: Accessing the Inter-Site Transfer Service**

461 In step 1, the user’s browser accesses the inter-site transfer service, with information about the
 462 desired target at the destination site attached to the URL.

463 No normative form is given for step 1. It is RECOMMENDED that the HTTP request take the
 464 following form:

465 `GET http://<inter-site transfer host name and path>?TARGET=<Target>...<HTTP-Version>`
 466 `<other HTTP 1.0 or 1.1 components>`

467 Where:

468 <inter-site transfer host name and path>

469 This provides the host name, port number, and path components of an inter-site transfer URL
 470 at the source site.

471 Target=<Target>

472 This name-value pair occurs in the <searchpart> and is used to convey information about
 473 the desired target resource at the destination site.

474 Confidentiality and message integrity MUST be maintained in step 1.

475 **4.1.1.4 Step 2: Redirecting to the Destination Site**

476 In step 2, the source site's inter-site transfer service responds and redirects the user's browser to
477 the assertion consumer service at the destination site.

478 The HTTP response MUST take the following form:

```
479 <HTTP-Version> 302 <Reason Phrase>  
480 <other headers>  
481 Location : http://<assertion consumer host name and path>?<SAML searchpart>  
482 <other HTTP 1.0 or 1.1 components>
```

483 Where:

484 <assertion consumer host name and path>

485 This provides the host name, port number, and path components of an assertion consumer
486 URL at the destination site.

487 <SAML searchpart>= ...TARGET=<Target>...SAMLart=<SAML artifact> ...

488 A single target description MUST be included in the <SAML searchpart> component. At
489 least one SAML artifact MUST be included in the SAML <SAML searchpart> component;
490 multiple SAML artifacts MAY be included. If more than one artifact is carried within <SAML
491 searchpart>, all the artifacts MUST have the same SourceID.

492 According to HTTP 1.1 [RFC2616] and HTTP 1.0 [RFC1945], the use of status code 302 is
493 recommended to indicate that "the requested resource resides temporarily under a different
494 URI". The response may also include additional headers and an optional message body as
495 described in those RFCs.

496 Confidentiality and message integrity MUST be maintained in step 2. It is RECOMMENDED
497 that the inter-site transfer URL be exposed over SSL 3.0 or TLS 1.0 (see Section 50). Otherwise,
498 the one or more artifacts returned in step 2 will be available in plain text to an attacker who
499 might then be able to impersonate the assertion subject.

500 **4.1.1.5 Step 3: Accessing the Assertion Consumer Service**

501 In step 3, the user's browser accesses the assertion consumer service, with a SAML artifact
502 representing the user's authentication information attached to the URL.

503 The HTTP request MUST take the form:

```
504 GET http://<assertion consumer host name and path>?<SAML searchpart> <HTTP-Version>  
505 <other HTTP 1.0 or 1.1 request components>
```

506 Where:

507 <assertion consumer host name and path>

508 This provides the host name, port number, and path components of an assertion consumer
509 URL at the destination site.

510 <SAML searchpart>= ...TARGET=<Target>...SAMLart=<SAML artifact> ...

511 A single target description MUST be included in the <SAML searchpart> component. At
512 least one SAML artifact MUST be included in the <SAML searchpart> component; multiple
513 SAML artifacts MAY be included. If more than one artifact is carried within <SAML
514 searchpart>, all the artifacts MUST have the same SourceID.

515 Confidentiality and message integrity MUST be maintained in step 3. It is RECOMMENDED
516 that the assertion consumer URL be exposed over SSL 3.0 or TLS 1.0 (see Section 50).

517 Otherwise, the artifacts transmitted in step 3 will be available in plain text to any attacker who
518 might then be able to impersonate the assertion subject.

519 **4.1.1.6 Steps 4 and 5: Acquiring the Corresponding Assertions**

520 In steps 4 and 5, the destination site, in effect, dereferences the one or more SAML artifacts in its
521 possession in order to acquire the SAML authentication assertion that corresponds to each artifact.

522 These steps MUST utilize a SAML protocol binding for a SAML request-response message
523 exchange between the destination and source sites. The destination site functions as a SAML
524 requester and the source site functions as a SAML responder.

525 The destination site MUST send a `<samlp:Request>` message to the source site, requesting
526 assertions by supplying assertion artifacts in the `<samlp:AssertionArtifact>` element.

527 If the source site is able to find or construct the requested assertions, it responds with a
528 `<samlp:Response>` message with the requested assertions. Otherwise, it returns an appropriate
529 error code, as defined within the selected SAML binding.

530 In the case where the source site returns assertions within `<samlp:Response>`, it MUST return
531 exactly one assertion for each SAML artifact found in the corresponding `<samlp:Request>`
532 element. The case where fewer or greater number of assertions is returned within the
533 `<samlp:Response>` element MUST be treated as an error state by the destination site.

534 The source site MUST implement a “one-time request” property for each SAML artifact. Many
535 simple implementations meet this constraint by an action such as deleting the relevant assertion
536 from persistent storage at the source site after one lookup. If a SAML artifact is presented to the
537 source site again, the source site MUST return the same message as it would if it were queried
538 with an unknown artifact.

539 The selected SAML protocol binding MUST provide confidentiality, message integrity and
540 bilateral authentication. The source site MUST implement the SAML SOAP binding with
541 support for confidentiality, message integrity, and bilateral authentication.

542 The source site MUST return a response with no assertions if it receives a `<samlp:Request>`
543 message from an authenticated destination site X containing an artifact issued by the source site
544 to some other destination site Y , where $X \diamond Y$. One way to implement this feature is to have
545 source sites maintain a list of artifact and destination site pairs.

546 At least one of the SAML assertions returned to the destination site MUST be an *SSO assertion*.

547 Authentication statements MAY be distributed across more than one returned assertion.

548 The `<saml:ConfirmationMethod>` element of each assertion MUST be set to “SAMLArtifact”
549 (see [SAMLCore]).

550 Based on the information obtained in the assertions retrieved by the destination site, the
551 destination site MAY engage in additional SAML message exchanges with the source site.

552 **4.1.1.7 Step 6: Responding to the User’s Request for a Resource**

553 In step 6, the user’s browser is sent an HTTP response that either allows or denies access to the
554 desired resource.

555 No normative form is mandated for the HTTP response. The destination site SHOULD provide
556 some form of helpful error message in the case where access to resources at that site is
557 disallowed.

558 **4.1.1.8 Artifact Format**

559 The artifact format includes a mandatory two-byte artifact type code, as follows:

```
560 SAML_artifact      := B64 (TypeCode RemainingArtifact)  
561 TypeCode          := Byte1Byte2
```

562 **Note:** Depending on the level of security desired and associated profile
563 protocol steps, many viable architectures could be developed for the SAML
564 artifact [**CoreAssnEx**] [**ShibMarlena**]. The type code structure
565 accommodates variability in the architecture.

566 The notation `B64 (TypeCode RemainingArtifact)` stands for the application of the base-64
567 [**RFC2045**] transformation to the catenation of the `TypeCode` and `RemainingArtifact`. This
568 profile defines an artifact type of type code `0x0001`, which is REQUIRED (mandatory to
569 implement) for any implementation of the browser/artifact profile. This artifact type is defined as
570 follows:

```
571 TypeCode           := 0x0001  
572 RemainingArtifact := SourceID AssertionHandle  
573 SourceID           := 20-byte_sequence  
574 AssertionHandle    := 20-byte_sequence
```

575 `SourceID` is a 20-byte sequence used by the destination site to determine source site identity and
576 location. It is assumed that the destination site will maintain a table of `SourceID` values as well
577 as the URL (or address) for the corresponding SAML responder. This information is
578 communicated between the source and destination sites out-of-band. On receiving the SAML
579 artifact, the destination site determines if the `SourceID` belongs to a known source site and
580 obtains the site location before sending a SAML request (as described in Section [4.1.1.60](#)).

581 Any two source sites with a common destination site MUST use distinct `SourceID` values.
582 Construction of `AssertionHandle` values is governed by the principle that they SHOULD have
583 no predictable relationship to the contents of the referenced assertion at the source site and it
584 MUST be infeasible to construct or guess the value of a valid, outstanding assertion handle.

585 The following practices are RECOMMENDED for the creation of SAML artifacts at source
586 sites:

- 587 • Each source site selects a single identification URL. The domain name used within this
588 URL is registered with an appropriate authority and administered by the source site.
- 589 • The source site constructs the `SourceID` component of the artifact by taking the SHA-1
590 hash of the identification URL.
- 591 • The `AssertionHandle` value is constructed from a cryptographically strong random or
592 pseudorandom number sequence [**RFC1750**] generated by the source site. The sequence
593 consists of values of at least eight bytes in size. These values should be padded to a total
594 length of 20 bytes.

595 **4.1.1.9 Threat Model and Countermeasures**

596 This section utilizes materials from [ShibMarlena] and [Rescorla-Sec].

597 **4.1.1.9.1 Stolen Artifact**

598 **Threat:** If an eavesdropper can copy the real user's SAML artifact, then the eavesdropper could
599 construct a URL with the real user's SAML artifact and be able to impersonate the user at the
600 destination site.

601 **Countermeasure:** As indicated in steps 2, 3, 4, and 5, confidentiality **MUST** be provided
602 whenever an artifact is communicated between a site and the user's browser. This provides
603 protection against an eavesdropper gaining access to a real user's SAML artifact.

604 If an eavesdropper defeats the measures used to ensure confidentiality, additional
605 countermeasures are available:

- 606 • The source and destination sites **SHOULD** make some reasonable effort to ensure that
607 clock settings at both sites differ by at most a few minutes. Many forms of time
608 synchronization service are available, both over the Internet and from proprietary
609 sources.
- 610 • SAML assertions communicated in step 5 **MUST** include an SSO assertion.
- 611 • The source site **SHOULD** track the time difference between when a SAML artifact is
612 generated and placed on a URL line and when a `<samlp:Request>` message carrying the
613 artifact is received from the destination. A maximum time limit of a few minutes is
614 recommended. Should an assertion be requested by a destination site query beyond this
615 time limit, a SAML error **SHOULD** be returned by the source site.
- 616 • It is possible for the source site to create SSO assertions either when the corresponding
617 SAML artifact is created or when a `<samlp:Request>` message carrying the artifact is
618 received from the destination. The validity period of the assertion **SHOULD** be set
619 appropriately in each case: longer for the former, shorter for the latter.
- 620 • Values for `NotBefore` and `NotOnOrAfter` attributes of SSO assertions **SHOULD** have
621 the shortest possible validity period consistent with successful communication of the
622 assertion from source to destination site. This is typically on the order of a few minutes.
623 This ensures that a stolen artifact can only be used successfully within a small time
624 window.
- 625 • The destination site **MUST** check the validity period of all assertions obtained from the
626 source site and reject expired assertions. A destination site **MAY** choose to implement a
627 stricter test of validity for SSO assertions, such as requiring the assertion's
628 `IssueInstant` or `AuthenticationInstant` attribute value to be within a few minutes of
629 the time at which the assertion is received at the destination site.
- 630 • If a received authentication statement includes a `<saml:AuthenticationLocality>`
631 element with the IP address of the user, the destination site **MAY** check the browser IP
632 address against the IP address contained in the authentication statement.

633 **4.1.1.9.2 Attacks on the SAML Protocol Message Exchange**

634 **Threat:** The message exchange in steps 4 and 5 could be attacked in a variety of ways, including
635 artifact or assertion theft, replay, message insertion or modification, and MITM (man-in-the-
636 middle attack).

637 **Countermeasure:** The requirement for the use of a SAML protocol binding with the properties
638 of bilateral authentication, message integrity, and confidentiality defends against these attacks.

639 **4.1.1.9.3 Malicious Destination Site**

640 **Threat:** Since the destination site obtains artifacts from the user, a malicious site could
641 impersonate the user at some new destination site. The new destination site would obtain
642 assertions from the source site and believe the malicious site to be the user.

643 **Countermeasure:** The new destination site will need to authenticate itself to the source site so
644 as to obtain the SAML assertions corresponding to the SAML artifacts. There are two cases to
645 consider:

- 646 1. If the new destination site has no relationship with the source site, it will be unable to
647 authenticate and this step will fail.
- 648 2. If the new destination site has an existing relationship with the source site, the source site
649 will determine that assertions are being requested by a site other than that to which the
650 artifacts were originally sent. In such a case, the source site **MUST** not provide the assertions
651 to the new destination site.

652 **4.1.1.9.4 Forged SAML Artifact**

653 **Threat:** A malicious user could forge a SAML artifact.

654 **Countermeasure:** Section [4.1.1.80](#) provides specific recommendations regarding the
655 construction of a SAML artifact such that it is infeasible to guess or construct the value of a
656 current, valid, and outstanding assertion handle. A malicious user could attempt to repeatedly
657 “guess” a valid SAML artifact value (one that corresponds to an existing assertion at a source
658 site), but given the size of the value space, this action would likely require a very large number
659 of failed attempts. A source site **SHOULD** implement measures to ensure that repeated attempts
660 at querying against non-existent artifacts result in an alarm.

661 **4.1.1.9.5 Browser State Exposure**

662 **Threat:** The SAML artifact profile involves “downloading” of SAML artifacts to the web
663 browser from a source site. This information is available as part of the web browser state and is
664 usually stored in persistent storage on the user system in a completely unsecured fashion. The
665 threat here is that the artifact may be “reused” at some later point in time.

666 **Countermeasure:** The “one-use” property of SAML artifacts ensures that they cannot be reused
667 from a browser. Due to the recommended short lifetimes of artifacts and mandatory SSO
668 assertions, it is difficult to steal an artifact and reuse it from some other browser at a later time.

669 **4.1.2 Browser/POST Profile of SAML**

670 **4.1.2.1 Required Information**

671 Identification:

672 <urn:oasis:names:tc:SAML:1.0:draft-sstc-bindings-model-132:profiles:browser-post>

673 Contact information:

674 security-services-comment@lists.oasis-open.org

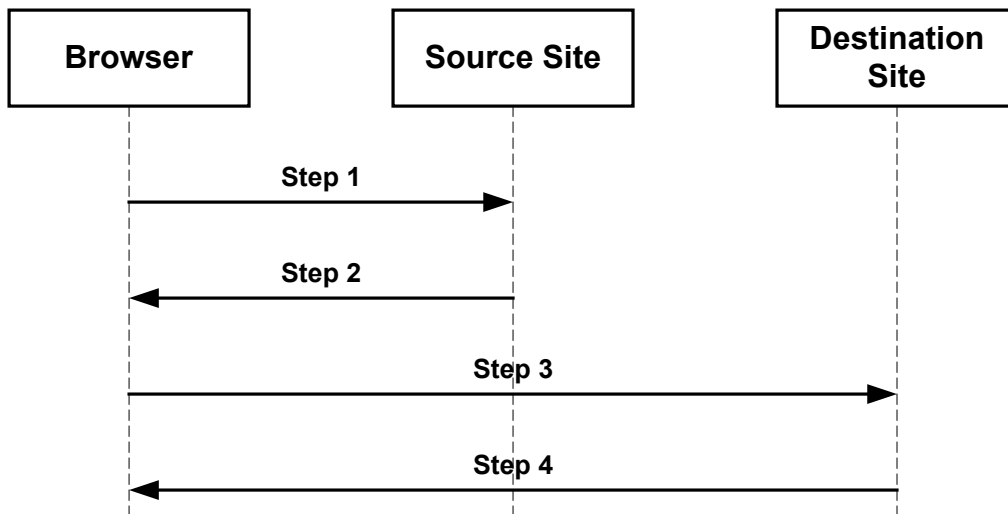
675 Description: Given below.

676 Updates: None.

677 **4.1.2.2 Preliminaries**

678 The browser/POST profile of SAML allows authentication information to be supplied to a
679 destination site without the use of an artifact. The following figure diagrams the interactions
680 between parties in the browser/POST profile.

681 The browser/artifact profile consists of a series of two interactions, the first between a user
682 equipped with a browser and a source site, and the second directly between the user and the
683 destination site. The interaction sequence is shown in the following figure, with the following
684 sections elucidating each step.
685



686

687 **4.1.2.3 Step 1: Accessing the Inter-Site Transfer Service**

688 In step 1, the user's browser accesses the inter-site transfer service, with information about the
689 desired target at the destination site attached to the URL.

690 No normative form is given for step 1. It is RECOMMENDED that the HTTP request take the
691 following form:

692 `GET http://<inter-site transfer host name and path>?TARGET=<Target>...<HTTP-Version>`

693 <other HTTP 1.0 or 1.1 components>

694 Where:

695 <inter-site transfer host name and path>

696 This provides the host name, port number, and path components of an inter-site transfer URL
697 at the source site.

698 Target=<Target>

699 This name-value pair occurs in the <searchpart> and is used to convey information about
700 the desired target resource at the destination site.

701 **4.1.2.4 Step 2: Generating and Supplying the Response**

702 In step 2, the source site generates HTML form data containing a SAML Response which
703 contains an SSO assertion.

704 The HTTP response MUST take the form:

705 <HTTP-Version 200 <Reason Phrase>

706 <other HTTP 1.0 or 1.1 components>

707 Where:

708 <other HTTP 1.0 or 1.1 components>

709 This MUST include an HTML FORM [Chapter 17, HTML 4.01] with the following FORM
710 body:

711 <Body>

712 <FORM Method="Post" Action="<assertion consumer host name and path>" ...>

713 <INPUT TYPE="Submit" NAME="button" Value="Submit">

714 <INPUT TYPE="hidden" NAME="SAMLResponse" Value="B64(<response>)">

715 ...
716 <INPUT TYPE="hidden" NAME="TARGET" Value="<Target>">

717 </Body>

718 <assertion consumer host name and path>

719 This provides the host name, port number, and path components of an assertion consumer
720 URL at the destination site.

721 Exactly one SAML response MUST be included within the FORM body with the control name
722 SAMLResponse; multiple SAML assertions MAY be included in the Response. At least one of the
723 assertions MUST be a SSO assertion. A single target description MUST be included with the
724 control name TARGET.

725 The notation B64 (<response>) stands for the result of applying the base-64 transformation to
726 the response.

727 The SAML response MUST be digitally signed following the guidelines given in [SAMLCore].
728 Included assertions MAY be digitally signed.

729 Confidentiality and message integrity MUST be maintained for step 2. It is RECOMMENDED
730 that the inter-site transfer URL be exposed over SSL 3.0 or TLS 1.0 (see Section 50). Otherwise,
731 the assertions returned will be available in plain text to any attacker who might then be able to
732 impersonate the assertion subject.

733 **4.1.2.5 Step 3: Posting the Form Containing the Response**

734 In step 3, the browser submits the form containing the SAML response using the following
735 HTTP request.

736 The HTTP request MUST include the following components:

```
737 POST http://<assertion consumer host name and path>  
738 <other HTTP 1.0 or 1.1 request components>
```

739 Where:

```
740 <other HTTP 1.0 or 1.1 request components>
```

741
742 This consists of the form data set derived by the browser processing of the form data received in
743 step 2 according to 17.13.3 of [HTML4.01]. Exactly one SAML Response MUST be included
744 within the form data set with control name `SAMLResponse`; multiple SAML assertions MAY be
745 included in the Response. A single target description MUST be included with the control name
746 set to `TARGET`.

747
748 The SAML Response MUST include the Recipient attribute [**SAMLCORE**] with its value set to
749 `<assertion consumer host name and path>`. At least one of the SAML assertions included within
750 the Response MUST be a SSO assertion.

751

752 The destination site MUST ensure a “single use” policy for SSO assertions communicated by
753 means of this profile.

754 **Note:** The implication here is that the destination site will need to save state.
755 A simple implementation might maintain a table of pairs, where each pair
756 consists of the assertion ID and the time at which the entry is to be deleted
757 (where this time is based on the SSO assertion lifetime.). The destination site
758 needs to ensure that there are no duplicate entries. Since SSO assertions
759 containing authentication statements are recommended to have short lifetimes
760 in the web browser context, such a table would be of bounded size.

761 Confidentiality and message integrity MUST be maintained for the HTTP request in step 3. It is
762 RECOMMENDED that the assertion consumer URL be exposed over SSL 3.0 or TLS 1.0 (see
763 Section 59). Otherwise, the assertions transmitted in step 3 will be available in plain text to any
764 attacker who might then impersonate the assertion subject.

765 The `<saml:ConfirmationMethod>` element of each assertion MUST be set to “Assertion
766 Bearer” (See [SAMLCORE]).

767 **Note:** Javascript can be used to avoid an additional “submit” step from the
768 user as follows [**Anders**]:

```
769 <HTML>  
770 <BODY Onload="javascript:document.forms[0].submit ()">  
771 <FORM METHOD="POST" ACTION="destination-site URL">  
772 ...  
773 <INPUT TYPE="HIDDEN" NAME="SAMLResponse"  
774 VALUE=" response in base64 coding">  
775 </FORM>  
776 </BODY>  
777 </HTML>
```

778 **4.1.2.6 Step 4: Responding to the User’s Request for a Resource**

779 In step 4, the user’s browser is sent an HTTP response that either allows or denies access to the
780 desired resource.

781 No normative form is mandated for the HTTP response. The destination site SHOULD provide
782 some form of helpful error message in the case where access to resources at that site is
783 disallowed.

784 **4.1.2.7 Threat Model and Countermeasures**

785 This section utilizes materials from [ShibMarlena] and [Rescorla-Sec].

786 **4.1.2.7.1 Stolen Assertion**

787 **Threat:** If an eavesdropper can copy the real user’s SAML response and included assertions,
788 then the eavesdropper could construct an appropriate POST body and be able to impersonate the
789 user at the destination site.

790 **Countermeasure:** As indicated in steps 2 and 3, confidentiality MUST be provided whenever a
791 response is communicated between a site and the user’s browser. This provides protection
792 against an eavesdropper obtaining a real user’s SAML response and assertions.

793 If an eavesdropper defeats the measures used to ensure confidentiality, additional
794 countermeasures are available:

- 795 • The source and destination sites SHOULD make some reasonable effort to ensure that
796 clock settings at both sites differ by at most a few minutes. Many forms of time
797 synchronization service are available, both over the Internet and from proprietary
798 sources.
- 799 • SAML assertions communicated in step 3 must MUST include an SSO assertion.
- 800 • Values for `NotBefore` and `NotOnOrAfter` attributes of SSO assertions SHOULD have
801 the shortest possible validity period consistent with successful communication of the
802 assertion from source to destination site. This is typically on the order of a few minutes.
803 This ensures that a stolen assertion can only be used successfully within a small time
804 window.
- 805 • The destination site MUST check the validity period of all assertions obtained from the
806 source site and reject expired assertions. A destination site MAY choose to implement a
807 stricter test of validity for SSO assertions, such as requiring the assertion’s
808 `IssueInstant` or `AuthenticationInstant` attribute value to be within a few minutes of
809 the time at which the assertion is received at the destination site.
- 810 • If a received authentication statement includes a `<saml:AuthenticationLocality>`
811 element with the IP address of the user, the destination site MAY check the browser IP
812 address against the IP address contained in the authentication statement.

813 **4.1.2.7.2 MITM Attack**

814 **Threat:** Since the destination site obtains bearer SAML assertions from the user by means of an
815 HTML form, a malicious site could impersonate the user at some new destination site. The new
816 destination site would believe the malicious site to be the subject of the assertion.

817 **Countermeasure:** The destination site MUST check the Recipient attribute of the SAML
818 Response to ensure that its value matches the <assertion consumer host name and path>.
819 As the response is digitally signed, the Recipient value cannot be altered by the malicious site.

820 **4.1.2.7.3 Forged Assertion**

821 **Threat:** A malicious user, or the browser user, could forge or alter a SAML assertion.

822 **Countermeasure:** The browser/POST profile requires the SAML Response carrying SAML
823 assertions to be signed, thus providing both message integrity and authentication. The destination
824 site MUST verify the signature and authenticate the issuer.

825 **4.1.2.7.4 Browser State Exposure**

826 **Threat:** The browser/POST profile involves uploading of assertions from the web browser to a
827 source site. This information is available as part of the web browser state and is usually stored in
828 persistent storage on the user system in a completely unsecured fashion. The threat here is that
829 the assertion may be “reused” at some later point in time.

830 **Countermeasure:** Assertions communicated using this profile must always include an SSO
831 assertion. SSO assertions are expected to have short lifetimes and destination sites are expected
832 to ensure that SSO assertions are not re-submitted.

833 **5 Use of SSL 3.0 or TLS 1.0**

834 In any SAML use of SSL 3.0 or TLS 1.0 [RFC2246], servers MUST authenticate to clients
835 using a X.509.v3 certificate. The client MUST establish server identity based on contents of the
836 certificate (typically through examination of the certificate subject DN field).

837 **5.1 SAML SOAP Binding**

838 TLS-capable implementations MUST implement the
839 TLS_RSA_WITH_3DES_EDE_CBC_SHA ciphersuite and MAY implement the
840 TLS_RSA_AES_128_CBC_SHA ciphersuite [AES].

841 **5.2 Web Browser Profiles of SAML**

842 SSL-capable implementations of the browser/artifact profile or browser/POST profile of SAML
843 MUST implement the SSL_RSA_WITH_3DES_EDE_CBC_SHA ciphersuite.

844 TLS-capable implementations MUST implement the
845 TLS_RSA_WITH_3DES_EDE_CBC_SHA ciphersuite.

6 References

- 846
- 847 **[Anders]** A suggestion on how to implement SAML browser bindings without using
848 “Artifacts”, <http://www.x-obi.com/OBI400/andersr-browser-artifact.ppt>.
- 849 **[AuthXML]** *AuthXML: A Specification for Authentication Information in XML*,
850 [http://www.oasis-open.org/committees/security/docs/draft-authxml-](http://www.oasis-open.org/committees/security/docs/draft-authxml-v2.pdf)
851 [v2.pdf](http://www.oasis-open.org/committees/security/docs/draft-authxml-v2.pdf).
- 852 **[MSURL]** Microsoft technical support article,
853 <http://support.microsoft.com/support/kb/articles/Q208/4/27.ASP>.
- 854 **[RFC2119]** S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
855 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 856 **[RFC2617]** *HTTP Authentication: Basic and Digest Access Authentication*,
857 <http://www.ietf.org/rfc/rfc2617.txt>, IETF RFC 2617.
- 858 **[S2ML]** *S2ML: Security Services Markup Language*, Version 0.8a, January 8,
859 2001. [http://www.oasis-open.org/committees/security/docs/draft-s2ml-](http://www.oasis-open.org/committees/security/docs/draft-s2ml-v08a.pdf)
860 [v08a.pdf](http://www.oasis-open.org/committees/security/docs/draft-s2ml-v08a.pdf).
- 861 **[SAMLCore]** Hallam-Baker, P. et al., *Assertions and Protocol for the OASIS Security*
862 *Assertion Markup Language (SAML)*, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-core-21.pdf)
863 [open.org/committees/security/docs/draft-sstc-core-21.pdf](http://www.oasis-open.org/committees/security/docs/draft-sstc-core-21.pdf), OASIS,
864 December 2001.
- 865 **[SAMLGloss]** J. Hodges et al., *Glossary for the OASIS Security Assertion Markup*
866 *Language (SAML)*, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-glossary-02.pdf)
867 [open.org/committees/security/docs/draft-sstc-glossary-02.pdf](http://www.oasis-open.org/committees/security/docs/draft-sstc-glossary-02.pdf), OASIS,
868 December 2001.
- 869 **[SAMLSec]** J. Hodges et al., *Security Considerations for the OASIS Security Assertion*
870 *Markup Language (SAML)*, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sec-consider-02.pdf)
871 [open.org/committees/security/docs/draft-sec-consider-02.pdf](http://www.oasis-open.org/committees/security/docs/draft-sec-consider-02.pdf), OASIS,
872 December 2001.
- 873 **[SAMLReqs]** D. Platt et al., *SAML Requirements and Use Cases*, OASIS, December
874 2001.
- 875 **[Shib]** Shibboleth Overview and Requirements
876 [http://middleware.internet2.edu/shibboleth/docs/draft-internet2-](http://middleware.internet2.edu/shibboleth/docs/draft-internet2-shibboleth-requirements-00.html)
877 [shibboleth-requirements-](http://middleware.internet2.edu/shibboleth/docs/draft-internet2-shibboleth-requirements-00.html)
878 [00.html](http://middleware.internet2.edu/shibboleth/docs/draft-internet2-shibboleth-requirements-00.html)[http://middleware.internet2.edu/shibboleth/docs/draft-internet2-](http://middleware.internet2.edu/shibboleth/docs/draft-internet2-shibboleth-requirements-00.html)
879 [shibboleth-requirements-00.html](http://middleware.internet2.edu/shibboleth/docs/draft-internet2-shibboleth-requirements-00.html)
- 880 **[ShibMarlena]** Marlena Erdos, *Shibboleth Architecture DRAFT v1.1*,
881 [http://middleware.internet2.edu/shibboleth/docs/draft-erdos-shibboleth-](http://middleware.internet2.edu/shibboleth/docs/draft-erdos-shibboleth-architecture-00.pdf)
882 [architecture-00.pdf](http://middleware.internet2.edu/shibboleth/docs/draft-erdos-shibboleth-architecture-00.pdf)
- 883
- 884 **[RFC2045]** [Multipurpose Internet Mail Extensions \(MIME\) Part One: Format of](#)
885 [Internet Message Bodies](#)
- 886

887 [RFC2616] Hypertext Transfer Protocol -- HTTP/1.1,
888 <http://www.ietf.org/rfc/rfc2616.txt>.
889

890 [RFC1738] Uniform Resource Locators (URL), <http://www.ietf.org/rfc/rfc1738.txt>

891 [RFC1750] Randomness Recommendations for Security.
892 <http://www.ietf.org/rfc/rfc1750.txt>

893 [RFC1945] Hypertext Transfer Protocol -- HTTP/1.0,
894 <http://www.ietf.org/rfc/rfc1945.txt>.

895 [RFC2246] The TLS Protocol Version 1.0, <http://www.ietf.org/rfc/rfc2246.html>.

896 [RFC2774] An HTTP Extension Framework, <http://www.ietf.org/rfc/rfc2774.txt>.

897 [SOAP1.1] D. Box et al., *Simple Object Access Protocol (SOAP) 1.1*,
898 <http://www.w3.org/TR/SOAP>, World Wide Web Consortium Note, May
899 2000.

900 [CoreAssnEx] Core Assertions Architecture, Examples and Explanations,
901 <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-phill-07.pdf>.
902

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

905 [WEBSSO] RL “Bob” Morgan, Interactions between Shibboleth and local-site web
906 sign-on services, [http://middleware.internet2.edu/shibboleth/docs/draft-
907 morgan-shibboleth-websso-00.txt](http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-websso-00.txt)

908 [SESSION] RL “Bob” Morgan, Support of target web server sessions in Shibboleth,
909 [http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-
910 session-00.txt](http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-session-00.txt)

911 [SSLv3] The SSL Protocol Version 3.0,
912 <http://www.mozilla.org/projects/security/pki/nss/ssl/draft302.txt>

913 [Rescorla-Sec] E. Rescorla et al., *Guidelines for Writing RFC Text on Security
914 Considerations*, [http://www.ietf.org/internet-drafts/draft-rescorla-sec-
915 cons-03.txt](http://www.ietf.org/internet-drafts/draft-rescorla-sec-cons-03.txt).

916 **7 URL Size Restriction (Non-Normative)**

917 This section describes the URL size restrictions that have been documented for widely used
918 commercial products.

919 A Microsoft technical support article [MSURL] provides the following information:

920 The information in this article applies to:

921 Microsoft Internet Explorer (Programming) versions 4.0, 4.01, 4.01 SP1, 4.01
922 SP2, 5, 5.01, 5.5

923 SUMMARY

924 Internet Explorer has a maximum uniform resource locator (URL) length of
925 2,083 characters, with a maximum path length of 2,048 characters. This limit
926 applies to both POST and GET request URLs.

927 If you are using the GET method, you are limited to a maximum of 2,048
928 characters (minus the number of characters in the actual path, of course).

929 POST, however, is not limited by the size of the URL for submitting
930 name/value pairs, because they are transferred in the header and not the URL.

931 RFC 2616, Hypertext Transfer Protocol -- HTTP/1.1, does not specify any
932 requirement for URL length.

933 REFERENCES

934 Further breakdown of the components can be found in the Wininet header file.
935 Hypertext Transfer Protocol -- HTTP/1.1 General Syntax, section 3.2.1

936 Additional query words: POST GET URL length

937 Keywords : kbIE kbIE400 kbie401 kbGrpDSInet kbie500 kbDSupport kbie501
938 kbie550 kbieFAQ

939 Issue type : kbinfo

940 Technology :

941 An article about [Netscape Enterprise Server](#) provides the following information:

942 Issue: 19971110-3 Product: Enterprise Server

943 Created: 11/10/1997 Version: 2.01

944 Last Updated: 08/10/1998 OS: AIX, Irix, Solaris

945 Does this article answer your question?

946 Please let us know!

947 Question:

948 How can I determine the maximum URL length that the Enterprise server will
949 accept? Is this configurable and, if so, how?

950 Answer:

951 Any single line in the headers has a limit of 4096 chars; it is not configurable.

952 **8 Alternative SAML Artifact Format**

953 **8.1 Required Information**

954 Identification:

955 <http://www.oasis-open.org/security/>

956 <urn:oasis:names:tc:SAML:1.0:draft-sstc-bindings-model-130.9:profiles:artifact-02>

957 Contact information:
958 security-services-comment@lists.oasis-open.org
959 Description: Given below.
960 Updates: None.

961 **8.2 Format Details**

962 An alternative artifact format is described here:

```
963 TypeCode           := 0x0002  
964 RemainingArtifact := AssertionHandle SourceLocation  
965 AssertionHandle   := 20-byte_sequence  
966 SourceLocation    := URI
```

967 The `SourceLocation` URI is the address of the SAML responder associated with the source site.
968 The `assertionHandle` is as described in Section [10](#), and governed by the same requirements.
969 The destination site **MUST** process the artifact in a manner identical to that described in Section
970 [4.1.10](#), with the exception that the location of the SAML responder at the source site **MAY** be
971 obtained directly from the artifact, rather than by look-up, based on `sourceID`.

972 Note: the destination site **MUST** confirm that assertions were issued by an acceptable issuer, not
973 relying merely on the fact that they were returned in response to a `samlp:request`.

974

975

976 **Appendix A. Notices**

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

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

989 Copyright © The Organization for the Advancement of Structured Information Standards
990 [OASIS] 2001. All Rights Reserved.

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

1000 The limited permissions granted above are perpetual and will not be revoked by OASIS or its
1001 successors or assigns.

1002 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1003 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT
1004 LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN
1005 WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF
1006 MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Page: 28

[elm1] What exactly does this information apply to? Can we cite a URL for it?