



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

OASIS Standard, 2 September 2003

Document identifier:

oasis-sstc-saml-bindings-1.1

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 is an OASIS Standard document produced by the Security Services Technical Committee. It was approved by the OASIS membership on 2 September 2003.

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

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 web page for the Security Services TC ([http://www.oasis-
open.org/committees/security/ipr.php](http://www.oasis-
43 open.org/committees/security/ipr.php)).

Table of Contents

45	1	Introduction	5
46	1.1	Protocol Binding and Profile Concepts	5
47	1.2	Notation	5
48	2	Specification of Additional Protocol Bindings and Profiles	7
49	2.1	Guidelines for Specifying Protocol Bindings and Profiles	7
50	2.2	Process Framework for Describing and Registering Protocol Bindings and Profiles	7
51	3	Protocol Bindings	8
52	3.1	SAML SOAP Binding	8
53	3.1.1	Required Information	8
54	3.1.2	Protocol-Independent Aspects of the SAML SOAP Binding	8
55	3.1.2.1	Basic Operation	8
56	3.1.2.2	SOAP Headers	9
57	3.1.2.3	Authentication	9
58	3.1.2.4	Message Integrity	9
59	3.1.2.5	Confidentiality	9
60	3.1.3	Use of SOAP over HTTP	9
61	3.1.3.1	HTTP Headers	10
62	3.1.3.2	Authentication	10
63	3.1.3.3	Message Integrity	10
64	3.1.3.4	Message Confidentiality	10
65	3.1.3.5	Security Considerations	10
66	3.1.3.6	Error Reporting	10
67	3.1.3.7	Example SAML Message Exchange Using SOAP over HTTP	11
68	4	Profiles	12
69	4.1	Web Browser SSO Profiles of SAML	12
70	4.1.1	Browser/Artifact Profile of SAML	13
71	4.1.1.1	Required Information	13
72	4.1.1.2	Preliminaries	14
73	4.1.1.3	Step 1: Accessing the Inter-Site Transfer Service	15
74	4.1.1.4	Step 2: Redirecting to the Destination Site	15
75	4.1.1.5	Step 3: Accessing the Artifact Receiver URL	16
76	4.1.1.6	Steps 4 and 5: Acquiring the Corresponding Assertions	16
77	4.1.1.7	Step 6: Responding to the User's Request for a Resource	17
78	4.1.1.8	Artifact Format	17
79	4.1.1.9	Threat Model and Countermeasures	18
80	4.1.1.9.1	Stolen Artifact	18
81	4.1.1.9.2	Attacks on the SAML Protocol Message Exchange	18
82	4.1.1.9.3	Malicious Destination Site	19
83	4.1.1.9.4	Forged SAML Artifact	19
84	4.1.1.9.5	Browser State Exposure	19
85	4.1.2	Browser/POST Profile of SAML	19
86	4.1.2.1	Required Information	19
87	4.1.2.2	Preliminaries	20

88	4.1.2.3 Step 1: Accessing the Inter-Site Transfer Service.....	20
89	4.1.2.4 Step 2: Generating and Supplying the Response	20
90	4.1.2.5 Step 3: Posting the Form Containing the Response	21
91	4.1.2.6 Step 4: Responding to the User's Request for a Resource.....	22
92	4.1.2.7 Threat Model and Countermeasures	22
93	4.1.2.7.1 Stolen Assertion	23
94	4.1.2.7.2 MITM Attack.....	23
95	4.1.2.7.3 Forged Assertion.....	23
96	4.1.2.7.4 Browser State Exposure.....	23
97	5 Confirmation Method Identifiers	24
98	5.1 Holder of Key	24
99	5.2 Sender Vouches	24
100	5.3 SAML Artifact.....	24
101	5.4 Bearer	24
102	6 Use of SSL 3.0 or TLS 1.0	25
103	6.1 SAML SOAP Binding	25
104	6.2 Web Browser Profiles of SAML	25
105	7 Alternative SAML Artifact Format	26
106	7.1 Required Information	26
107	7.2 Format Details	26
108	8 URL Size Restriction (Non-Normative).....	27
109	9 References	28
110	Appendix A. Acknowledgments	30
111	Appendix B. Notices.....	31

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, 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 SAML assertions into and extract them from 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 site, 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.

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

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

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

155 2 Specification of Additional Protocol Bindings and 156 Profiles

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

163 2.1 Guidelines for Specifying Protocol Bindings and Profiles

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

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

181 2.2 Process Framework for Describing and Registering Protocol 182 Bindings and Profiles

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

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

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

196 3 Protocol Bindings

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

199 3.1 SAML SOAP Binding

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

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

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

209 3.1.1 Required Information

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

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

212 **Description:** Given below.

213 **Updates:** None.

214 3.1.2 Protocol-Independent Aspects of the SAML SOAP Binding

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

217 3.1.2.1 Basic Operation

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

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

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

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

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

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

244 3.1.2.2 SOAP Headers

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

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

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

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

254 3.1.2.3 Authentication

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

259 3.1.2.4 Message Integrity

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

264 3.1.2.5 Confidentiality

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

269 3.1.3 Use of SOAP over HTTP

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

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

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

277 **3.1.3.1 HTTP Headers**

278 HTTP proxies MUST NOT cache responses carrying SAML assertions.

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

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

284 There are no other restrictions on HTTP headers.

285 **3.1.3.2 Authentication**

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

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

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

293 **3.1.3.3 Message Integrity**

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

296 **3.1.3.4 Message Confidentiality**

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

299 **3.1.3.5 Security Considerations**

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

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

305 **3.1.3.6 Error Reporting**

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

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

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

- 316 • As the only child of the `<SOAP-ENV:Body>` element. This mechanism is deprecated in SAML V1.1
317 and will be removed in the next major revision of SAML.

- 318 • As the only child of a SAML <Response> element within the SOAP body (RECOMMENDED).
319 For more information about SAML status codes, see the SAML assertion and protocol specification
320 [SAMLCore].

321 3.1.3.7 Example SAML Message Exchange Using SOAP over HTTP

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

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

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

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

362

4 Profiles

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

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

- 367 • The browser/artifact profile of SAML
- 368 • The browser/POST profile of SAML

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

371 Some additional profiles that have been published outside the Security Services Technical Committee
372 are:

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

377 4.1 Web Browser SSO Profiles of SAML

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

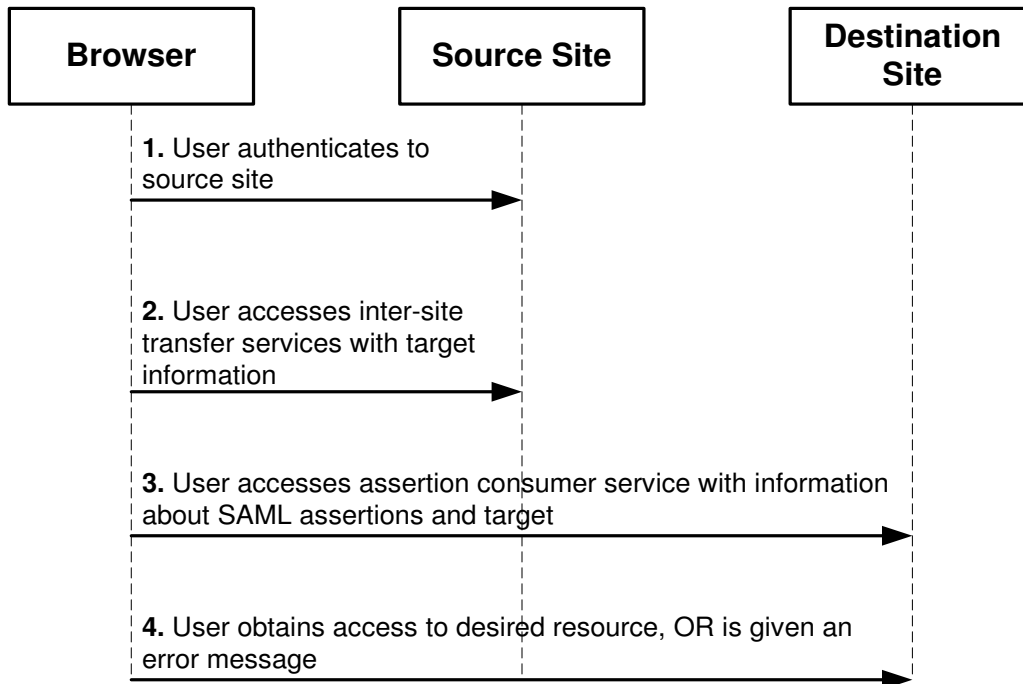
381 The following assumptions are made about this scenario for the purposes of these profiles:

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

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

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

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



401

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

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

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

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

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

419 4.1.1 Browser/Artifact Profile of SAML

420 4.1.1.1 Required Information

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

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

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

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

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

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

428 **Description:** Given below.

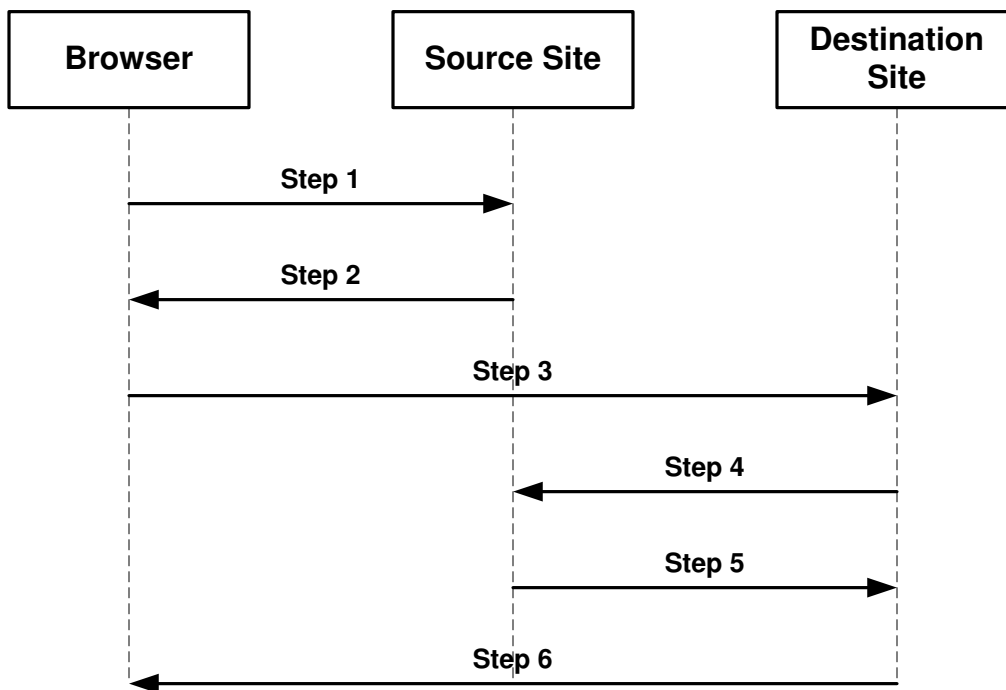
429 **Updates:** None.

430 4.1.1.2 Preliminaries

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

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

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



447
448 Terminology from RFC 1738 [RFC1738] is used to describe components of a URL. An HTTP URL has
449 the following form:

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

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

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

455 4.1.1.3 Step 1: Accessing the Inter-Site Transfer Service

456 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.

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

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

462 Where:

463 <inter-site transfer host name>

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

466 <path>

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

468 Target=<Target>

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

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

472 4.1.1.4 Step 2: Redirecting to the Destination Site

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

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

478 The HTTP response MUST take the following form:

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

483 Where:

484 <artifact receiver host name and path>

485 This provides the host name, port number, and path components of an artifact receiver URL
486 associated with the assertion consumer service 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 least one
489 SAML artifact MUST be included in the SAML <SAML searchpart> component; multiple SAML
490 artifacts MAY be included. If more than one artifact is carried within <SAML searchpart>, all the
491 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 URI". The
494 response may also include additional headers and an optional message body as described in those
495 RFCs.

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

500 **4.1.1.5 Step 3: Accessing the Artifact Receiver URL**

501 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.

503 The HTTP request MUST take the form:

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

506 Where:

507 <artifact receiver host name>

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

510 <path>

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

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

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

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

521 **4.1.1.6 Steps 4 and 5: Acquiring the Corresponding Assertions**

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

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

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

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

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

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

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

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

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

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

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

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

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

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

559 4.1.1.7 Step 6: Responding to the User's Request for a Resource

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

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

564 4.1.1.8 Artifact Format

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

```
566 SAML_artifact      := B64(TypeCode RemainingArtifact)  
567 TypeCode           := Byte1Byte2
```

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

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

```
575 TypeCode           := 0x0001  
576 RemainingArtifact := SourceID AssertionHandle  
577 SourceID           := 20-byte_sequence  
578 AssertionHandle   := 20-byte_sequence
```

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

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

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

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

597 4.1.1.9 Threat Model and Countermeasures

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

599 4.1.1.9.1 Stolen Artifact

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

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

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

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

632 4.1.1.9.2 Attacks on the SAML Protocol Message Exchange

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

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

637 4.1.1.9.3 Malicious Destination Site

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

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

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

649 4.1.1.9.4 Forged SAML Artifact

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

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

657 4.1.1.9.5 Browser State Exposure

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

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

665 4.1.2 Browser/POST Profile of SAML

666 4.1.2.1 Required Information

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

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

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

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

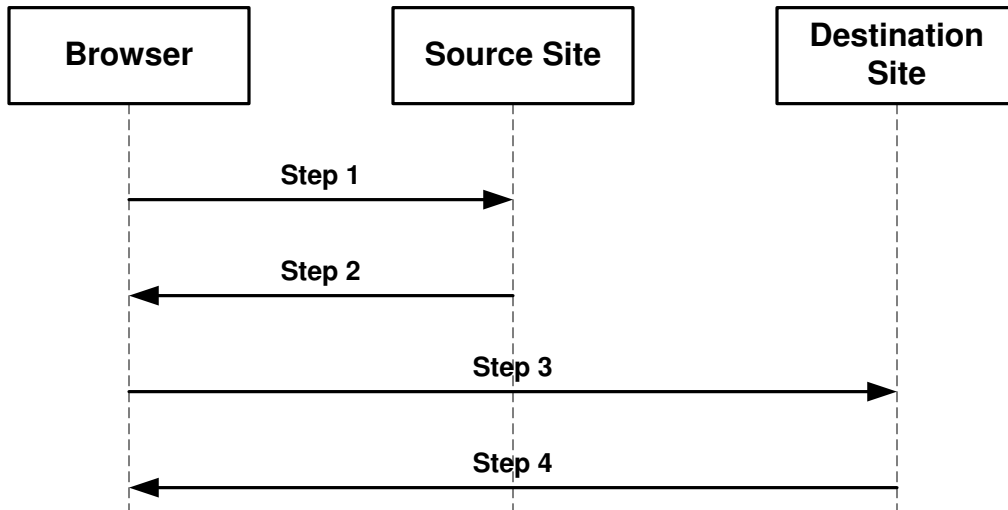
672 **Description:** Given below.

673 **Updates:** None.

674 **4.1.2.2 Preliminaries**

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

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



681

682 **4.1.2.3 Step 1: Accessing the Inter-Site Transfer Service**

683 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.

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

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

689 Where:

690 <inter-site transfer host name>

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

693 <path>

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

695 Target=<Target>

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

698 **4.1.2.4 Step 2: Generating and Supplying the Response**

699 In step 2, the source site generates HTML form data containing a SAML response message which
700 contains an SSO assertion.

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

703 The HTTP response MUST take the form:

```
704 <HTTP-Version> 200 <Reason Phrase>
705 <other HTTP 1.0 or 1.1 components>
```

706 Where:

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

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

```
709 <Body>
710 <FORM Method="Post" Action="https://<assertion consumer host name and path>" ...>
711 <INPUT TYPE="hidden" NAME="SAMLResponse" Value="B64(<response>)">
712 ...
713 <INPUT TYPE="hidden" NAME="TARGET" Value="<Target>">
714 </Body>
```

```
715 <assertion consumer host name and path>
```

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

718 Exactly one SAML response MUST be included within the FORM body with the control name
719 SAMLResponse; multiple SAML assertions MAY be included in the response. At least one of the
720 assertions MUST be an SSO assertion. A single target description MUST be included with the control
721 name TARGET.

722 The notation B64(<response>) stands for the result of applying the Base64 Content-Transfer-Encoding
723 to the response, as defined by [RFC2045] §6.8, and SHOULD consist of lines of encoded data of up to
724 76 characters. The first encoded line begins after the opening quote signifying the "value" attribute of the
725 SAMLResponse form element.

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

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

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

736 4.1.2.5 Step 3: Posting the Form Containing the Response

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

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

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

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

```
745 <HTML>
746 <BODY Onload="document.forms[0].submit()">
747 <FORM METHOD="POST" ACTION=" https://<assertion consumer host
748 name and path>">
749 ...
750 <INPUT TYPE="HIDDEN" NAME="SAMLResponse"
751 VALUE="base64 encoded SAML Protocol Response">
```

752
753
754
755

```
<INPUT TYPE="hidden" NAME="TARGET" Value="<Target>">
</FORM>
</BODY>
</HTML>
```

756 The HTTP request MUST include the following components:

757
758

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

759 Where:

760 <assertion consumer host name>

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

763 <path>

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

765 <other HTTP 1.0 or 1.1 request components>

766 This consists of the form data set derived by the browser processing of the form data received in step
767 2 according to § 17.13.3 of [HTML401]. Exactly one SAML response MUST be included within the
768 form data set with control name SAMLResponse; multiple SAML assertions MAY be included in the
769 response. A single target description MUST be included with the control name set to TARGET.

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

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

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

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

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

788 4.1.2.6 Step 4: Responding to the User's Request for a Resource

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

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

794 4.1.2.7 Threat Model and Countermeasures

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

796 4.1.2.7.1 Stolen Assertion

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

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

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

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

821 4.1.2.7.2 MITM Attack

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

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

828 4.1.2.7.3 Forged Assertion

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

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

833 4.1.2.7.4 Browser State Exposure

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

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

841 5 Confirmation Method Identifiers

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

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

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

860 5.1 Holder of Key

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

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

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

866 5.2 Sender Vouches

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

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

870 5.3 SAML Artifact

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

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

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

875 5.4 Bearer

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

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

878 **6 Use of SSL 3.0 or TLS 1.0**

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

882 **6.1 SAML SOAP Binding**

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

885 **6.2 Web Browser Profiles of SAML**

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

888 TLS-capable implementations MUST implement the TLS_RSA_WITH_3DES_EDE_CBC_SHA cipher
889 suite.

890 7 Alternative SAML Artifact Format

891 7.1 Required Information

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

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

894 **Description:** Given below.

895 **Updates:** None.

896 7.2 Format Details

897 An alternative artifact format is described here:

```
898 TypeCode           := 0x0002
899 RemainingArtifact := AssertionHandle SourceLocation
900 AssertionHandle    := 20-byte_sequence
901 SourceLocation     := URI
```

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

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

910 8 URL Size Restriction (Non-Normative)

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

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

914 The information in this article applies to:

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

916 SUMMARY

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

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

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

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

926 REFERENCES

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

929 Last Reviewed: 9/13/2001

930 Keywords: kbDSupport kbFAQ kbinfo KB208427

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

932 Issue: 19971110-3 Product: Enterprise Server

933 Created: 11/10/1997 Version: 2.01

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

935 Does this article answer your question?

936 Please let us know!

937 Question:

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

940 Answer:

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

942

943

9 References

- 944 [AES] FIPS-197, Advanced Encryption Standard (AES), available from
945 <http://www.nist.gov/>.
- 946 [Anders] A suggestion on how to implement SAML browser bindings without using
947 "Artifacts", <http://www.x-obi.com/OBI400/andersr-browser-artifact.ppt>.
- 948 [CoreAssnEx] Core Assertions Architecture, Examples and Explanations, <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-phill-07.pdf>.
949
- 950 [HTML401] HTML 4.01 Specification, W3C Recommendation 24 December 1999,
951 <http://www.w3.org/TR/html4>.
- 952 [Liberty] The Liberty Alliance Project, <http://www.projectliberty.org>.
- 953 [MSURL] Microsoft technical support article,
954 <http://support.microsoft.com/support/kb/articles/Q208/4/27.ASP>.
- 955 [Rescorla-Sec] E. Rescorla et al., *Guidelines for Writing RFC Text on Security Considerations*,
956 <http://www.ietf.org/internet-drafts/draft-iab-sec-cons-03.txt>.
- 957 [RFC1738] Uniform Resource Locators (URL), <http://www.ietf.org/rfc/rfc1738.txt>
- 958 [RFC1750] Randomness Recommendations for Security. <http://www.ietf.org/rfc/rfc1750.txt>
- 959 [RFC1945] Hypertext Transfer Protocol -- HTTP/1.0, <http://www.ietf.org/rfc/rfc1945.txt>.
- 960 [RFC2045] Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet
961 Message Bodies, <http://www.ietf.org/rfc/rfc2045.txt>
- 962 [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, IETF
963 RFC 2119, March 1997, <http://www.ietf.org/rfc/rfc2119.txt>.
- 964 [RFC2246] The TLS Protocol Version 1.0, <http://www.ietf.org/rfc/rfc2246.txt>.
- 965 [RFC2279] UTF-8, a transformation format of ISO 10646, <http://www.ietf.org/rfc/rfc2279.txt>.
- 966 [RFC2616] Hypertext Transfer Protocol -- HTTP/1.1, <http://www.ietf.org/rfc/rfc2616.txt>.
- 967 [RFC2617] *HTTP Authentication: Basic and Digest Access Authentication*, IETF RFC 2617,
968 <http://www.ietf.org/rfc/rfc2617.txt>.
- 969 [SAMLCore] E. Maler et al. *Assertions and Protocol for the OASIS Security Assertion Markup
970 Language (SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-core-
971 1.1. <http://www.oasis-open.org/committees/security/>.
- 972 [SAMLGloss] E. Maler et al. *Glossary for the OASIS Security Assertion Markup Language
973 (SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-glossary-1.1.
974 <http://www.oasis-open.org/committees/security/>.
- 975 [SAMLSec] E. Maler et al. *Security Considerations for the OASIS Security Assertion Markup
976 Language (SAML)*, OASIS, September 2003, Document ID oasis-sstc-saml-sec-
977 consider-1.1. <http://www.oasis-open.org/committees/security/>.
- 978 [SAMLReqs] Darren Platt et al., *SAML Requirements and Use Cases*, OASIS, April 2002,
979 <http://www.oasis-open.org/committees/security/>.
- 980 [SAMLWeb] OASIS Security Services Technical Committee website, <http://www.oasis-open.org/committees/security>.
- 981
- 982 [SESSION] RL "Bob" Morgan, Support of target web server sessions in Shibboleth,
983 [http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-session-
984 00.txt](http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-session-00.txt)
- 985 [ShibMarlena] Marlena Erdos, Shibboleth Architecture DRAFT v1.1,
986 <http://shibboleth.internet2.edu/draft-internet2-shibboleth-arch-v05.html> .
- 987 [SOAP1.1] D. Box et al., *Simple Object Access Protocol (SOAP) 1.1*, World Wide Web
988 Consortium Note, May 2000, <http://www.w3.org/TR/SOAP>.
- 989 [SSL3] A. Frier et al., *The SSL 3.0 Protocol*, Netscape Communications Corp, November
990 1996.

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

Appendix A. Acknowledgments

999 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
1000 Committee, whose voting members at the time of publication were:

- 1001 • Frank Siebenlist, Argonne National Laboratory
- 1002 • Irving Reid, Baltimore Technologies
- 1003 • Hal Lockhart, BEA Systems
- 1004 • Steven Lewis, Booz Allen Hamilton
- 1005 • John Hughes, Entegriy Solutions
- 1006 • Carlisle Adams, Entrust
- 1007 • Jason Rouault, Hewlett-Packard
- 1008 • Maryann Hondo, IBM
- 1009 • Anthony Nadalin, IBM
- 1010 • Scott Cantor, individual
- 1011 • RL "Bob" Morgan, individual
- 1012 • Trevor Perrin, individual
- 1013 • Pdraig Moloney, NASA
- 1014 • Prateek Mishra, Netegrity (co-chair)
- 1015 • Frederick Hirsch, Nokia
- 1016 • Senthil Sengodan, Nokia
- 1017 • Timo Skytta, Nokia
- 1018 • Charles Knouse, Oblix
- 1019 • Steve Anderson, OpenNetwork
- 1020 • Simon Godik, Overxeer
- 1021 • Rob Philpott, RSA Security (co-chair)
- 1022 • Dipak Chopra, SAP
- 1023 • Jahan Moreh, Sigaba
- 1024 • Bhavna Bhatnagar, Sun Microsystems
- 1025 • Jeff Hodges, Sun Microsystems
- 1026 • Eve Maler, Sun Microsystems (coordinating editor)
- 1027 • Emily Xu, Sun Microsystems
- 1028 • Phillip Hallam-Baker, VeriSign

1029

Appendix B. Notices

1030 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
1031 might be claimed to pertain to the implementation or use of the technology described in this document or
1032 the extent to which any license under such rights might or might not be available; neither does it
1033 represent that it has made any effort to identify any such rights. Information on OASIS's procedures with
1034 respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights
1035 made available for publication and any assurances of licenses to be made available, or the result of an
1036 attempt made to obtain a general license or permission for the use of such proprietary rights by
1037 implementors or users of this specification, can be obtained from the OASIS Executive Director.

1038 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications,
1039 or other proprietary rights which may cover technology that may be required to implement this
1040 specification. Please address the information to the OASIS Executive Director.

1041 **Copyright © OASIS Open 2003. All Rights Reserved.**

1042 This document and translations of it may be copied and furnished to others, and derivative works that
1043 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published
1044 and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice
1045 and this paragraph are included on all such copies and derivative works. However, this document itself
1046 may not be modified in any way, such as by removing the copyright notice or references to OASIS,
1047 except as needed for the purpose of developing OASIS specifications, in which case the procedures for
1048 copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to
1049 translate it into languages other than English.

1050 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
1051 or assigns.

1052 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1053 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
1054 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
1055 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

1056 JavaScript is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and
1057 other countries.