



Bindings and Profiles for the OASIS Security Assertion Markup Language (SAML) V1.1

Committee Specification, 18 July 2003

Document identifier:

sstc-saml-bindings-1.1-cs-02

Location:

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

Editors:

Eve Maler, Sun Microsystems (eve.maler@sun.com)
Prateek Mishra, Netegrity (pmishra@netegrity.com)
Robert Philpott, RSA Security (rphilpott@rsasecurity.com)

Contributors:

Irving Reid, Baltimore Technologies
Krishna Sankar, Cisco Systems
John Hughes, Entegriy Solutions
Tim Moses, Entrust
Evan Prodromou, former member
Bob Blakley, IBM
Marlena Erdos, IBM
Scott Cantor, individual
RL "Bob" Morgan, individual
Simon Godik, Overxeer
Jahan Moreh, Sigaba
Chris Ferris, formerly of Sun Microsystems
Jeff Hodges, Sun Microsystems

Abstract:

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

Status:

This document is a **Committee Specification** of the OASIS Security Services Technical Committee. This document is updated periodically on no particular schedule. Send comments to the editors.

Committee members should send comments on this specification to the security-services@lists.oasis-open.org list. Others should subscribe to and send comments to the security-services-comment@lists.oasis-open.org list. 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.

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

44 For information on errata discovered in this specification, please refer to the most recent errata
45 document which can be found in the document repository at the Security Services TC web page
46 (<http://www.oasis-open.org/committees/security/>).

Table of Contents

48	1	Introduction	5
49	1.1	Protocol Binding and Profile Concepts	5
50	1.2	Notation	5
51	2	Specification of Additional Protocol Bindings and Profiles	7
52	2.1	Guidelines for Specifying Protocol Bindings and Profiles	7
53	2.2	Process Framework for Describing and Registering Protocol Bindings and Profiles	7
54	3	Protocol Bindings	8
55	3.1	SAML SOAP Binding	8
56	3.1.1	Required Information	8
57	3.1.2	Protocol-Independent Aspects of the SAML SOAP Binding	8
58	3.1.2.1	Basic Operation	8
59	3.1.2.2	SOAP Headers	9
60	3.1.2.3	Authentication	9
61	3.1.2.4	Message Integrity	9
62	3.1.2.5	Confidentiality	9
63	3.1.3	Use of SOAP over HTTP	9
64	3.1.3.1	HTTP Headers	10
65	3.1.3.2	Authentication	10
66	3.1.3.3	Message Integrity	10
67	3.1.3.4	Message Confidentiality	10
68	3.1.3.5	Security Considerations	10
69	3.1.3.6	Error Reporting	10
70	3.1.3.7	Example SAML Message Exchange Using SOAP over HTTP	11
71	4	Profiles	12
72	4.1	Web Browser SSO Profiles of SAML	12
73	4.1.1	Browser/Artifact Profile of SAML	13
74	4.1.1.1	Required Information	13
75	4.1.1.2	Preliminaries	14
76	4.1.1.3	Step 1: Accessing the Inter-Site Transfer Service	15
77	4.1.1.4	Step 2: Redirecting to the Destination Site	15
78	4.1.1.5	Step 3: Accessing the Artifact Receiver URL	16
79	4.1.1.6	Steps 4 and 5: Acquiring the Corresponding Assertions	16
80	4.1.1.7	Step 6: Responding to the User's Request for a Resource	17
81	4.1.1.8	Artifact Format	17
82	4.1.1.9	Threat Model and Countermeasures	18
83	4.1.1.9.1	Stolen Artifact	18
84	4.1.1.9.2	Attacks on the SAML Protocol Message Exchange	18
85	4.1.1.9.3	Malicious Destination Site	19
86	4.1.1.9.4	Forged SAML Artifact	19
87	4.1.1.9.5	Browser State Exposure	19
88	4.1.2	Browser/POST Profile of SAML	19
89	4.1.2.1	Required Information	19
90	4.1.2.2	Preliminaries	20

91	4.1.2.3 Step 1: Accessing the Inter-Site Transfer Service.....	20
92	4.1.2.4 Step 2: Generating and Supplying the Response	20
93	4.1.2.5 Step 3: Posting the Form Containing the Response	21
94	4.1.2.6 Step 4: Responding to the User's Request for a Resource.....	22
95	4.1.2.7 Threat Model and Countermeasures	22
96	4.1.2.7.1 Stolen Assertion	23
97	4.1.2.7.2 MITM Attack.....	23
98	4.1.2.7.3 Forged Assertion.....	23
99	4.1.2.7.4 Browser State Exposure.....	23
100	5 Confirmation Method Identifiers	24
101	5.1 Holder of Key	24
102	5.2 Sender Vouches	24
103	5.3 SAML Artifact.....	24
104	5.4 Bearer	24
105	6 Use of SSL 3.0 or TLS 1.0	25
106	6.1 SAML SOAP Binding	25
107	6.2 Web Browser Profiles of SAML	25
108	7 Alternative SAML Artifact Format	26
109	7.1 Required Information	26
110	7.2 Format Details	26
111	8 URL Size Restriction (Non-Normative).....	27
112	9 References	28
113	Appendix A. Acknowledgments	30
114	Appendix B. Notices.....	31

1 Introduction

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

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

1.1 Protocol Binding and Profile Concepts

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

125 For example, a SAML SOAP binding describes how SAML request and response message exchanges
126 are mapped into SOAP message exchanges.

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

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

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

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

1.2 Notation

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

143 `Listings of productions or other normative code appear like this.`

144 `Example code listings appear like this.`

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

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

- 149 • The prefix `saml`: stands for the SAML assertion namespace [**SAMLCore**].
- 150 • The prefix `samlp`: stands for the SAML request-response protocol namespace [**SAMLCore**].
- 151 • The prefix `ds`: stands for the W3C XML Signature namespace,
152 `http://www.w3.org/2000/09/xmldsig#` [**XMLSig**].
- 153 • The prefix `SOAP-ENV`: stands for the SOAP 1.1 namespace,
154 `http://schemas.xmlsoap.org/soap/envelope` [**SOAP1.1**].

155 This specification uses the following typographical conventions in text: <SAML**E**lement>,
156 <ns:ForeignElement>, Attribute, **Datatype**, OtherCode. In some cases, angle brackets are used
157 to indicate non-terminals, rather than XML elements; the intent will be clear from the context.

2 Specification of Additional Protocol Bindings and Profiles

158
159
160
161
162
163
164
165

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

2.1 Guidelines for Specifying Protocol Bindings and Profiles

166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183

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

1. Describe the set of interactions between parties involved in the binding or profile. Any restrictions on applications used by each party and the protocols involved in each interaction must be explicitly called out.
2. Identify the parties involved in each interaction, including how many parties are involved and whether intermediaries may be involved.
3. Specify the method of authentication of parties involved in each interaction, including whether authentication is required and acceptable authentication types.
4. Identify the level of support for message integrity, including the mechanisms used to ensure message integrity.
5. Identify the level of support for confidentiality, including whether a third party may view the contents of SAML messages and assertions, whether the binding or profile requires confidentiality, and the mechanisms recommended for achieving confidentiality.
6. Identify the error states, including the error states at each participant, especially those that receive and process SAML assertions or messages.
7. Identify security considerations, including analysis of threats and description of countermeasures.
8. Identify SAML confirmation method identifiers defined and/or utilized by the binding or profile.

2.2 Process Framework for Describing and Registering Protocol Bindings and Profiles

184
185
186
187
188
189
190
191
192
193
194
195
196
197
198

For any new protocol binding or profile to be interoperable, it needs to be openly specified. The OASIS Security Services Technical Committee will maintain a registry and repository of submitted bindings and profiles titled "Additional Bindings and Profiles" at the SAML website [**SAMLWeb**] in order to keep the SAML community informed. The committee will also provide instructions for submission of bindings and profiles by OASIS members.

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

1. Identification: Specify a URI that uniquely identifies this protocol binding or profile.
2. Contact information: Specify the postal or electronic contact information for the author of the protocol binding or profile.
3. Description: Provide a text description of the protocol binding or profile. The description **SHOULD** follow the guidelines described in Section 2.1.
4. Updates: Provide references to previously registered protocol bindings or profiles that the current entry improves or obsoletes.

199 3 Protocol Bindings

200 The following sections define SAML protocol bindings sanctioned by the OASIS Security Services
201 Technical Committee. Only one binding, the SAML SOAP binding, is currently defined.

202 3.1 SAML SOAP Binding

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

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

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

212 3.1.1 Required Information

213 **Identification:** urn:oasis:names:tc:SAML:1.0:bindings:SOAP-binding

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

215 **Description:** Given below.

216 **Updates:** None.

217 3.1.2 Protocol-Independent Aspects of the SAML SOAP Binding

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

220 3.1.2.1 Basic Operation

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

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

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

227 1. A system entity acting as a SAML requester transmits a SAML `<Request>` element within the body
228 of a SOAP message to a system entity acting as a SAML responder. The SAML requester MUST
229 NOT include more than one SAML request per SOAP message or include any additional XML
230 elements in the SOAP body.

231 2. The SAML responder MUST return either a `<Response>` element within the body of another SOAP
232 message or a SOAP fault code. The SAML responder MUST NOT include more than one SAML
233 response per SOAP message or include any additional XML elements in the SOAP body. If a SAML
234 responder cannot, for some reason, process a SAML request, it MUST return a SOAP fault code.
235 SOAP fault codes MUST NOT be sent for errors within the SAML problem domain, for example,
236 inability to find an extension schema or as a signal that the subject is not authorized to access a
237 resource in an authorization query. (SOAP 1.1 faults and fault codes are discussed in **[SOAP1.1]**
238 §4.1.)

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

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

247 3.1.2.2 SOAP Headers

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

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

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

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

257 3.1.2.3 Authentication

258 Authentication of both the SAML requester and the SAML responder is OPTIONAL and depends on the
259 environment of use. Authentication protocols available from the underlying substrate protocol MAY be
260 utilized to provide authentication. Section 3.1.3.2 describes authentication in the SOAP over HTTP
261 environment.

262 3.1.2.4 Message Integrity

263 Message integrity of both SAML requests and SAML responses is OPTIONAL and depends on the
264 environment of use. The security layer in the underlying substrate protocol MAY be used to ensure
265 message integrity. Section 3.1.3.3 describes support for message integrity in the SOAP over HTTP
266 environment.

267 3.1.2.5 Confidentiality

268 Confidentiality of both SAML requests and SAML responses is OPTIONAL and depends on the
269 environment of use. The security layer in the underlying substrate protocol MAY be used to ensure
270 message confidentiality. Section 3.1.3.4 describes support for confidentiality in the SOAP over HTTP
271 environment.

272 3.1.3 Use of SOAP over HTTP

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

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

279 `http://www.oasis-open.org/committees/security`

280 3.1.3.1 HTTP Headers

281 HTTP proxies MUST NOT cache responses carrying SAML assertions.

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

- 283 • If the value of the `Cache-Control` header field is **not** set to `no-store`, then the SAML responder
284 MUST NOT include the `Cache-Control` header field in the response.
- 285 • If the `Expires` response header field is **not** disabled by a `Cache-Control` header field with a value
286 of `no-store`, then the `Expires` field SHOULD NOT be included.

287 There are no other restrictions on HTTP headers.

288 3.1.3.2 Authentication

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

- 290 1. No client or server authentication.
- 291 2. HTTP basic client authentication [**RFC2617**] with and without SSL 3.0 or TLS 1.0.
- 292 3. HTTP over SSL 3.0 or TLS 1.0 (see Section 6) server authentication with a server-side certificate.
- 293 4. HTTP over SSL 3.0 or TLS 1.0 mutual authentication with both server-side and a client-side
294 certificate.

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

296 3.1.3.3 Message Integrity

297 When message integrity needs to be guaranteed, SAML responders MUST use HTTP over SSL 3.0 or
298 TLS 1.0 (see Section 6) with a server-side certificate.

299 3.1.3.4 Message Confidentiality

300 When message confidentiality is required, SAML responders MUST use HTTP over SSL 3.0 or TLS 1.0
301 (see Section 6) with a server-side certificate.

302 3.1.3.5 Security Considerations

303 Before deployment, each combination of authentication, message integrity, and confidentiality
304 mechanisms SHOULD be analyzed for vulnerability in the context of the deployment environment. See
305 the SAML security considerations document [**SAMLSec**] for a detailed discussion.

306 RFC 2617 [**RFC2617**] describes possible attacks in the HTTP environment when basic or message-
307 digest authentication schemes are used.

308 3.1.3.6 Error Reporting

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

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

317 In the case of a SAML processing error, the SOAP HTTP server MUST respond with "200 OK" and
318 include a SAML-specified `<Status>` element using one of the following mechanisms:

- 319 • As the only child of the <SOAP-ENV:Body> element. This mechanism is deprecated in SAML v1.1
320 and will be removed in the next major revision of SAML.
- 321 • As the only child of a SAML <Response> element within the SOAP body (RECOMMENDED).
- 322 For more information about SAML status codes, see the SAML assertion and protocol specification
323 [SAMLCore].

324 3.1.3.7 Example SAML Message Exchange Using SOAP over HTTP

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

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

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

```
345 HTTP/1.1 200 OK
346 Content-Type: text/xml
347 Content-Length: nnnn
348
349 <SOAP-ENV:Envelope
350   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
351   <SOAP-ENV:Body>
352     <samlp:Response xmlns:samlp="..." xmlns:saml="..." xmlns:ds="...">
353       <Status>
354         <StatusCodevalue="samlp:Success"/>
355       </Status>
356       <ds:Signature> ... </ds:Signature>
357       <saml:Assertion>
358         <saml:AuthenticationStatement>
359           ...
360         </saml:AuthenticationStatement>
361       </saml:Assertion>
362     </samlp:Response>
363   </SOAP-Env:Body>
364 </SOAP-ENV:Envelope>
```

365

4 Profiles

366 The following sections define profiles of SAML that are sanctioned by the OASIS Security Services
367 Technical Committee.

368 Two web browser-based profiles are defined to support single sign-on (SSO), supporting Scenario 1-1 of
369 the SAML requirements document [**SAMLReqs**]:

- 370 • The browser/artifact profile of SAML
- 371 • The browser/POST profile of SAML

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

374 Some additional profiles that have been published outside the Security Services Technical Committee
375 are:

- 376 • The OASIS Web Services Security Technical Committee has produced a draft “SAML token profile”
377 of the WSS specification [**WSS-SAML**], which describes how to use SAML assertions to secure a
378 web service message.
- 379 • The Liberty Alliance Project [**Liberty**] has produced a set of profiles for its extended version of SAML.

380 4.1 Web Browser SSO Profiles of SAML

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

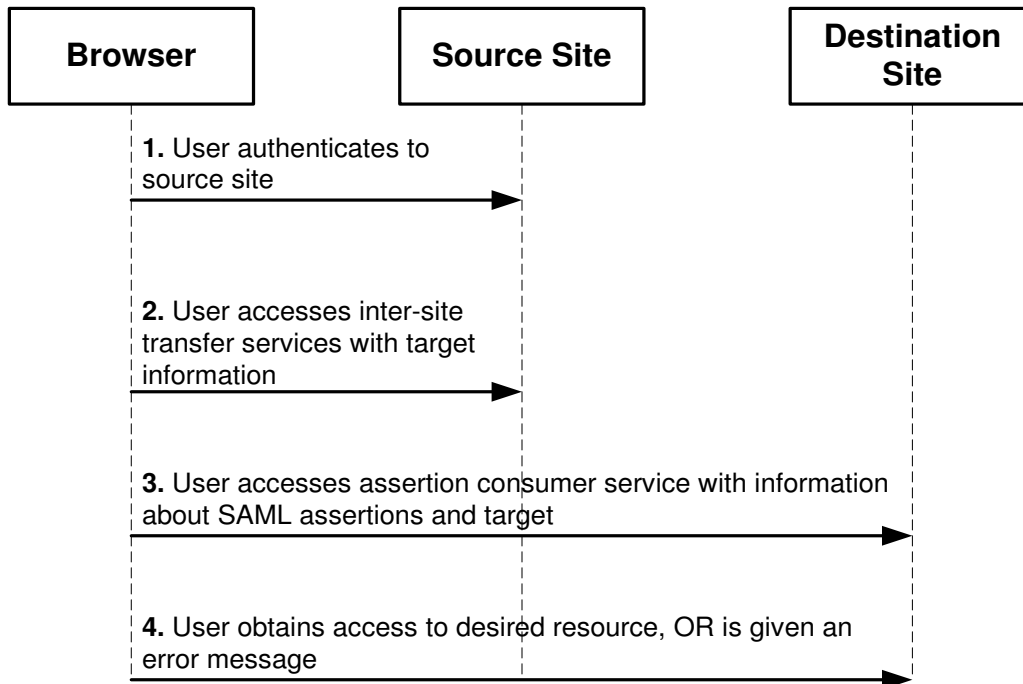
384 The following assumptions are made about this scenario for the purposes of these profiles:

- 385 • The user is using a standard commercial browser and has authenticated to a source site by some
386 means outside the scope of SAML.
- 387 • The source site has some form of security engine in place that can track locally authenticated users
388 [**WEBSSO**]. Typically, this takes the form of a session that might be represented by an encrypted
389 cookie or an encoded URL or by the use of some other technology [**SESSION**]. This is a substantial
390 requirement but one that is met by a large class of security engines.

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

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

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



404

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

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

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

416 Cookies are not employed in these profiles, as cookies impose the limitation that both the source and
 417 destination site belong to the same "cookie domain."

418 In the discussion of the web browser SSO profiles, the term *SSO assertion* will be used to refer to an
 419 assertion that has a `<saml:Conditions>` element with `NotBefore` and `NotOnOrAfter` attributes
 420 present, and also contains at least one or more authentication statements about the subject. Note that an
 421 SSO assertion MAY also include additional information about the subject, such as attributes.

422 4.1.1 Browser/Artifact Profile of SAML

423 4.1.1.1 Required Information

424 **Identification:** urn:oasis:names:tc:SAML:1.0:profiles:artifact-01

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

426 **SAML Confirmation Method Identifiers:** The "SAML artifact" confirmation method identifier is used by
 427 this profile. The following RECOMMENDED identifier has been assigned to this confirmation method:

428 urn:oasis:names:tc:SAML:1.0:cm:artifact

429 The following identifier is deprecated and is planned to be removed in the next major revision of SAML:

430 urn:oasis:names:tc:SAML:1.0:cm:artifact-01

431 **Description:** Given below.

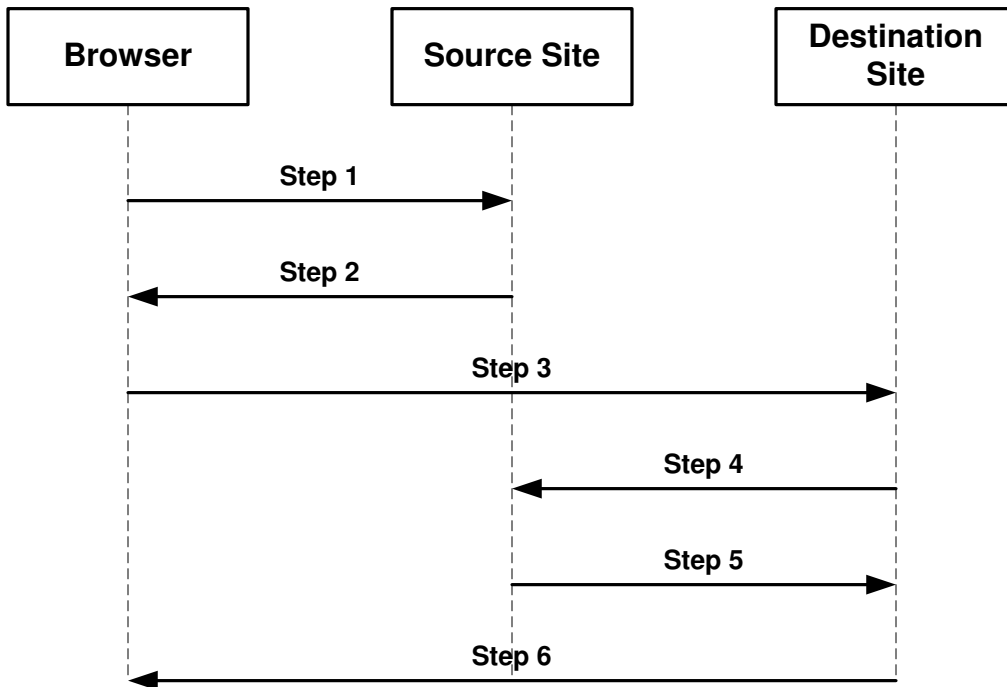
432 **Updates:** None.

433 4.1.1.2 Preliminaries

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

437 **Note:** The need for a “small” SAML artifact is motivated by restrictions on URL size
438 imposed by commercial web browsers. While RFC 2616 [RFC2616] does not specify any
439 restrictions on URL length, in practice commercial web browsers and application servers
440 impose size constraints on URLs, for a maximum size of approximately 2000 characters
441 (see Section 8). 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, that is, with
443 predefined maximum size. These measures ensure that the artifact can be reliably
444 carried as part of the URL query string and thereby transferred successfully from source
445 to destination site.

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



450
451 Terminology from RFC 1738 [RFC1738] is used to describe components of a URL. An HTTP URL has
452 the following form:

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

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

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

4.1.1.3 Step 1: Accessing the Inter-Site Transfer Service

In step 1, the user's browser accesses the inter-site transfer service at host <https://<inter-site transfer host name>>, with information about the desired target at the destination site attached to the URL.

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

```
GET <path>?...TARGET=<Target>...<HTTP-Version>
<other HTTP 1.0 or 1.1 components>
```

Where:

<inter-site transfer host name>

This provides the host name and optional port number at the source site where an inter-site transfer service is available.

<path>

This provides the path components of an inter-site transfer service URL at the source site.

Target=<Target>

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

Confidentiality and message integrity MUST be maintained in step 1.

4.1.1.4 Step 2: Redirecting to the Destination Site

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

Note: In the browser/artifact profile, the URL used by the source site to access the assertion consumer service at the destination site is referred to as the *artifact receiver URL*.

The HTTP response MUST take the following form:

```
<HTTP-Version> 302 <Reason Phrase>
<other headers>
Location : https://<artifact receiver host name and path>?<SAML searchpart>
<other HTTP 1.0 or 1.1 components>
```

Where:

<artifact receiver host name and path>

This provides the host name, port number, and path components of an artifact receiver URL associated with the assertion consumer service at the destination site.

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

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

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

Confidentiality and message integrity MUST be maintained in step 2. It is RECOMMENDED that the inter-site transfer URL be protected by SSL 3.0 or TLS 1.0 (see Section 6). Otherwise, the one or more artifacts returned in step 2 will be available in plain text to an attacker who might then be able to impersonate the subject.

503 **4.1.1.5 Step 3: Accessing the Artifact Receiver URL**

504 In step 3, the user's browser accesses the artifact receiver service at host <https://<artifact receiver host name>>, with a SAML artifact representing the user's authentication information attached to the URL.

506 The HTTP request MUST take the form:

```
507 GET <path>?...<SAML searchpart>...<HTTP-Version>  
508 <other HTTP 1.0 or 1.1 request components>
```

509 Where:

510 <artifact receiver host name>

511 This provides the host name and optional port number at the destination site where the artifact receiver service URL associated with the assertion consumer service is available.

513 <path>

514 This provides the path components of the artifact receiver service URL at the destination site.

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

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

520 Confidentiality and message integrity MUST be maintained in step 3. It is RECOMMENDED that the artifact receiver URL be protected by SSL 3.0 or TLS 1.0 (see Section 6). Otherwise, the artifacts transmitted in step 3 will be available in plain text to any attacker who might then be able to impersonate the assertion subject.

524 **4.1.1.6 Steps 4 and 5: Acquiring the Corresponding Assertions**

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

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

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

532 If the source site is able to find or construct the requested assertions, it responds with a <samlp:Response> message with the requested assertions. Otherwise, it responds with a <samlp:Response> message with no assertions. The <samlp:Status> element of the <samlp:Response> MUST include a <samlp:StatusCode> element with the value Success.

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

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

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

547 The source site MUST return a response with no assertions if it receives a <samlp:Request> message from an authenticated destination site X containing an artifact issued by the source site to some other

549 destination site *Y*, where $X \leftrightarrow Y$. One way to implement this feature is to have source sites maintain a list
550 of artifact and destination site pairs. The `<samlp:Status>` element of the `<samlp:Response>` MUST
551 include a `<samlp:StatusCode>` element with the value `Success`.

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

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

554 Every subject-based statement in the assertion(s) returned to the destination site MUST contain a
555 `<saml:SubjectConfirmation>` element as follows:

- 556 • The `<saml:ConfirmationMethod>` element MUST be set to either
557 `urn:oasis:names:tc:SAML:1.0:cm:artifact-01` (deprecated) or `urn:oasis:names:tc:SAML:1.0:cm:artifact`
558 (RECOMMENDED).
- 559 • The `<SubjectConfirmationData>` element SHOULD NOT be specified.

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

562 4.1.1.7 Step 6: Responding to the User's Request for a Resource

563 In step 6, the user's browser is sent an HTTP response that either allows or denies access to the desired
564 resource.

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

567 4.1.1.8 Artifact Format

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

```
569 SAML_artifact      := B64(TypeCode RemainingArtifact)  
570 TypeCode          := Byte1Byte2
```

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

574 The notation `B64(TypeCode RemainingArtifact)` stands for the application of the base64
575 **[RFC2045]** transformation to the catenation of the `TypeCode` and `RemainingArtifact`. This profile
576 defines an artifact type of type code `0x0001`, which is REQUIRED (mandatory to implement) for any
577 implementation of the browser/artifact profile. This artifact type is defined as follows:

```
578 TypeCode           := 0x0001  
579 RemainingArtifact := SourceID AssertionHandle  
580 SourceID          := 20-byte_sequence  
581 AssertionHandle   := 20-byte_sequence
```

582 `SourceID` is a 20-byte sequence used by the destination site to determine source site identity and
583 location. It is assumed that the destination site will maintain a table of `SourceID` values as well as the
584 URL (or address) for the corresponding SAML responder. This information is communicated between the
585 source and destination sites out-of-band. On receiving the SAML artifact, the destination site determines
586 if the `SourceID` belongs to a known source site and obtains the site location before sending a SAML
587 request (as described in Section 4.1.1.6).

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

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

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

600 4.1.1.9 Threat Model and Countermeasures

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

602 4.1.1.9.1 Stolen Artifact

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

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

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

- 610 • The source and destination sites SHOULD make some reasonable effort to ensure that clock settings
611 at both sites differ by at most a few minutes. Many forms of time synchronization service are
612 available, both over the Internet and from proprietary sources.
- 613 • SAML assertions communicated in step 5 MUST include an SSO assertion.
- 614 • The source site SHOULD track the time difference between when a SAML artifact is generated and
615 placed on a URL line and when a `<samlp:Request>` message carrying the artifact is received from
616 the destination. A maximum time limit of a few minutes is recommended. Should an assertion be
617 requested by a destination site query beyond this time limit, the source site MUST not provide the
618 assertions to the destination site.
- 619 • It is possible for the source site to create SSO assertions either when the corresponding SAML
620 artifact is created or when a `<samlp:Request>` message carrying the artifact is received from the
621 destination. The validity period of the assertion SHOULD be set appropriately in each case: longer for
622 the former, shorter for the latter.
- 623 • Values for `NotBefore` and `NotOnOrAfter` attributes of SSO assertions SHOULD have the shortest
624 possible validity period consistent with successful communication of the assertion from source to
625 destination site. This is typically on the order of a few minutes. This ensures that a stolen artifact can
626 only be used successfully within a small time window.
- 627 • The destination site MUST check the validity period of all assertions obtained from the source site
628 and reject expired assertions. A destination site MAY choose to implement a stricter test of validity for
629 SSO assertions, such as requiring the assertion's `IssueInstant` or `AuthenticationInstant`
630 attribute value to be within a few minutes of the time at which the assertion is received at the
631 destination site.
- 632 • If a received authentication statement includes a `<saml:SubjectLocality>` element with the IP
633 address of the user, the destination site MAY check the browser IP address against the IP address
634 contained in the authentication statement.

635 4.1.1.9.2 Attacks on the SAML Protocol Message Exchange

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

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

640 4.1.1.9.3 Malicious Destination Site

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

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

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

652 4.1.1.9.4 Forged SAML Artifact

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

654 **Countermeasure:** Section 4.1.1.8 provides specific recommendations regarding the construction of a
655 SAML artifact such that it is infeasible to guess or construct the value of a current, valid, and outstanding
656 assertion handle. A malicious user could attempt to repeatedly “guess” a valid SAML artifact value (one
657 that corresponds to an existing assertion at a source site), but given the size of the value space, this
658 action would likely require a very large number of failed attempts. A source site **SHOULD** implement
659 measures to ensure that repeated attempts at querying against non-existent artifacts result in an alarm.

660 4.1.1.9.5 Browser State Exposure

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

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

668 4.1.2 Browser/POST Profile of SAML

669 4.1.2.1 Required Information

670 **Identification:** urn:oasis:names:tc:SAML:1.0:profiles:browser-post

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

672 **SAML Confirmation Method Identifiers:** The "Bearer" confirmation method identifier is used by this
673 profile. The following identifier has been assigned to this confirmation method:

674 urn:oasis:names:tc:SAML:1.0:cm:bearer

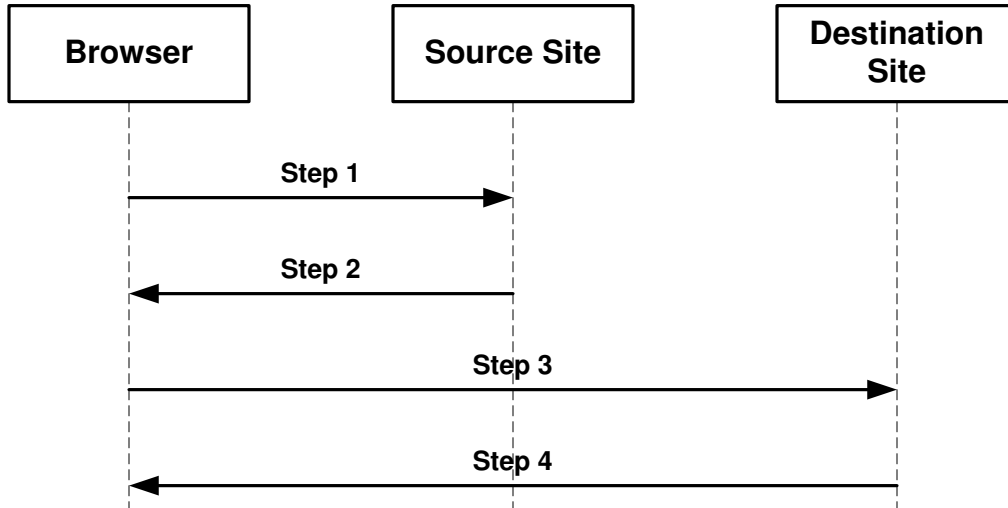
675 **Description:** Given below.

676 **Updates:** None.

677 **4.1.2.2 Preliminaries**

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

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



684

685 **4.1.2.3 Step 1: Accessing the Inter-Site Transfer Service**

686 In step 1, the user's browser accesses the inter-site transfer service at host <https://<inter-site transfer host name>>, with information about the desired target at the destination site attached to the URL.

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

```
690 GET <path>?...TARGET=<Target>...<HTTP-Version>  
691 <other HTTP 1.0 or 1.1 components>
```

692 Where:

693 <inter-site transfer host name>

694 This provides the host name and optional port number at the source site where an inter-site transfer
695 service is available.

696 <path>

697 This provides the path components of an inter-site transfer service URL at the source site.

698 Target=<Target>

699 This name-value pair occurs in the <searchpart> and is used to convey information about the
700 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 message which
703 contains an SSO assertion.

704 **Note:** In the browser/POST profile, the URL used to access the assertion consumer
705 service at the destination site is referred to as the assertion consumer URL.

706 The HTTP response MUST take the form:

```
707 <HTTP-Version> 200 <Reason Phrase>
708 <other HTTP 1.0 or 1.1 components>
```

709 Where:

```
710 <other HTTP 1.0 or 1.1 components>
```

711 This MUST include an HTML FORM (see Chapter 17, [HTML401]) with the following FORM body:

```
712 <Body>
713 <FORM Method="Post" Action="https://<assertion consumer host name and path>" ...>
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 URL at
720 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 an SSO assertion. A single target description MUST be included with the control
724 name TARGET.

725 The notation B64(<response>) stands for the result of applying the Base64 Content-Transfer-Encoding
726 to the response, as defined by [RFC2045] §6.8, and SHOULD consist of lines of encoded data of up to
727 76 characters. The first encoded line begins after the opening quote signifying the "value" attribute of the
728 SAMLResponse form element.

729 The character set used to represent the encoded data is determined by the "charset" attribute of the
730 Content-Type of the HTML document containing the form. The character set of the XML document
731 resulting from decoding the data is determined in the normal fashion, and defaults to UTF-8 if no
732 character set is indicated.

733 The SAML response MUST be digitally signed following the guidelines given in [SAMLCore]. Assertions
734 included in the SAML response MAY be digitally signed.

735 Confidentiality and message integrity MUST be maintained for step 2. It is RECOMMENDED that the
736 inter-site transfer URL be protected by SSL 3.0 or TLS 1.0 (see Section 6). Otherwise, the assertions
737 returned will be available in plain text to any attacker who might then be able to impersonate the assertion
738 subject.

739 4.1.2.5 Step 3: Posting the Form Containing the Response

740 In step 3, the browser submits the form containing the SAML response using the following HTTP request
741 to the assertion consumer service at host <https://<assertion consumer host name>>.

742 **Note:** Posting the form can be triggered by various means. For example, a "submit"
743 button could be included in Step 2 by including the following line:

```
744 <INPUT TYPE="Submit" NAME="button" Value="Submit">
```

745 This requires the user to explicitly "submit" the form for the POST request to be sent.
746 Alternatively, JavaScript™ can be used to avoid an additional "submit" step from the user
747 as follows [Anders]:

```
748 <HTML>
749 <BODY Onload="document.forms[0].submit()">
750 <FORM METHOD="POST" ACTION=" https://<assertion consumer host
751 name and path>">
752 ...
753 <INPUT TYPE="HIDDEN" NAME="SAMLResponse"
```

754
755
756
757
758

```
VALUE="base64 encoded SAML Protocol Response">
  <INPUT TYPE="hidden" NAME="TARGET" Value="<Target>">
</FORM>
</BODY>
</HTML>
```

759 The HTTP request MUST include the following components:

760
761

```
POST <path> <HTTP-Version>
<other HTTP 1.0 or 1.1 request components>
```

762 Where:

763 <assertion consumer host name>

764 This provides the host name and optional port number at the destination site where the assertion
765 consumer service URL is available.

766 <path>

767 This provides the path components of the assertion consumer service URL at the destination site.

768 <other HTTP 1.0 or 1.1 request components>

769 This consists of the form data set derived by the browser processing of the form data received in step
770 2 according to § 17.13.3 of [HTML401]. Exactly one SAML response MUST be included within the
771 form data set with control name SAMLResponse; multiple SAML assertions MAY be included in the
772 response. A single target description MUST be included with the control name set to TARGET.

773 The SAML response MUST include the Recipient attribute [SAMLCore] with its value set to
774 https://<assertion consumer host name and path>. At least one of the SAML assertions
775 included within the response MUST be an SSO assertion.

776 The destination site MUST ensure a "single use" policy for SSO assertions communicated by means of
777 this profile.

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

784 Confidentiality and message integrity MUST be maintained for the HTTP request in step 3. It is
785 RECOMMENDED that the assertion consumer URL be protected by SSL 3.0 or TLS 1.0 (see Section 6).
786 Otherwise, the assertions transmitted in step 3 will be available in plain text to any attacker who might
787 then impersonate the assertion subject.

788 Every subject-based statement in the assertion(s) returned to the destination site MUST contain a
789 <saml:SubjectConfirmation> element. The <ConfirmationMethod> element in the
790 <SubjectConfirmation> MUST be set to urn:oasis:names:tc:SAML:1.0:cm:bearer.

791 4.1.2.6 Step 4: Responding to the User's Request for a Resource

792 In step 4, the user's browser is sent an HTTP response that either allows or denies access to the desired
793 resource. The TARGET form element may be used to decide how to respond to the request and what
794 resource to return, possibly via a redirect or some other means,

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

797 4.1.2.7 Threat Model and Countermeasures

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

799 4.1.2.7.1 Stolen Assertion

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

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

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

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

824 4.1.2.7.2 MITM Attack

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

828 **Countermeasure:** The destination site MUST check the Recipient attribute of the SAML response to
829 ensure that its value matches the `https://<assertion consumer host name and path>`. As the
830 response is digitally signed, the `Recipient` value cannot be altered by the malicious site.

831 4.1.2.7.3 Forged Assertion

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

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

836 4.1.2.7.4 Browser State Exposure

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

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

844 5 Confirmation Method Identifiers

845 The SAML assertion and protocol specification [SAMLCore] defines `<ConfirmationMethod>` as part
846 of the `<SubjectConfirmation>` element. The `<SubjectConfirmation>` element SHOULD be used
847 by the relying party to confirm that the request or message came from the System Entity that corresponds
848 to the subject in the statement. The `<ConfirmationMethod>` element indicates the specific method
849 that the relying party should use to make this judgment. This may or may not have any relationship to an
850 authentication that was performed previously. Unlike the authentication method, the subject confirmation
851 method will often be accompanied by some piece of information, such as a certificate or key, in the
852 `<SubjectConfirmationData>` and/or `<ds:KeyInfo>` elements that will allow the relying party to
853 perform the necessary check.

854 It is anticipated that profiles and bindings will define and use several different values for
855 `<ConfirmationMethod>`, each corresponding to a different SAML usage scenario. Some examples
856 are as follows:

- 857 • A website employs the browser/artifact profile of SAML to sign in a user. The
858 `<ConfirmationMethod>` element in the resulting assertion is set to either
859 `urn:oasis:names:tc:SAML:1.0:cm:artifact-01` (deprecated) or `urn:oasis:names:tc:SAML:1.0:cm:artifact`
860 (RECOMMENDED).
- 861 • There is no login, but an application request sent to a relying party includes SAML assertions and is
862 digitally signed. The associated public key from the `<ds:KeyInfo>` element is used for confirmation.

863 5.1 Holder of Key

864 **URI:** `urn:oasis:names:tc:SAML:1.0:cm:holder-of-key`

865 A `<ds:KeyInfo>` element MUST be present within the `<SubjectConfirmation>` element.

866 As described in [XMLSig], the `<ds:KeyInfo>` element holds a key or information that enables an
867 application to obtain a key. The subject of the statement(s) in the assertion is the party that can
868 demonstrate that it is the holder of the key.

869 5.2 Sender Vouches

870 **URI:** `urn:oasis:names:tc:SAML:1.0:cm:sender-vouches`

871 Indicates that no other information is available about the context of use of the assertion. The relying party
872 SHOULD utilize other means to determine if it should process the assertion further.

873 5.3 SAML Artifact

874 **Recommended URI:** `urn:oasis:names:tc:SAML:1.0:cm:artifact`

875 **Deprecated URI:** `urn:oasis:names:tc:SAML:1.0:cm:artifact-01`

876 The subject of the assertion is the party that presented a SAML artifact, which the relying party used to
877 obtain the assertion from the party that created the artifact. See also Section 4.1.1.1.

878 5.4 Bearer

879 **URI:** `urn:oasis:names:tc:SAML:1.0:cm:bearer`

880 The subject of the assertion is the bearer of the assertion. See also Section 4.1.2.1.

881 **6 Use of SSL 3.0 or TLS 1.0**

882 In any SAML use of SSL 3.0 [**SSL3**] or TLS 1.0 [**RFC2246**], servers MUST authenticate to clients using a
883 X.509 v3 certificate. The client MUST establish server identity based on contents of the certificate
884 (typically through examination of the certificate's subject DN field).

885 **6.1 SAML SOAP Binding**

886 TLS-capable implementations MUST implement the TLS_RSA_WITH_3DES_EDE_CBC_SHA cipher
887 suite and MAY implement the TLS_RSA_AES_128_CBC_SHA cipher suite [**AES**].

888 **6.2 Web Browser Profiles of SAML**

889 SSL-capable implementations of the browser/artifact profile or browser/POST profile of SAML MUST
890 implement the SSL_RSA_WITH_3DES_EDE_CBC_SHA cipher suite.

891 TLS-capable implementations MUST implement the TLS_RSA_WITH_3DES_EDE_CBC_SHA cipher
892 suite.

893 7 Alternative SAML Artifact Format

894 7.1 Required Information

895 **Identification:** urn:oasis:names:tc:SAML:1.0:profiles:artifact-02

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

897 **Description:** Given below.

898 **Updates:** None.

899 7.2 Format Details

900 An alternative artifact format is described here:

901	TypeCode	:=	0x0002
902	RemainingArtifact	:=	AssertionHandle SourceLocation
903	AssertionHandle	:=	20-byte_sequence
904	SourceLocation	:=	URI

905 The `SourceLocation` URI is the address of the SAML responder associated with the source site. The
906 `assertionHandle` is as described in Section 4.1.1.8, and governed by the same requirements. The
907 `SourceLocation` URI is mapped to a sequence of bytes based on use of the UTF-8 **[RFC2279]**
908 encoding. The destination site **MUST** process the artifact in a manner identical to that described in
909 Section 4.1.1, with the exception that the location of the SAML responder at the source site **MAY** be
910 obtained directly from the artifact, rather than by look-up, based on `sourceID`.

911 Note: the destination site **MUST** confirm that assertions were issued by an acceptable issuer, not relying
912 merely on the fact that they were returned in response to a `<samlp:Request>` message.

913 8 URL Size Restriction (Non-Normative)

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

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

917 The information in this article applies to:

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

919 SUMMARY

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

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

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

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

929 REFERENCES

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

932 Last Reviewed: 9/13/2001

933 Keywords: kbDSupport kbFAQ kbinfo KB208427

934 An article about Netscape Enterprise Server provides the following information:

935 Issue: 19971110-3 Product: Enterprise Server

936 Created: 11/10/1997 Version: 2.01

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

938 Does this article answer your question?

939 Please let us know!

940 Question:

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

943 Answer:

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

945

946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992

9 References

- [AES]** FIPS-197, Advanced Encryption Standard (AES), available from <http://www.nist.gov/>.
- [Anders]** A suggestion on how to implement SAML browser bindings without using “Artifacts”, <http://www.x-obi.com/OBI400/andersr-browser-artifact.ppt>.
- [CoreAssnEx]** Core Assertions Architecture, Examples and Explanations, <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-phill-07.pdf>.
- [HTML401]** HTML 4.01 Specification, W3C Recommendation 24 December 1999, <http://www.w3.org/TR/html4>.
- [Liberty]** The Liberty Alliance Project, <http://www.projectliberty.org>.
- [MSURL]** Microsoft technical support article, <http://support.microsoft.com/support/kb/articles/Q208/4/27.ASP>.
- [Rescorla-Sec]** E. Rescorla et al., *Guidelines for Writing RFC Text on Security Considerations*, <http://www.ietf.org/internet-drafts/draft-iab-sec-cons-03.txt>.
- [RFC1738]** Uniform Resource Locators (URL), <http://www.ietf.org/rfc/rfc1738.txt>
- [RFC1750]** Randomness Recommendations for Security. <http://www.ietf.org/rfc/rfc1750.txt>
- [RFC1945]** Hypertext Transfer Protocol -- HTTP/1.0, <http://www.ietf.org/rfc/rfc1945.txt>.
- [RFC2045]** Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, <http://www.ietf.org/rfc/rfc2045.txt>
- [RFC2119]** S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, IETF RFC 2119, March 1997, <http://www.ietf.org/rfc/rfc2119.txt>.
- [RFC2246]** The TLS Protocol Version 1.0, <http://www.ietf.org/rfc/rfc2246.txt>.
- [RFC2279]** UTF-8, a transformation format of ISO 10646, <http://www.ietf.org/rfc/rfc2279.txt>.
- [RFC2616]** Hypertext Transfer Protocol -- HTTP/1.1, <http://www.ietf.org/rfc/rfc2616.txt>.
- [RFC2617]** *HTTP Authentication: Basic and Digest Access Authentication*, IETF RFC 2617, <http://www.ietf.org/rfc/rfc2617.txt>.
- [SAMLCore]** E. Maler et al., *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)*, OASIS, May 2003, <http://www.oasis-open.org/committees/security/>.
- [SAMLGloss]** E. Maler et al., *Glossary for the OASIS Security Assertion Markup Language (SAML)*, OASIS, May 2003, <http://www.oasis-open.org/committees/security/>.
- [SAMLSec]** E. Maler et al., *Security Considerations for the OASIS Security Assertion Markup Language (SAML)*, OASIS, May 2003, <http://www.oasis-open.org/committees/security/>.
- [SAMLReqs]** Darren Platt et al., *SAML Requirements and Use Cases*, OASIS, April 2002, <http://www.oasis-open.org/committees/security/>.
- [SAMLWeb]** OASIS Security Services Technical Committee website, <http://www.oasis-open.org/committees/security>.
- [SESSION]** RL “Bob” Morgan, Support of target web server sessions in Shibboleth, <http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-session-00.txt>
- [ShibMarlena]** Marlena Erdos, Shibboleth Architecture DRAFT v1.1, <http://shibboleth.internet2.edu/draft-internet2-shibboleth-arch-v05.html> .
- [SOAP1.1]** D. Box et al., *Simple Object Access Protocol (SOAP) 1.1*, World Wide Web Consortium Note, May 2000, <http://www.w3.org/TR/SOAP>.
- [SSL3]** A. Frier et al., *The SSL 3.0 Protocol*, Netscape Communications Corp, November 1996.

993 **[WEBSO]** RL “Bob” Morgan, Interactions between Shibboleth and local-site web sign-on
994 services, [http://middleware.internet2.edu/shibboleth/docs/draft-morgan-](http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-websso-00.txt)
995 [shibboleth-websso-00.txt](http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-websso-00.txt)
996 **[WSS-SAML]** P. Hallam-Baker et al., *Web Services Security: SAML Token Profile*, OASIS,
997 March 2003, <http://www.oasis-open.org/committees/wss>.
998 **[XMLSig]** D. Eastlake et al., *XML-Signature Syntax and Processing*, World Wide Web
999 Consortium, <http://www.w3.org/TR/xmlsig-core/>.

1000 **Appendix A. Acknowledgments**

1001 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
1002 Committee, whose voting members at the time of publication were:

- 1003 • Frank Siebenlist, Argonne National Laboratory
- 1004 • Irving Reid, Baltimore Technologies
- 1005 • Hal Lockhart, BEA Systems
- 1006 • Steven Lewis, Booz Allen Hamilton
- 1007 • John Hughes, Entegriy Solutions
- 1008 • Carlisle Adams, Entrust
- 1009 • Jason Rouault, Hewlett-Packard
- 1010 • Maryann Hondo, IBM
- 1011 • Anthony Nadalin, IBM
- 1012 • Scott Cantor, individual
- 1013 • RL "Bob" Morgan, individual
- 1014 • Trevor Perrin, individual
- 1015 • Padraig Moloney, NASA
- 1016 • Prateek Mishra, Netegrity (co-chair)
- 1017 • Frederick Hirsch, Nokia
- 1018 • Senthil Sengodan, Nokia
- 1019 • Timo Skytta, Nokia
- 1020 • Charles Knouse, Oblix
- 1021 • Steve Anderson, OpenNetwork
- 1022 • Simon Godik, Overxeer
- 1023 • Rob Philpott, RSA Security (co-chair)
- 1024 • Dipak Chopra, SAP
- 1025 • Jahan Moreh, Sigaba
- 1026 • Bhavna Bhatnagar, Sun Microsystems
- 1027 • Jeff Hodges, Sun Microsystems
- 1028 • Eve Maler, Sun Microsystems (coordinating editor)
- 1029 • Emily Xu, Sun Microsystems
- 1030 • Phillip Hallam-Baker, VeriSign

1031

Appendix B. Notices

1032 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
1033 might be claimed to pertain to the implementation or use of the technology described in this document or
1034 the extent to which any license under such rights might or might not be available; neither does it
1035 represent that it has made any effort to identify any such rights. Information on OASIS's procedures with
1036 respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights
1037 made available for publication and any assurances of licenses to be made available, or the result of an
1038 attempt made to obtain a general license or permission for the use of such proprietary rights by
1039 implementors or users of this specification, can be obtained from the OASIS Executive Director.

1040 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications,
1041 or other proprietary rights which may cover technology that may be required to implement this
1042 specification. Please address the information to the OASIS Executive Director.

1043 **Copyright © OASIS Open 2003. All Rights Reserved.**

1044 This document and translations of it may be copied and furnished to others, and derivative works that
1045 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published
1046 and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice
1047 and this paragraph are included on all such copies and derivative works. However, this document itself
1048 may not be modified in any way, such as by removing the copyright notice or references to OASIS,
1049 except as needed for the purpose of developing OASIS specifications, in which case the procedures for
1050 copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to
1051 translate it into languages other than English.

1052 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
1053 or assigns.

1054 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1055 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
1056 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
1057 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

1058 JavaScript is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and
1059 other countries.