



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

## Last Call Working Draft 06, 2 May 2003

### Document identifier:

sstc-saml-bindings-1.1-draft-06

### Location:

[http://www.oasis-open.org/committees/documents.php?wg\\_abbrev=security](http://www.oasis-open.org/committees/documents.php?wg_abbrev=security)

### Editors:

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

### Contributors:

Irving Reid, Baltimore Technologies  
Krishna Sankar, Cisco Systems  
Simon Godik, Crosslogix  
Tim Moses, Entrust  
Scott Cantor, Ohio State University and Internet2  
Evan Prodromou, formerly with Securant  
Jahan Moreh, Sigaba  
Chris Ferris, Sun Microsystems  
Jeff Hodges, Sun Microsystems  
Bob Blakley, Tivoli  
Marlena Erdos, Tivoli  
RL "Bob" Morgan, University of Washington and Internet2

### 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 **last-call working draft** of the OASIS Security Services Technical Committee. We solicit your comments; they must be received by Friday, 16 May 2003 in order for the committee to consider them for inclusion in the Committee Specification.

If you are on the [security-services@lists.oasis-open.org](mailto:security-services@lists.oasis-open.org) list for committee members, send comments there. If you are not on that list, subscribe to the [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org) list and send comments there. To subscribe, send an email message to [security-services-comment-request@lists.oasis-open.org](mailto:security-services-comment-request@lists.oasis-open.org) with the word "subscribe" as the body of the message.

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

## Table of Contents

44	1	Introduction .....	5
45	1.1	Protocol Binding and Profile Concepts .....	5
46	1.2	Notation .....	5
47	2	Specification of Additional Protocol Bindings and Profiles .....	7
48	2.1	Guidelines for Specifying Protocol Bindings and Profiles .....	7
49	2.2	Process Framework for Describing and Registering Protocol Bindings and Profiles .....	7
50	3	Protocol Bindings .....	8
51	3.1	SAML SOAP Binding .....	8
52	3.1.1	Required Information .....	8
53	3.1.2	Protocol-Independent Aspects of the SAML SOAP Binding .....	8
54	3.1.2.1	Basic Operation .....	8
55	3.1.2.2	SOAP Headers .....	9
56	3.1.2.3	Authentication .....	9
57	3.1.2.4	Message Integrity .....	9
58	3.1.2.5	Confidentiality .....	9
59	3.1.3	Use of SOAP over HTTP .....	9
60	3.1.3.1	HTTP Headers .....	10
61	3.1.3.2	Authentication .....	10
62	3.1.3.3	Message Integrity .....	10
63	3.1.3.4	Message Confidentiality .....	10
64	3.1.3.5	Security Considerations .....	10
65	3.1.3.6	Error Reporting .....	10
66	3.1.3.7	Example SAML Message Exchange Using SOAP over HTTP .....	11
67	4	Profiles .....	12
68	4.1	Web Browser SSO Profiles of SAML .....	12
69	4.1.1	Browser/Artifact Profile of SAML .....	13
70	4.1.1.1	Required Information .....	13
71	4.1.1.2	Preliminaries .....	14
72	4.1.1.3	Step 1: Accessing the Inter-Site Transfer Service .....	15
73	4.1.1.4	Step 2: Redirecting to the Destination Site .....	15
74	4.1.1.5	Step 3: Accessing the Artifact Receiver URL .....	16
75	4.1.1.6	Steps 4 and 5: Acquiring the Corresponding Assertions .....	16
76	4.1.1.7	Step 6: Responding to the User's Request for a Resource .....	17
77	4.1.1.8	Artifact Format .....	17
78	4.1.1.9	Threat Model and Countermeasures .....	18
79	4.1.1.9.1	Stolen Artifact .....	18
80	4.1.1.9.2	Attacks on the SAML Protocol Message Exchange .....	18
81	4.1.1.9.3	Malicious Destination Site .....	19
82	4.1.1.9.4	Forged SAML Artifact .....	19
83	4.1.1.9.5	Browser State Exposure .....	19
84	4.1.2	Browser/POST Profile of SAML .....	19
85	4.1.2.1	Required Information .....	19
86	4.1.2.2	Preliminaries .....	19

87	4.1.2.3 Step 1: Accessing the Inter-Site Transfer Service.....	20
88	4.1.2.4 Step 2: Generating and Supplying the Response .....	20
89	4.1.2.5 Step 3: Posting the Form Containing the Response .....	21
90	4.1.2.6 Step 4: Responding to the User's Request for a Resource.....	22
91	4.1.2.7 Threat Model and Countermeasures .....	22
92	4.1.2.7.1 Stolen Assertion .....	22
93	4.1.2.7.2 MITM Attack.....	23
94	4.1.2.7.3 Forged Assertion.....	23
95	4.1.2.7.4 Browser State Exposure.....	23
96	5 Confirmation Method Identifiers .....	24
97	5.1 Holder of Key .....	24
98	5.2 Sender Vouches .....	24
99	5.3 SAML Artifact.....	24
100	5.4 Bearer .....	24
101	6 Use of SSL 3.0 or TLS 1.0 .....	25
102	6.1 SAML SOAP Binding .....	25
103	6.2 Web Browser Profiles of SAML .....	25
104	7 Alternative SAML Artifact Format .....	26
105	7.1 Required Information .....	26
106	7.2 Format Details .....	26
107	8 URL Size Restriction (Non-Normative).....	27
108	9 References .....	28
109	Appendix A. Acknowledgments .....	30
110	Appendix B. Notices.....	31
111	Appendix C. Revision History.....	32

---

# 1 Introduction

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

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

## 1.1 Protocol Binding and Profile Concepts

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

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

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

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

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

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

## 1.2 Notation

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

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

141 `Example code listings appear like this.`

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

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

- 146 • The prefix `saml`: stands for the SAML assertion namespace [**SAMLCore**].
- 147 • The prefix `samlp`: stands for the SAML request-response protocol namespace [**SAMLCore**].
- 148 • The prefix `ds`: stands for the W3C XML Signature namespace,  
149 `http://www.w3.org/2000/09/xmldsig#` [**XMLSig**].
- 150 • The prefix `SOAP-ENV`: stands for the SOAP 1.1 namespace,  
151 `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](mailto: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 `<StatusCode>` 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         <StatusCode value="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*.

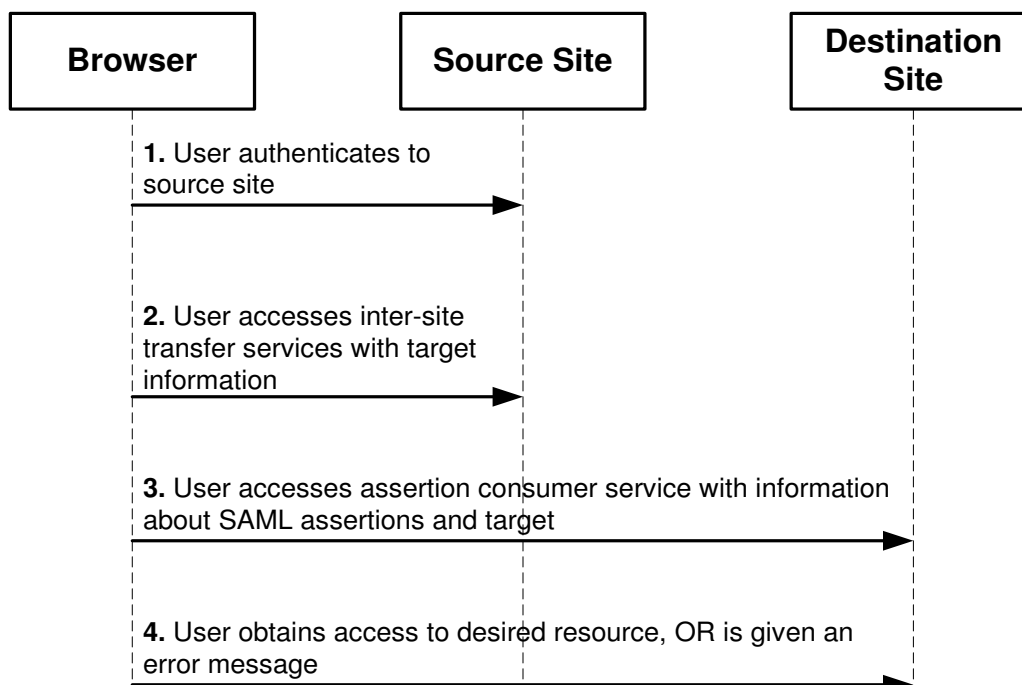
381 The following assumptions are made about this scenario for the purposes of these profiles:

- 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](mailto: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.

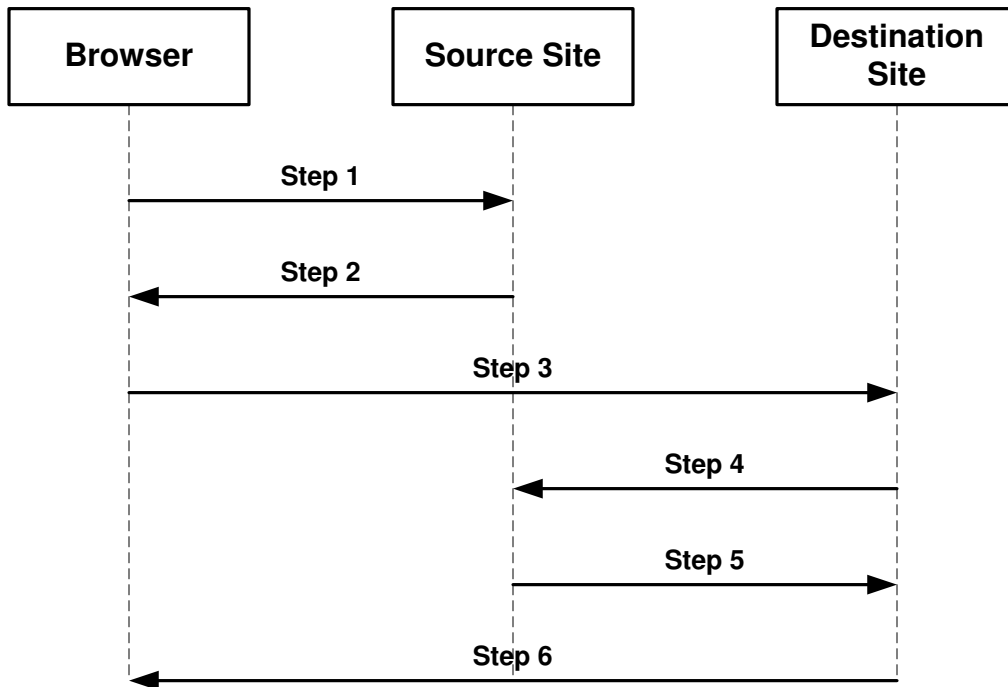
429 **Updates:** None.

#### 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, with information about the desired  
457 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 https://<inter-site transfer host name and path>?TARGET=<Target>...<HTTP-  
461 Version>  
462 <other HTTP 1.0 or 1.1 components>
```

463 Where:

464 <inter-site transfer host name and path>

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

467 Target=<Target>

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

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

#### 471 4.1.1.4 Step 2: Redirecting to the Destination Site

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

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

477 The HTTP response MUST take the following form:

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

482 Where:

483 <artifact receiver host name and path>

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

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

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

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

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

### 4.1.1.5 Step 3: Accessing the Artifact Receiver URL

In step 3, the user's browser accesses the artifact receiver URL, with a SAML artifact representing the user's authentication information attached to the URL.

The HTTP request MUST take the form:

```
GET https://<artifact receiver host name and path>?<SAML searchpart> <HTTP-  
Version>  
<other HTTP 1.0 or 1.1 request 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 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.

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.

### 4.1.1.6 Steps 4 and 5: Acquiring the Corresponding Assertions

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 SSO assertion that corresponds to each artifact.

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.

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

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.

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.

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.

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.

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 destination site Y, where X <> Y. One way to implement this feature is to have source sites maintain a list

545 of artifact and destination site pairs. The `<samlp:Status>` element of the `<samlp:Response>` MUST  
546 include a `<samlp:StatusCode>` element with the value `Success`.

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

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

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

551 • The `<saml:ConfirmationMethod>` element MUST be set to either  
552 `urn:oasis:names:tc:SAML:1.0:cm:artifact-01` (deprecated) or `urn:oasis:names:tc:SAML:1.0:cm:artifact`  
553 (RECOMMENDED).

554 • The `<SubjectConfirmationData>` element SHOULD NOT be specified.

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

#### 557 4.1.1.7 Step 6: Responding to the User's Request for a Resource

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

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

#### 562 4.1.1.8 Artifact Format

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

```
564 SAML_artifact      := B64(TypeCode RemainingArtifact)  
565 TypeCode           := Byte1Byte2
```

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

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

```
573 TypeCode           := 0x0001  
574 RemainingArtifact := SourceID AssertionHandle  
575 SourceID           := 20-byte_sequence  
576 AssertionHandle   := 20-byte_sequence
```

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

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

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

588 • Each source site selects a single identification URL. The domain name used within this URL is  
589 registered with an appropriate authority and administered by the source site.

- 590 • The source site constructs the `SourceID` component of the artifact by taking the SHA-1 hash of the  
591 identification URL.
- 592 • The `AssertionHandle` value is constructed from a cryptographically strong random or  
593 pseudorandom number sequence [RFC1750] generated by the source site. The sequence consists of  
594 values of at least eight bytes in size. These values should be padded to a total length of 20 bytes.

#### 595 4.1.1.9 Threat Model and Countermeasures

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

##### 597 4.1.1.9.1 Stolen Artifact

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

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

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

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

##### 630 4.1.1.9.2 Attacks on the SAML Protocol Message Exchange

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

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

### 635 **4.1.1.9.3 Malicious Destination Site**

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

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

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

### 647 **4.1.1.9.4 Forged SAML Artifact**

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

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

### 655 **4.1.1.9.5 Browser State Exposure**

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

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

## 663 **4.1.2 Browser/POST Profile of SAML**

### 664 **4.1.2.1 Required Information**

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

666 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

667 **SAML Confirmation Method Identifiers:** The “Bearer” confirmation method identifier is used by this  
668 profile. The following identifier has been assigned to this confirmation method:

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

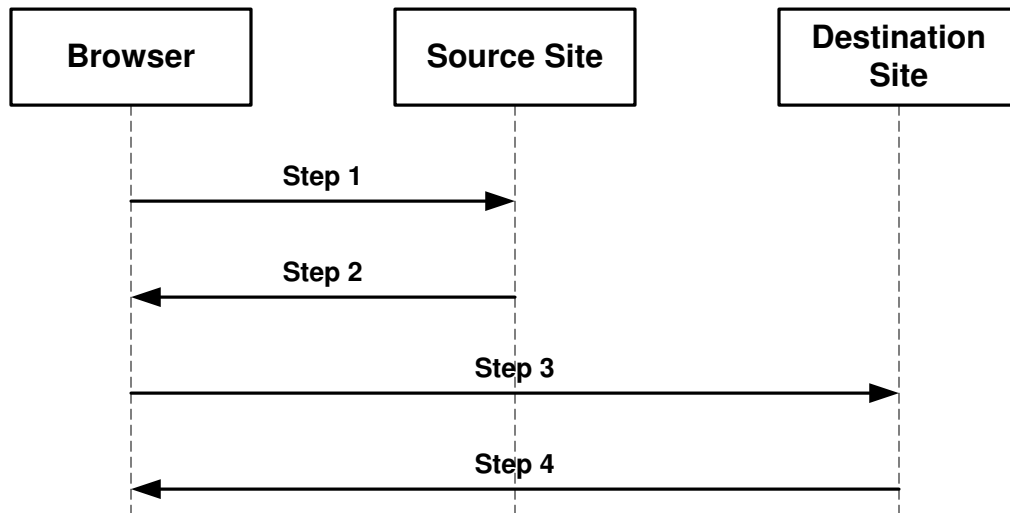
670 **Description:** Given below.

671 **Updates:** None.

### 672 **4.1.2.2 Preliminaries**

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

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



679

#### 680 4.1.2.3 Step 1: Accessing the Inter-Site Transfer Service

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

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

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

688 Where:

689 <inter-site transfer host name and path>

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

692 Target=<Target>

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

#### 695 4.1.2.4 Step 2: Generating and Supplying the Response

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

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

700 The HTTP response MUST take the form:

```
701 <HTTP-Version 200 <Reason Phrase>  
702 <other HTTP 1.0 or 1.1 components>
```

703 Where:

704 <other HTTP 1.0 or 1.1 components>

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

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

712 <assertion consumer host name and path>

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

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

719 The notation B64(<response>) stands for the result of applying the Base64 Content-Transfer-Encoding  
720 to the response, as defined by [RFC2045] §6.8, and SHOULD consist of lines of encoded data of up to  
721 76 characters. The first encoded line begins after the opening quote signifying the "value" attribute of the  
722 SAMLResponse form element.

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

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

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

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

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

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

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

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

```
741 <HTML>
742 <BODY Onload="document.forms[0].submit()">
743 <FORM METHOD="POST" ACTION="<assertion consumer host name and
744 path>">
745 ...
746 <INPUT TYPE="HIDDEN" NAME="SAMLResponse"
747 VALUE="response in base64 coding">
748 <INPUT TYPE="hidden" NAME="TARGET" Value="<Target>">
749 </FORM>
750 </BODY>
751 </HTML>
```

752 The HTTP request MUST include the following components:

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

755 Where:

756 <other HTTP 1.0 or 1.1 request components>

757 This consists of the form data set derived by the browser processing of the form data received in step  
758 2 according to § 17.13.3 of [HTML401]. Exactly one SAML response MUST be included within the  
759 form data set with control name `SAMLResponse`; multiple SAML assertions MAY be included in the  
760 response. A single target description MUST be included with the control name set to `TARGET`.

761 The SAML response MUST include the `Recipient` attribute [SAMLCore] with its value set to  
762 <assertion consumer host name and path>. At least one of the SAML assertions included within  
763 the response MUST be an SSO assertion.

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

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

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

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

#### 779 4.1.2.6 Step 4: Responding to the User’s Request for a Resource

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

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

#### 784 4.1.2.7 Threat Model and Countermeasures

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

##### 786 4.1.2.7.1 Stolen Assertion

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

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

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

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

#### 811 4.1.2.7.2 MITM Attack

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

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

#### 818 4.1.2.7.3 Forged Assertion

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

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

#### 823 4.1.2.7.4 Browser State Exposure

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

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

---

## 831 5 Confirmation Method Identifiers

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

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

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

### 850 5.1 Holder of Key

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

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

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

### 856 5.2 Sender Vouches

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

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

### 860 5.3 SAML Artifact

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

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

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

### 865 5.4 Bearer

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

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

---

## 868 **6 Use of SSL 3.0 or TLS 1.0**

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

### 872 **6.1 SAML SOAP Binding**

873 TLS-capable implementations MUST implement the TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA cipher  
874 suite and MAY implement the TLS\_RSA\_AES\_128\_CBC\_SHA cipher suite [**AES**].

### 875 **6.2 Web Browser Profiles of SAML**

876 SSL-capable implementations of the browser/artifact profile or browser/POST profile of SAML MUST  
877 implement the SSL\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA cipher suite.

878 TLS-capable implementations MUST implement the TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA cipher  
879 suite.

---

## 880 7 Alternative SAML Artifact Format

### 881 7.1 Required Information

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

883 **Contact information:** [security-services-comment@lists.oasis-open.org](mailto:security-services-comment@lists.oasis-open.org)

884 **Description:** Given below.

885 **Updates:** None.

### 886 7.2 Format Details

887 An alternative artifact format is described here:

888	TypeCode	:= 0x0002
889	RemainingArtifact	:= AssertionHandle SourceLocation
890	AssertionHandle	:= 20-byte_sequence
891	SourceLocation	:= URI

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

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

---

## 900 8 URL Size Restriction (Non-Normative)

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

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

904 The information in this article applies to:

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

906 SUMMARY

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

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

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

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

916 REFERENCES

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

919 Last Reviewed: 9/13/2001

920 Keywords: kbDSupport kbFAQ kbinfo KB208427

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

922 Issue: 19971110-3 Product: Enterprise Server

923 Created: 11/10/1997 Version: 2.01

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

925 Does this article answer your question?

926 Please let us know!

927 Question:

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

930 Answer:

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

932

933

## 9 References

- 934 [AES] FIPS-197, Advanced Encryption Standard (AES), available from  
935 <http://www.nist.gov/>.
- 936 [Anders] A suggestion on how to implement SAML browser bindings without using  
937 "Artifacts", <http://www.x-obi.com/OBI400/andersr-browser-artifact.ppt>.
- 938 [CoreAssnEx] Core Assertions Architecture, Examples and Explanations, <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-phill-07.pdf>.  
939
- 940 [HTML401] HTML 4.01 Specification, W3C Recommendation 24 December 1999,  
941 <http://www.w3.org/TR/html4>.
- 942 [Liberty] The Liberty Alliance Project, <http://www.projectliberty.org>.
- 943 [MSURL] Microsoft technical support article,  
944 <http://support.microsoft.com/support/kb/articles/Q208/4/27.ASP>.
- 945 [Rescorla-Sec] E. Rescorla et al., *Guidelines for Writing RFC Text on Security Considerations*,  
946 <http://www.ietf.org/internet-drafts/draft-iab-sec-cons-03.txt>.
- 947 [RFC1738] Uniform Resource Locators (URL), <http://www.ietf.org/rfc/rfc1738.txt>
- 948 [RFC1750] Randomness Recommendations for Security. <http://www.ietf.org/rfc/rfc1750.txt>
- 949 [RFC1945] Hypertext Transfer Protocol -- HTTP/1.0, <http://www.ietf.org/rfc/rfc1945.txt>.
- 950 [RFC2045] Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet  
951 Message Bodies, <http://www.ietf.org/rfc/rfc2045.txt>
- 952 [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, IETF  
953 RFC 2119, March 1997, <http://www.ietf.org/rfc/rfc2119.txt>.
- 954 [RFC2246] The TLS Protocol Version 1.0, <http://www.ietf.org/rfc/rfc2246.txt>.
- 955 [RFC2279] UTF-8, a transformation format of ISO 10646, <http://www.ietf.org/rfc/rfc2279.txt>.
- 956 [RFC2616] Hypertext Transfer Protocol -- HTTP/1.1, <http://www.ietf.org/rfc/rfc2616.txt>.
- 957 [RFC2617] *HTTP Authentication: Basic and Digest Access Authentication*, IETF RFC 2617,  
958 <http://www.ietf.org/rfc/rfc2617.txt>.
- 959 [SAMLCore] Phillip Hallam-Baker et al., *Assertions and Protocol for the OASIS Security  
960 Assertion Markup Language (SAML)*, OASIS, November 2002, <http://www.oasis-open.org/committees/security/>.  
961
- 962 [SAMLGloss] Jeff Hodges et al., *Glossary for the OASIS Security Assertion Markup Language  
963 (SAML)*, OASIS, November 2002, <http://www.oasis-open.org/committees/security/>.  
964
- 965 [SAMLSec] Chris McLaren et al., *Security Considerations for the OASIS Security Assertion  
966 Markup Language (SAML)*, OASIS, November 2002, <http://www.oasis-open.org/committees/security/>.  
967
- 968 [SAMLReqs] Darren Platt et al., *SAML Requirements and Use Cases*, OASIS, April 2002,  
969 <http://www.oasis-open.org/committees/security/>.
- 970 [SAMLWeb] OASIS Security Services Technical Committee website, <http://www.oasis-open.org/committees/security>.  
971
- 972 [SESSION] RL "Bob" Morgan, Support of target web server sessions in Shibboleth,  
973 <http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-session-00.txt>  
974
- 975 [ShibMarlena] Marlena Erdos, Shibboleth Architecture DRAFT v1.1,  
976 <http://shibboleth.internet2.edu/draft-internet2-shibboleth-arch-v05.html> .
- 977 [SOAP1.1] D. Box et al., *Simple Object Access Protocol (SOAP) 1.1*, World Wide Web  
978 Consortium Note, May 2000, <http://www.w3.org/TR/SOAP>.
- 979 [SSL3] A. Frier et al., *The SSL 3.0 Protocol*, Netscape Communications Corp, November  
980 1996.

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

---

## 988 **Appendix A. Acknowledgments**

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

- 991 • Irving Reid, Baltimore Technologies
- 992 • Hal Lockhart, BEA Systems
- 993 • Ronald Jacobson, Computer Associates
- 994 • John Hughes, Entegriety Solutions
- 995 • Carlisle Adams, Entrust
- 996 • Robert Griffin, Entrust
- 997 • Scott Cantor, Individual
- 998 • Bob Morgan, Individual
- 999 • Clifford Thompson, Individual
- 1000 • Pdraig Moloney, NASA
- 1001 • Prateek Mishra, Netegrity (co-chair)
- 1002 • Frederick Hirsch, Nokia
- 1003 • Senthil Sengodan, Nokia
- 1004 • Timo Skytta, Nokia
- 1005 • Charles Knouse, Oblix
- 1006 • Steve Anderson, OpenNetwork
- 1007 • Simon Godik, OverXeer
- 1008 • Rob Philpott, RSA Security (co-chair)
- 1009 • Dipak Chopra, SAP
- 1010 • Jahan Moreh, Sigaba
- 1011 • Bhavna Bhatnagar, Sun Microsystems
- 1012 • Jeff Hodges, Sun Microsystems
- 1013 • Eve Maler, Sun Microsystems (coordinating editor)
- 1014 • Emily Xu, Sun Microsystems
- 1015 • Phillip Hallam-Baker, VeriSign

1016

---

## Appendix B. Notices

1017 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that  
1018 might be claimed to pertain to the implementation or use of the technology described in this document or  
1019 the extent to which any license under such rights might or might not be available; neither does it  
1020 represent that it has made any effort to identify any such rights. Information on OASIS's procedures with  
1021 respect to rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights  
1022 made available for publication and any assurances of licenses to be made available, or the result of an  
1023 attempt made to obtain a general license or permission for the use of such proprietary rights by  
1024 implementors or users of this specification, can be obtained from the OASIS Executive Director.

1025 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications,  
1026 or other proprietary rights which may cover technology that may be required to implement this  
1027 specification. Please address the information to the OASIS Executive Director.

1028 **Copyright © OASIS Open 2003. All Rights Reserved.**

1029 This document and translations of it may be copied and furnished to others, and derivative works that  
1030 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published  
1031 and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice  
1032 and this paragraph are included on all such copies and derivative works. However, this document itself  
1033 may not be modified in any way, such as by removing the copyright notice or references to OASIS,  
1034 except as needed for the purpose of developing OASIS specifications, in which case the procedures for  
1035 copyrights defined in the OASIS Intellectual Property Rights document must be followed, or as required to  
1036 translate it into languages other than English.

1037 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors  
1038 or assigns.

1039 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
1040 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY  
1041 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR  
1042 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

1043 JavaScript is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and  
1044 other countries.

## Appendix C. Revision History

Draft	Who	What
01	Eve Maler	Cosmetic changes to bring spec up to 1.1 WD status. Fixed E1 (section cross-ref fixes), E2 (extra backslash), E3 (section formatting), PE1 (https in examples). Editorial cleanup of status code treatment.
02	Eve Maler	Copyedits throughout. Fixed PE4 (URI encoding in alternative artifact format), PE15 (browser POST profile encoding). Made into a candidate last-call working draft.
03	Eve Maler, Rob Philpott	Incorporated changes based on decisions made in the 29 April 2003 meeting. Fixed PE17 (StatusCode as the only child of Response vs. of the SOAP body), PE6 (cm:artifact URI vs. cm:artifact-01, plus additional tweaking of prose), E9 (incorrect identifier for alternative SAML artifact format), PE19 (clarification of status code when no assertion returned), PE20 (fixed browser/POST ConfirmationData). Mentioned WSS and Liberty profiles. Fixed [ShibMarlena] URL.
04	Rob Philpott	Last Call Draft – accept all changes.
05	Prateek Mishra, Rob Philpott	Clarified response when a source site artifact lifetime has been exceeded. Resorted contributor list by organization.
06	Rob Philpott	Finalized Last Call draft – accepted all changes.