



1

2 Bindings for the OASIS Security 3 Assertion Markup Language (SAML) 4 V2.0

5 OASIS Draft, ~~279~~ March~~April~~ 2004

6 **Document identifier:**

7 [sstc-saml-bindings-2.0-draft-098](#)

8 **Location:**

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

10 **Editors:**

11 Frederick Hirsch, Nokia

12 **Contributors:**

13 TBD
14 Scott Cantor, Individual
15 Frederick Hirsch, Nokia
16 John Kemp, Nokia

17 **Abstract:**

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

20 **Status:**

21 This is a Draft.

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

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

32 Table of Contents

33	1 Introduction.....	5
34	1.1 Protocol Binding Concepts.....	5
35	1.2 Notation.....	5
36	2 Specification of Additional Protocol Bindings.....	6
37	2.1 Guidelines for Specifying Protocol Bindings.....	6
38	2.2 Process Framework for Describing and Registering Protocol Bindings.....	6
39	3 Protocol Bindings.....	7
40	3.1 General Considerations.....	7
41	3.1.1 Use of SSL 3.0 or TLS 1.0.....	7
42	3.2 SAML SOAP Binding.....	7
43	3.2.1 Required Information.....	7
44	3.2.2 Protocol-Independent Aspects of the SAML SOAP Binding.....	8
45	3.2.2.1 Basic Operation.....	8
46	3.2.2.2 SOAP Headers.....	8
47	3.2.2.3 Authentication.....	9
48	3.2.2.4 Message Integrity.....	9
49	3.2.2.5 Confidentiality.....	9
50	3.2.3 Use of SOAP over HTTP.....	9
51	3.2.3.1 HTTP Headers.....	9
52	3.2.3.2 Authentication.....	9
53	3.2.3.3 Message Integrity.....	10
54	3.2.3.4 Message Confidentiality.....	10
55	3.2.3.5 Security Considerations.....	10
56	3.2.3.6 Error Reporting.....	10
57	3.2.3.7 Metadata Considerations.....	10
58	3.2.3.8 Example SAML Message Exchange Using SOAP over HTTP.....	10
59	3.3 Reverse SOAP (PAOS) Binding.....	11
60	3.3.1 Required Information.....	11
61	3.3.2 Overview.....	11
62	3.3.3 HTTP Headers.....	12
63	3.4 HTTP Redirect/POST Binding.....	12
64	3.4.1 Required Information.....	12
65	3.4.2 Overview.....	12
66	3.4.3 Message Encoding.....	12
67	3.4.3.1 RelayState.....	13
68	3.4.3.2 URL Encodings.....	13
69	3.4.3.3 Form Encoding.....	15

70	3.4.4 Message Exchange.....	16
71	3.4.4.1 HTTP Considerations.....	16
72	3.4.4.2 Authentication.....	16
73	3.4.4.3 Message Integrity.....	17
74	3.4.4.4 Confidentiality.....	17
75	3.4.5 Security Considerations.....	17
76	3.4.6 Error Reporting.....	17
77	3.4.7 Metadata Considerations.....	17
78	3.4.8 Example SAML Message Exchange Using HTTP-Redirect/POST.....	17
79	3.5 HTTP Artifact Binding.....	18
80	3.5.1 Required Information.....	18
81	3.5.2 Overview.....	18
82	3.5.3 Message Encoding.....	18
83	3.5.3.1 RelayState.....	18
84	3.5.3.2 URL Encoding.....	19
85	3.5.3.3 Form Encoding.....	19
86	3.5.4 Artifact Types.....	19
87	3.5.5 Message Exchange.....	19
88	3.5.5.1 HTTP Considerations.....	20
89	3.5.5.2 Authentication.....	21
90	3.5.5.3 Message Integrity.....	21
91	3.5.5.4 Confidentiality.....	21
92	3.5.6 Security Considerations.....	21
93	3.5.7 Error Reporting.....	21
94	3.5.8 Metadata Considerations.....	21
95	3.5.9 Example SAML Message Exchange Using HTTP Artifact.....	22
96	3.6 SAML URI Binding.....	22
97	3.6.1 Required Information.....	22
98	3.6.2 Protocol-Independent Aspects of the SAML URI Binding.....	22
99	3.6.2.1 Basic Operation.....	22
100	3.6.2.2 Authentication.....	22
101	3.6.2.3 Message Integrity.....	22
102	3.6.2.4 Confidentiality.....	23
103	3.6.3 General Security Considerations.....	23
104	3.6.4 MIME Encapsulation.....	23
105	3.6.5 Use of HTTP URIs.....	23
106	3.6.5.1 URI Syntax.....	23
107	3.6.5.2 HTTP Headers.....	23
108	3.6.5.3 Authentication.....	24

109	3.6.5.4 Message Integrity.....	24
110	3.6.5.5 Message Confidentiality.....	24
111	3.6.5.6 Security Considerations.....	24
112	3.6.5.7 Error Reporting.....	24
113	3.6.5.8 Metadata Considerations.....	24
114	3.6.5.9 Example SAML Message Exchange Using an HTTP URI.....	25
115	4 SAML Artifact Formats.....	26
116	4.1 SAML 1.x Format.....	26
117	4.1.1 Required Information.....	26
118	4.1.2 Format Details.....	26
119	4.2 SAML 1.x Supplemental Format.....	27
120	4.2.1 Required Information.....	27
121	4.2.2 Format Details.....	27
122	5 URL Size Restriction (Non-Normative).....	28
123	6 References.....	29

1 Introduction

124

125 | This document specifies [SAML](#) protocol bindings for the use of SAML assertions and request-response
126 | messages in communications protocols and frameworks.

127 | [SAMLCore] defines the SAML assertions and request-response messages themselves, and[SAMLProfile]
128 | defines specific usage patterns that reference both [SAMLCore] and bindings defined in this specification
129 | or elsewhere.

1.1 Protocol Binding Concepts

130

131 | Mappings of SAML request-response message exchanges onto standard messaging or communication
132 | protocols are called SAML *protocol bindings* (or just *bindings*). An instance of mapping SAML request-
133 | response message exchanges into a specific [communication](#) protocol <FOO> is termed a <FOO> *binding*
134 | for SAML or a SAML <FOO> *binding*.

135 | For example, a SAML SOAP binding describes how SAML request and response message exchanges
136 | are mapped into SOAP message exchanges.

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

139 | Unless otherwise specified, a binding should be understood to support the transmission of any SAML
140 | protocol message derived from the **samlp:RequestAbstractType** and **samlp:StatusResponseType**
141 | types. Further, when a binding refers to "SAML requests and responses", it should be understood to mean
142 | any protocol messages derived from those types.

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

1.2 Notation

144

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

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

149 | `Example code listings appear like this.`

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

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

- 153 | • The prefix saml: stands for the SAML assertion namespace [SAMLCore].
- 154 | • The prefix samlp: stands for the SAML request-response protocol namespace [SAMLCore].
- 155 | • The prefix ds: stands for the W3C XML Signature namespace, <http://www.w3.org/2000/09/xmldsig#>
156 | [XMLSig].
- 157 | • The prefix SOAP-ENV: stands for the SOAP 1.1 namespace,
158 | <http://schemas.xmlsoap.org/soap/envelope> [SOAP1.1].

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

2 Specification of Additional Protocol Bindings

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

2.1 Guidelines for Specifying Protocol Bindings

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

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

2.2 Process Framework for Describing and Registering Protocol Bindings

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

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

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

201 3 Protocol Bindings

202 The following sections define SAML protocol bindings sanctioned by the OASIS Security Services
203 Technical Committee. These include the SAML SOAP binding and the SAML reverse SOAP (PAOS)
204 bindings, as well as HTTP bindings that involve a user agent intermediary and URI dereferencing of
205 assertions

206 3.1 General Considerations

207 3.1.1 Use of SSL 3.0 or TLS 1.0

208 Unless otherwise specified, in any SAML binding's use of SSL 3.0 [SSL3] or TLS 1.0 [RFC2246], servers
209 MUST authenticate to clients using a X.509 v3 certificate. The client MUST establish server identity based
210 on contents of the certificate (typically through examination of the certificate's subject DN field).

211 TLS-capable implementations MUST implement the TLS_RSA_WITH_3DES_EDE_CBC_SHA cipher
212 suite and MAY implement the TLS_RSA_WITH_AES_128_CBC_SHA cipher suite [AES].

213 FIPS TLS-capable implementations MUST implement the corresponding
214 TLS_RSA_FIPS_WITH_3DES_EDE_CBC_SHA cipher suite and MAY implement the corresponding
215 TLS_RSA_FIPS_AES_128_CBC_SHA cipher suite [AES] [FIPS].

216 SSL-capable implementations MUST implement the SSL_RSA_WITH_3DES_EDE_CBC_SHA cipher
217 suite.

218 FIPS SSL-capable implementations MUST implement the FIPS ciphersuite corresponding to the SSL
219 SSL_RSA_WITH_3DES_EDE_CBC_SHA cipher suite [FIPS].

220 3.2 SAML SOAP Binding

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

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

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

229 3.2.1 Required Information

230 **Identification:** urn:oasis:names:tc:SAML:2.0:bindings:SOAP-binding

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

232 **Description:** Given below.

233 **Updates:** urn:oasis:names:tc:SAML:1.0:bindings:SOAP-binding

234 3.2.2 Protocol-Independent Aspects of the SAML SOAP Binding

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

237 3.2.2.1 Basic Operation

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

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

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

- 244 1. A system entity acting as a SAML requester transmits a SAML request element within the body of
245 a SOAP message to a system entity acting as a SAML responder. The SAML requester MUST
246 NOT include more than one SAML request per SOAP message or include any additional XML
247 elements in the SOAP body.
- 248 2. The SAML responder MUST return either a SAML response element within the body of another
249 SOAP message or generate a SOAP fault. The SAML responder MUST NOT include more than
250 one SAML response per SOAP message or include any additional XML elements in the SOAP
251 body. If a SAML responder cannot, for some reason, process a SAML request, it MUST generate a
252 SOAP fault. SOAP fault codes MUST NOT be sent for errors within the SAML problem domain, for
253 example, inability to find an extension schema or as a signal that the subject is not authorized to
254 access a resource in an authorization query. (SOAP 1.1 faults and fault codes are discussed in
255 [SOAP1.1] §4.1.)

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

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

264 3.2.2.2 SOAP Headers

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

267 **Note:** The reason other headers need to be allowed is that some SOAP software and
268 libraries might add headers to a SOAP message that are out of the control of the SAML-
269 aware process. Also, some headers might be needed for underlying protocols that require
270 routing of messages or by message security mechanisms

271 A SAML responder MUST NOT require any headers in the SOAP message to correctly process the SAML
272 message itself, but MAY require additional headers that address underlying routing or message security
273 requirements.

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

276 3.2.2.3 Authentication

277 Authentication of both the SAML requester and the SAML responder is OPTIONAL and depends on the
278 environment of use. Authentication mechanisms available from the underlying substrate protocol MAY be
279 utilized to provide authentication. Section 3.2.3.2 describes authentication in the SOAP over HTTP
280 environment. Authentication mechanisms designed specifically for SOAP message exchange MAY also
281 be utilized.

282 3.2.2.4 Message Integrity

283 Message integrity of both SAML requests and SAML responses is OPTIONAL and depends on the
284 environment of use. The security layer in the underlying substrate protocol MAY be used to ensure
285 message integrity. Section 3.2.3.3 describes support for message integrity in the SOAP over HTTP
286 environment. Integrity mechanisms designed specifically for SOAP message exchange MAY also be
287 utilized.

288 3.2.2.5 Confidentiality

289 Confidentiality of both SAML requests and SAML responses is OPTIONAL and depends on the
290 environment of use. The security layer in the underlying substrate protocol MAY be used to ensure
291 message confidentiality. Section 3.2.3.4 describes support for confidentiality in the SOAP over HTTP
292 environment. Confidentiality mechanisms designed specifically for SOAP message exchange MAY also
293 be utilized.

294 3.2.3 Use of SOAP over HTTP

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

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

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

302 3.2.3.1 HTTP Headers

303 HTTP proxies MUST NOT cache responses carrying SAML assertions.

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

- 305 1. If the value of the `Cache-Control` header field is **not** set to `no-store`, then the SAML responder
306 MUST NOT include the `Cache-Control` header field in the response.
- 307 2. If the `Expires` response header field is **not** disabled by a `Cache-Control` header field with a
308 value of `no-store`, then the `Expires` field SHOULD NOT be included.

309 There are no other restrictions on HTTP headers.

310 3.2.3.2 Authentication

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

- 312 1. No client or server authentication.
- 313 2. HTTP basic client authentication [RFC2617] with and without SSL 3.0 or TLS 1.0.
- 314 3. HTTP over SSL 3.0 or TLS 1.0 (see Section 3.1.1) server authentication with a server-side

315 certificate.
316 4. HTTP over SSL 3.0 or TLS 1.0 mutual authentication with both server-side and a client-side
317 certificate.

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

319 **3.2.3.3 Message Integrity**

320 When message integrity needs to be guaranteed, SAML responders MUST implement HTTP over SSL
321 3.0 or TLS 1.0 (see Section 3.1.1) with a server-side certificate.

322 **3.2.3.4 Message Confidentiality**

323 When message confidentiality is required, SAML responders MUST implement HTTP over SSL 3.0 or
324 TLS 1.0 (see Section 3.1.1) with a server-side certificate.

325 **3.2.3.5 Security Considerations**

326 Before deployment, each combination of authentication, message integrity, and confidentiality
327 mechanisms SHOULD be analyzed for vulnerability in the context of the specific protocol exchange, and
328 the deployment environment. See specific protocol processing rules in [SAMLCore], and the SAML
329 security considerations document [SAMLSec] for a detailed discussion.

330 [RFC2617] describes possible attacks in the HTTP environment when basic or message-digest
331 authentication schemes are used.

332 **3.2.3.6 Error Reporting**

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

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

341 In the case of a SAML processing error, the SOAP HTTP server MUST respond with "200 OK" and
342 include a SAML-specified <Status> element in the SAML response within the SOAP body.

343 For more information about SAML status codes, see the SAML assertion and protocol specification
344 [SAMLCore].

345 **3.2.3.7 Metadata Considerations**

346 Support for the SOAP binding SHOULD be reflected by indicating either a URL endpoint at which requests
347 contained in SOAP messages for a particular protocol or profile are to be sent, or alternatively with a
348 WSDL port/endpoint definition.

349 **3.2.3.8 Example SAML Message Exchange Using SOAP over HTTP**

350 Following is an example of a query that asks for an assertion containing an attribute statement from a
351 SAML attribute authority.

352 `POST /SamlService HTTP/1.1`

```

353 Host: www.example.com
354 Content-Type: text/xml
355 Content-Length: nnn
356 SOAPAction: http://www.oasis-open.org/committees/security
357 <SOAP-ENV:Envelope
358   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
359   <SOAP-ENV:Body>
360     <samlp:AttributeQuery xmlns:samlp="..."
361   xmlns:saml="..." xmlns:ds="..." RequestID='6c3a4f8b9c2d' MajorVersion='2'
362   MinorVersion='0' IssueInstant='
363     <ds:Signature> ... </ds:Signature>
364     <saml:Subject>
365     ...
366     </saml:Subject>
367   </samlp:AttributeQuery>
368 </SOAP-ENV:Body>
369 </SOAP-ENV:Envelope>

```

370 Following is an example of the corresponding response, which supplies an assertion containing the
371 attribute statement as requested.

```

372 HTTP/1.1 200 OK
373 Content-Type: text/xml
374 Content-Length: nnnn
375 <SOAP-ENV:Envelope
376   xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
377   <SOAP-ENV:Body>
378     <samlp:Response xmlns:samlp="..." xmlns:saml="..." xmlns:ds="..."
379   ResponseID='6c3a4f8b9c2d' MajorVersion='2' MinorVersion='0'
380   IssueInstant='2004-03-27T08:42:00Z'>
381     <ds:Signature> ... </ds:Signature>
382     <Status>
383       <StatusCodevalue="samlp:Success"/>
384     </Status>
385
386     <saml:Assertion>
387       <saml:AttributeStatement>
388       ...
389       </saml:AttributeStatement>
390     </saml:Assertion>
391   </samlp:Response>
392 </SOAP-Env:Body>
393 </SOAP-ENV:Envelope>

```

394 **3.3 Reverse SOAP (PAOS) Binding**

395 This binding leverages the Reverse HTTP Binding for SOAP specification [PAOS]. Implementers MUST
396 comply with the general processing rules specified in [PAOS] in addition to those specified in this
397 document. In case of conflict, [PAOS] is normative.

398 **3.3.1 Required Information**

399 **Identification:** urn:oasis:names:tc:SAML:2.0:bindings:paos-binding

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

401 **Description:** Given below.

402 **Updates:** None.

403 **3.3.2 Overview**

404 [The reverse SOAP binding is a mechanism by which an HTTP requester can advertise the ability to act as](#)

405 | [a SOAP intermediary to an HTTP responder and handle a SAML protocol message from that responder in](#)
406 | [a SOAP envelope returned in the body of an HTTP response. The HTTP requester may or may not be the](#)
407 | [intended recipient of the SAML protocol message.](#)

408 | 3.3.3 HTTP Headers

409 | ~~To indicate support for this binding, the HTTP requester includes The HTTP request from the client to the~~
410 | ~~service provider is an HTTP GET request with~~ additional HTTP headers ~~indicating the ability to adhere to~~
411 | ~~the reverse SOAP binding in its request(s).~~ Specifically, the HTTP request meets the requirements
412 | outlined in the PAOS specification [PAOS] and this document as follows:

- 413 | 1. The HTTP Accept Header field SHOULD indicate [an](#) ability to accept [the](#)
414 | "application/vnd.paos+xml" [content type](#).
- 415 | 2. The HTTP PAOS Header field SHOULD be present and specify the PAOS version with
416 | urn:liberty:paos:2003-08 at a minimum.

417 | Additional PAOS headers such as the service value are specified by profiles that use the PAOS binding.

418 | 3.4 HTTP Redirect/POST Binding

419 | Naming issue

420 | Sometimes referred to as the "front-channel", the HTTP Redirect/POST binding defines a mechanism by
421 | which SAML protocol messages may be transmitted either within URL parameters or within the base64-
422 | encoded content of an HTML form control. Permissible URL length is infinite in the abstract, but quite
423 | unpredictably limited in actual practice ~~across user agents~~. Therefore, specialized encodings must be
424 | used to carry XML messages on a URL, and larger or more complex message content [is better should be](#)
425 | sent using the POST mechanism.

426 | 3.4.1 Required Information

427 | **Identification:** urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect-[POST](#)

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

429 | **Description:** Given below.

430 | **Updates:** Effectively replaces the binding aspects of the Browser/POST profile in [SAML 1.1].

431 | 3.4.2 Overview

432 | The HTTP-Redirect/POST binding is intended for cases in which the SAML requester and responder need
433 | to communicate using an HTTP user agent ([as defined in \[RFC2616\], HTTP 1.1](#)) as an intermediary. This
434 | may be necessary, for example, if the communicating parties do not share a direct path of communication.
435 | It may also be needed if the responder requires an interaction with the user agent in order to fulfill the
436 | request, such as when the user agent must authenticate to it.

437 | Note that some HTTP user agents may have the capacity to play a more active role in the protocol
438 | exchange and may support other bindings that use HTTP, such as the SOAP [and Reverse SOAP](#)
439 | [bindings](#). This binding ~~assumes~~ nothing apart from the capabilities of a common web browser.

440 | 3.4.3 Message Encoding

441 | There are two general ways [of](#) ~~to~~ [encoding](#) messages for use with this binding. One is to encode the
442 | message into URL parameters and the other is to encode the XML instance in base-64 and then place the
443 | result in an HTML form control. When URL encoding is used, the HTTP GET method is used to deliver

444 the message, while POST is used with form encoding.

445 All endpoints that support this binding MUST support form encoding and at least one URL encoding
446 technique. ~~Endpoint URLs MUST NOT include query string parameters independent of the URL encoding~~
447 ~~of messages.~~

448 **3.4.3.1 RelayState**

449 ~~An additional, optional piece of data called "RelayState" MAY be included with a SAML protocol message~~
450 ~~transmitted with this binding. The value MUST NOT exceed 80 bytes in length and SHOULD be integrity~~
451 ~~protected by the entity creating the message independent of any other protections that may or may not~~
452 ~~exist during message transmission.~~

453 ~~If a SAML request message is accompanied by this optional data element, then the SAML responder~~
454 ~~MUST return its SAML protocol response using a binding that also supports a "RelayState" mechanism,~~
455 ~~and it MUST place the exact value it received with the request into the corresponding "RelayState"~~
456 ~~parameter in the response.~~

457 ~~If no such value is included with a SAML request message, or if the SAML response message is being~~
458 ~~generated without a corresponding request, then the SAML responder MAY include a "RelayState" value~~
459 ~~to be interpreted by the recipient based on the use of a profile or prior agreement between the parties.~~

460 **3.4.3.2 URL Encodings**

461 There are many possible ways to encode XML into a URL, depending upon the constraints in effect. This
462 specification defines ~~such one such~~ methods without precluding others. Binding endpoints SHOULD
463 indicate which encodings they support using metadata, when appropriate. Particular encodings MUST be
464 uniquely identified with a URI when defined. It is NOT a requirement that all possible SAML messages be
465 encodable with a particular set of rules, but the rules MUST clearly indicate which messages or content
466 can or cannot be so encoded.

467 A URL encoding MUST place the message entirely within the URL query string, and MUST reserve the
468 rest of the URL for the endpoint of the message recipient.

469 **Plain URL Encoding**

470 ~~SAML protocol messages MAY be encoded by taking each specific element and attribute present in the~~
471 ~~original message, and embedding each of these as individual query parameter components of the URL.~~
472 ~~Such encoding may take place according to the rules specified for the application/x-www-form-~~
473 ~~urlencoded content type described in [HTML401]. Specifically, rules apply to the <query> component~~
474 ~~of an HTTP URL as described in [RFC2396].~~

475 ~~The original XML protocol message MUST be encoded as follows:~~

- 476 • ~~If an element or attribute is mandatory in a message, based on the specification of that message in~~
477 ~~[SAMLCore], then that element or attribute MUST appear in the URL-encoded form of the message.~~
- 478 • ~~If an element or attribute is designated as optional in the specification of that message in [SAMLCore],~~
479 ~~and does not appear in the protocol message, then the corresponding data item MUST NOT appear in~~
480 ~~the URL-encoded message.~~
- 481 • ~~Each item specified as part of the <query> component MUST be the value of the XML protocol~~
482 ~~message element or attribute.~~
- 483 • ~~Several of the elements present in SAML protocol messages may have multiple values (for example,~~
484 ~~elements with maxOccurs="unbounded" specified in the XML schema). Such elements MUST be~~
485 ~~encoded as URL-encoded, space-separated lists of values (as shown below).~~

486 ~~[TODO] <EXAMPLE>~~

487 • ~~In certain cases (such as `samlp:StatusCode` value), elements may be drawn from different XML~~
488 ~~namespaces, and qualified using an XML QName format. In such cases, values SHOULD be URL-~~
489 ~~encoded including the QName prefix that appears in the XML protocol message, with the prefixes `saml`~~
490 ~~and `samlp` defined to map to the namespaces specified in [SAMLCore] for those prefixes. Other~~
491 ~~QName prefixes MAY be defined in URL-encoded messages by including query parameters of the form~~
492 ~~`xmlns:<prefix>`, where `<prefix>` MUST be the QName prefix that appears in the protocol~~
493 ~~message for the element. Thus, `xmlns:ds` might appear in a URL-encoded message as `xmlns%`
494 ~~`3Ads=http%3A%2F%2Fwww.w3.org%2F2000%2F09%2Fxmldsig%23`. If XML QName prefixes are~~
495 ~~encoded using this method, the prefix definition MUST appear in the encoded URL before the QName~~
496 ~~value that uses the prefix.~~~~

497 • ~~Elements that may appear with the same name in nested XML structures (such as~~
498 ~~`<samlp:StatusCode>`) MUST be encoded by producing a URL-encoded, space-separated string,~~
499 ~~illustrated in the example below:~~

500 ~~[TODO] <EXAMPLE>~~

501 • ~~URL-encoded URL values SHOULD NOT exceed 80 bytes in length. Similarly, the URL-encoded value~~
502 ~~of the `<RelayState>` element SHOULD NOT exceed 80 bytes in length.~~

503 • ~~Certain XML protocol messages support arbitrary extension, via the presence of an `<Extension>`~~
504 ~~element. Messages that contain such content MUST adhere to the following additional rules:~~

505 • ~~Attribute values and elements based on complex content models MUST NOT be included in URL-~~
506 ~~encoded messages.~~

507 • ~~All attributes and elements that are included in URL-encoded form MUST have an empty namespace,~~
508 ~~and unique local names.~~

509 • ~~All values included SHOULD NOT exceed 80 bytes in length.~~

510 ~~[TODO — SEE IF THERE ARE OTHER SPECIFIC PER MESSAGE ISSUES FOR URL ENCODING]~~

511 **GZIP Encoding**~~zipped URL encoding~~

512 **Identification:** `urn:oasis:names:tc:SAML:2.0:bindings:URL-Encoding:gzip`

513 SAML protocol messages ~~MAY also can~~ be encoded into a URL via the gzip compression method (see
514 [RFC1952]). In such an encoding, the following procedure should be applied to the original SAML protocol
515 message's XML serialization:

516 i) Any signature, including the ~~containing~~ `<ds:Signature>` XML element itself, MUST be removed
517 from the SAML protocol message ~~itself~~. Note that if the content of the message includes another
518 signature, such as for example, a signed SAML assertion, this embedded signature is NOT removed.
519 However, the length of such a message after encoding essentially precludes using this mechanism.
520 Thus SAML protocol messages that contain signed content SHOULD NOT be encoded using this
521 mechanism.

522 ii) The gzip encoding, as specified in [RFC1952] ~~should be is then~~ applied to the entire XML content of the
523 original SAML protocol message, excepting any signature removed in i). The XML content MUST be
524 encoded as a single "member" of the compressed data set. The OS field of the member header
525 SHOULD be set to 255. The decoder of the data set MAY ignore the OS field and SHOULD treat the
526 result as binary data since the XML self-describes the character encoding used.

527 iii) The gzip-encoded data ~~MUST be is~~ subsequently base_64 encoded according to the rules specified in
528 [RFC2045]. Linefeeds or other whitespace MUST be removed from the result.

529 iv) The base_64 encoded, gziped data ~~should is then be~~ URL-encoded, and added to the URL as a
530 `<query>` ~~component~~ query string parameter which MUST be named `SAMLRequestMessage`. ~~(if the~~
531 ~~message is a SAML request) or `SAMLResponse` (if the message is a SAML response).~~

532 v) If the original [SAML](#) protocol message was signed using an XML digital signature, a [new](#) signature
533 covering the encoded data as specified above, MUST be [applied/attached](#) using the rules stated in this
534 specification regarding URL-encoding and digital signatures.

535 vi) [If a "RelayState" value is to accompany the SAML protocol message, it MUST be URL-encoded and](#)
536 [placed in an additional query string parameter named RelayState.](#)

537 **URL-encoding and Digital Signatures**

538 XML digital signatures are not directly URL-encoded according to the above rules, due to space concerns.
539 If the [underlying XML-SAML](#) protocol message is signed with an XML signature [XMLDSig], [any](#) URL-
540 encoded form of the message MUST be signed as follows:

541 • The signature algorithm identifier MUST be included as an additional `<query>` [component/string](#)
542 parameter, named `SigAlg`. The value of this parameter MUST be the algorithm used to sign the URL-
543 encoded [SAML protocol](#) message, specified according to [\[XMLSig\]\[XMLDSig\]](#). ~~The signature itself~~
544 ~~MUST be omitted from this parameter.~~

545 • [To construct the signature, a string consisting of the concatenation of the SigAlg and SAMLRequest](#)
546 [\(or SAMLResponse\) query string parameters is constructed as follows:](#)

547 [SAMLRequest=value&SigAlg=value](#)
548 [or](#)
549 [SAMLResponse=value&SigAlg=value](#)

550 • [The resulting string of bytes is the octet string to be fed into the signature algorithm. Any other content](#)
551 [in the original query string is not included and not signed.](#)

552 • ~~The signature should cover the encoded content of the <query> component of the URL, that is,~~
553 ~~everything after the ? character, other than the &Signature portion of the <query> component. The~~
554 ~~<query> component MUST NOT contain any other content than the URL-encoded protocol message~~
555 ~~and the Signature and SigAlg <query> component parameters.~~

556 • The signature [value](#) MUST be encoded using [a](#) [base-64](#) encoding (see [RFC2045]) [with any](#)
557 [whitespace removed](#) and included as a `<query>` [query string component](#) parameter named
558 `Signature`.

559 • ~~Note it should be noted~~ that some characters in the [base-64](#)-encoded signature [value](#) may themselves
560 require URL-encoding, before being added ~~as a <query> component parameter.~~

561 • The following signature algorithms (see [XMLDSig]) and their URI representations, MUST be
562 supported in [support of](#) this encoding [mechanism](#):

563 • DSAwithSHA1 – <http://www.w3.org/200/09/xmlsig#dsa-sha1>

564 • RSAwithSHA1 – <http://www.w3.org/200/09/xmlsig#rsa-sha1>

565 ~~[TODO — URL-encoded, signed message]~~

566 **3.4.3.3 Form Encoding**

567 A SAML protocol message is form-encoded by applying the base-64 encoding rules to the XML
568 representation of the message and placing the result in a hidden form control within a form as defined by
569 [HTML401], chapter 17. [The HTML document MUST adhere to the XHTML specification, \[XHTML\]. The](#)
570 [base-64 encoded value MAY be line-wrapped at a reasonable length in accordance with common](#)
571 [practice.](#)

572 If the message is a SAML request, then the form control MUST be named `"SAMLRequest"`. If the
573 message is a SAML response, then the form control MUST be named `SAMLResponse`. Any additional
574 form controls or presentation MAY be included but MUST NOT be required in order for the recipient to

575 process the message.

576 The action attribute of the form MUST be the recipient's HTTP endpoint for the protocol or profile using
577 this binding to which the [SAML](#) message is to be delivered. The method attribute MUST be ~~set to~~ "POST".

578 [If a "RelayState" value is to accompany the SAML protocol message, it MUST be placed in an additional](#)
579 [hidden form control named RelayState.](#)

580 Any technique supported by the user agent MAY be used to cause the submission of the form, and any
581 form content necessary to support this MAY be included, such as submit controls and client-side scripting
582 commands. However, the recipient MUST be able to process the message without regard for the
583 mechanism by which the form submission is initiated.

584 **3.4.4 Message Exchange**

585 The system model used for SAML conversations via this binding is a request-response model, but these
586 messages are sent to the user agent in an HTTP response and delivered to the message recipient in an
587 HTTP request. The HTTP interactions before, between, and after these exchanges take place is
588 undefined. Both the SAML requester and responder are assumed to be HTTP responders.

589 [TODO: sequence diagram](#)

- 590 1. A system entity acting as a SAML requester responds to an HTTP request from the user agent by
591 returning a SAML request.
- 592 2. If URL-encoded, the request is returned encoded into the HTTP response's Location header,
593 and the HTTP status MUST be either 303 or 302. The SAML requester MAY include additional
594 presentation and content in the HTTP response to facilitate the user agent's transmission of
595 the message, as defined in [HTTP 1.1](#), [RFC2616]. The user agent delivers the SAML request
596 by issuing an HTTP GET request to the SAML responder.
- 597 3. If form-encoded, then the request is returned in an [XHTML] document containing the form and
598 content defined in section [3.4.3.3-2](#). The user agent delivers the SAML request by issuing an
599 HTTP POST request to the SAML responder.
- 600 4. In general, the SAML responder MAY respond to the SAML request by immediately returning a
601 SAML response or MAY return arbitrary content to facilitate subsequent interaction with the user
602 agent necessary to fulfill the request. Specific protocols and profiles may include mechanisms to
603 indicate the requester's (un)willingness to permit this kind of interaction. Eventually the responder
604 SHOULD return a SAML response to the user agent to be returned to the SAML requester. The
605 SAML response is returned in the same fashion as described for the SAML request.

606 **3.4.4.1 HTTP Considerations**

607 HTTP proxies and the user agent intermediary MUST NOT cache SAML requests or responses. [HTTP](#)
608 [headers SHOULD be used to protect against such caching behavior.](#)

609 ~~Both of the following conditions apply when using HTTP 1.1 during message exchange:~~

- 610 1. ~~If the value of the Cache-Control header field is not set to no-store, then the HTTP responder~~
611 ~~returning a SAML request or response MUST NOT include the Cache-Control header field in the~~
612 ~~HTTP response.~~
- 613 2. ~~If the Expires HTTP response header field is not disabled by a Cache-Control header field with~~
614 ~~a value of no-store, then the Expires header SHOULD NOT be included in the HTTP response.~~

615 There are no other restrictions on the use of HTTP headers.

616 **3.4.4.2 Authentication**

617 Authentication of both the SAML requester and the SAML responder is OPTIONAL and depends on the

618 environment of use. The presence of the user agent intermediary means that the requester and responder
619 cannot rely on the transport layer for authentication, and must authenticate the messages received
620 instead. SAML provides for a signature on protocol messages for this purpose. Form-encoded messages
621 MAY be signed before the base-64 encoding is applied. URL-encoded messages MAY be signed if the
622 encoding method specifies a means for signing.

623 **3.4.4.3 Message Integrity**

624 Message integrity of both SAML requests and SAML responses is OPTIONAL and depends on the
625 environment of use. The presence of the user agent intermediary means that the requester and responder
626 cannot rely on the transport layer for integrity protection. SAML provides for a signature on protocol
627 messages for this purpose. Form-encoded messages MAY be signed before the base-64 encoding is
628 applied. URL-encoded messages MAY be signed if the encoding method specifies a means for signing.

629 **3.4.4.4 Confidentiality**

630 This binding **MUST NOT** be used if the content of the request or response cannot be exposed to the
631 user agent intermediary. Otherwise, confidentiality of both SAML requests and SAML responses is
632 OPTIONAL and depends on the environment of use. If confidentiality is necessary, SSL 3.0 or TLS 1.0
633 SHOULD be used to protect the message in transit between the user agent and the SAML requester and
634 responder.

635 **3.4.5 Security Considerations**

636 Before deployment, each combination of authentication, message integrity, and confidentiality
637 mechanisms SHOULD be analyzed for vulnerability in the context of the specific protocol exchange, and
638 the deployment environment. See specific protocol processing rules in [SAMLCore], and the SAML
639 security considerations document [SAMLSec] for a detailed discussion.

640 In general, this binding relies on message-level authentication and integrity protection via signing and
641 does not support confidentiality of messages from the user agent intermediary.

642 **3.4.6 Error Reporting**

643 A SAML responder that refuses to perform a message exchange with the SAML requester SHOULD
644 return a `<StatusResponse>` [response](#) message with a second level `<StatusCode>` value of
645 `RequestDenied`.

646 HTTP interactions during the message exchange MUST NOT use HTTP error status codes to indicate
647 failures in SAML processing, since the user agent is not a full party to the SAML protocol [exchange](#).

648 For more information about SAML status codes, see the SAML assertion and protocol specification
649 [SAMLCore].

650 **3.4.7 Metadata Considerations**

651 Support for the HTTP-Redirect/POST binding SHOULD be reflected by indicating URL endpoints at which
652 requests and responses for a particular protocol or profile should be sent. Either a single endpoint or
653 distinct request and response endpoints should be possible.

654 **3.4.8 Example SAML Message Exchange Using HTTP-Redirect/POST**

655 TBD

656 3.5 HTTP Artifact Binding

657 Similar to the Redirect/POST binding, the artifact binding provides a composable substitute in which the
658 SAML request, response, or both are transmitted by reference using a small stand-in called an artifact. A
659 separate, [synchronous](#) binding, such as the SAML SOAP binding, is used to exchange the artifact for the
660 actual protocol message using the Artifact Protocol defined in [SAMLCore]. It is composable with the
661 Redirect/POST binding in that they can easily be combined such that the request can be transmitted with
662 one binding and the response with the other because they use the same transport, use the same kind of
663 user agent intermediary, and have similar general processing rules.

664 3.5.1 Required Information

665 **Identification:** urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact

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

667 **Description:** Given below.

668 **Updates:** Effectively replaces the binding aspects of the Browser/Artifact profile in [SAML 1.1].

669 3.5.2 Overview

670 The HTTP Artifact binding is intended for cases in which the SAML requester and responder need to
671 communicate using an HTTP user agent as an intermediary, but the intermediary's limitations preclude or
672 discourage the transmission of an entire message (or message exchange) through it. This may be for
673 technical reasons or because of a reluctance to expose the message content to the intermediary (and if
674 the use of encryption is not practical).

675 Note that because of the need to subsequently dereference the artifact using another more direct binding,
676 such as SOAP, a direct communication path must exist between the SAML message sender and recipient
677 in the reverse direction of the artifact's transmission (the receiver of the message and artifact must be
678 able to send an artifact request back to the artifact sender). The message sender must also maintain state
679 while the artifact is pending, which has implications for load-balanced environments.

680 3.5.3 Message Encoding

681 There are two methods of encoding an artifact for use with this binding. One is to encode the artifact into a
682 URL parameter and the other is to place the artifact in an HTML form control. When URL encoding is
683 used, the HTTP GET method is used to deliver the message, while POST is used with form encoding.

684 All endpoints that support this binding MUST support both techniques. ~~Endpoint URLs MUST NOT include~~
685 ~~query string parameters independent of the URL encoding of messages.~~

686 3.5.3.1 RelayState

687 An additional, optional piece of data called "RelayState" MAY be included with a SAML artifact transmitted
688 with this binding. The value MUST NOT exceed 80 bytes in length and SHOULD be integrity protected by
689 the entity creating the message independent of any other protections that may or may not exist during
690 message transmission.

691 If an artifact that represents a SAML request is accompanied by this optional data element, then the SAML
692 responder MUST return its SAML protocol response using a binding that also supports a "RelayState"
693 mechanism, and it MUST place the exact value it received with the artifact into the corresponding
694 "RelayState" parameter in the response.

695 If no such value is included with an artifact representing a SAML request, or if the SAML response
696 message is being generated without a corresponding request, then the SAML responder MAY include a

697 ["RelayState" value to be interpreted by the recipient based on the use of a profile or prior agreement](#)
698 [between the parties.](#)

699 **3.5.3.2 URL Encoding**

700 To encode an artifact into a URL, the artifact value is [URL-encoded and](#) placed in a query string
701 parameter named ["SAMLart"](#). ~~If a "RelayState" value is to accompany the SAML artifact, it MUST be~~
702 [URL-encoded and placed in an additional query string parameter named RelayState.](#) ~~In addition, if the~~
703 ~~artifact represents a SAML response message that contains a <RelayState> value, that value MUST be~~
704 ~~placed in a second query string parameter named "RelayState". This is to insure that any relevant state is~~
705 ~~included in case the artifact cannot be dereferenced.~~

706 **3.5.3.3 Form Encoding**

707 A SAML artifact is form-encoded by placing it in a hidden form control within a form as defined by
708 [HTML401], chapter 17. [The HTML document MUST adhere to the XHTML specification, \[XHTML\].](#) The
709 form control MUST be named ["SAMLart"](#). ~~In addition, if the artifact represents a SAML response~~
710 ~~message that contains a <RelayState> value, that value MUST be placed in a second hidden form~~
711 ~~control named "RelayState". This is to insure that any relevant state is included in case the artifact cannot~~
712 ~~be dereferenced.~~ Any additional form controls or presentation MAY be included but MUST NOT be
713 required in order for the recipient to process the artifact.

714 The action attribute of the form MUST be the recipient's HTTP endpoint for the protocol or profile using
715 this binding to which the artifact is to be delivered. The method attribute MUST be set to "POST".

716 [If a "RelayState" value is to accompany the SAML artifact, it MUST be placed in an additional hidden form](#)
717 [control named RelayState.](#)

718 Any technique supported by the user agent MAY be used to cause the submission of the form, and any
719 form content necessary to support this MAY be included, such as submit controls and client-side scripting
720 commands. However, the recipient MUST be able to process the artifact without regard for the
721 mechanism by which the form submission is initiated.

722 **3.5.4 Artifact Types**

723 With respect to this binding, an artifact is a short, opaque string. Different types can be defined and used
724 without affecting the binding. The important characteristics are the ability of an artifact receiver to identify
725 the issuer of the artifact, resistance to tampering and forgery, uniqueness, and compactness.

726 [TODO: define a normative artifact format for 2.0. SAML 1.x and ID-FF 1.2 define the following artifact](#)
727 [formats:](#)

728 [SourceID as a fixed-length opaque value](#)

729 [SourceID as a URL location at which to dereference the artifact](#)

730 [SourceID as a hash of the Liberty ProviderID, the unique key that references metadata](#)

731 [The most interesting answer may be a fourth: using the ProviderID directly as the SourceID, and using](#)
732 [metadata to publish the actual location. The downside is the loss of fixed length, but the value of that is](#)
733 [unclear.](#)

734 **3.5.5 Message Exchange**

735 The system model used for SAML conversations via this binding is a request-response model in which an
736 artifact reference takes the place of the actual message content, and the artifact reference is sent to the
737 user agent in an HTTP response and delivered to the message recipient in an HTTP request. The HTTP
738 interactions before, between, and after these exchanges take place is undefined. Both the SAML
739 requester and responder are assumed to be HTTP responders.

740 Additionally, it is assumed that upon receipt of an artifact by way of the user agent, the recipient invokes a
741 separate, direct exchange with the artifact issuer using the Artifact Protocol defined in [SAMLCore]. This
742 exchange MUST use a binding that does not use the HTTP user agent as an intermediary, such as a
743 SOAP binding. Upon the successful acquisition of a [SAML](#) protocol message, the artifact is discarded and
744 the processing of the primary SAML protocol exchange resumes (or ends, if the message is a response).

745 Issuing and delivering an artifact, along with the subsequent dereference, constitutes half of the overall
746 SAML protocol exchange. This binding can be used to deliver either or both halves of a [SAML](#) protocol
747 exchange. A binding composable with it, such as the HTTP-Redirect/POST binding, MAY be used to carry
748 the other half of the exchange. The following sequence assumes that the artifact binding is used for both
749 halves.

750 [TODO: sequence diagram](#)

- 751 1. A system entity acting as a SAML requester responds to an HTTP request from the user agent by
752 returning an artifact representing a SAML request.
- 753 2. If URL-encoded, the artifact is returned encoded into the HTTP response's Location header,
754 and the HTTP status MUST be either 303 or 302. The SAML requester MAY include additional
755 presentation and content in the HTTP response to facilitate the user agent's transmission of
756 the message, as defined in [HTTP 1.1](#), [RFC2616]. The user agent delivers the artifact by
757 issuing an HTTP GET request to the SAML responder.
- 758 3. If form-encoded, then the artifact is returned in an [XHTML] document containing the form and
759 content defined in section 3.5.3.32. The user agent delivers the artifact by issuing an HTTP
760 POST request to the SAML responder.
- 761 4. The SAML responder determines the SAML requester by examining the artifact ([the exact process
762 depends on the type of artifact](#)), and issues an <ArtifactRequest> containing the artifact to the
763 SAML requester using a direct [SAML](#) binding, temporarily reversing roles. Assuming the necessary
764 conditions are met, the SAML requester returns an <ArtifactResponse> containing the original
765 SAML request message it wishes the responder to process.
- 766 5. In general, the SAML responder MAY respond to the SAML request by immediately returning a
767 SAML artifact or MAY return arbitrary content to facilitate subsequent interaction with the user agent
768 necessary to fulfill the request. Specific protocols and profiles may include mechanisms to indicate
769 the requester's (un)willingness to permit this kind of interaction. Eventually the responder SHOULD
770 return a SAML artifact to the user agent to be returned to the SAML requester. The SAML response
771 artifact is returned in the same fashion as described for the SAML request artifact.
- 772 6. The SAML requester determines the SAML responder by examining the artifact, and issues an
773 <ArtifactRequest> containing the artifact to the SAML responder using a direct [SAML](#) binding.
774 Assuming the necessary conditions are met, the SAML responder returns an
775 <ArtifactResponse> containing the SAML response message it wishes the requester to
776 process.

777 3.5.5.1 HTTP Considerations

778 HTTP proxies and the user agent intermediary MUST NOT cache SAML artifacts. [HTTP headers](#)
779 [SHOULD](#) be used to protect against such caching behavior.

780 ~~Both of the following conditions apply when using HTTP 1.1 during message exchange:~~

- 781 1. ~~If the value of the Cache-Control header field is not set to no-store, then the HTTP responder~~
782 ~~returning a SAML artifact MUST NOT include the Cache-Control header field in the HTTP~~
783 ~~response.~~
- 784 2. ~~If the Expires HTTP response header field is not disabled by a Cache-Control header field with~~
785 ~~a value of no-store, then the Expires header SHOULD NOT be included in the HTTP response.~~

786 There are no other restrictions on the use of HTTP headers.

787 **3.5.5.2 Authentication**

788 This binding uses a combination of indirect transmission of a message reference followed by a direct
789 exchange to return the actual message. As a result, the message reference (artifact) need not itself be
790 authenticated, but the callback request/response exchange that returns the actual message MAY be
791 mutually authenticated, depending on the environment of use.

792 If the actual [SAML protocol](#) message is intended for a specific recipient, then the [artifact's sender/issuer](#)
793 MUST authenticate the [sender of the subsequent](#) <ArtifactRequest> message ~~or sender~~ before
794 returning the actual message.

795 **3.5.5.3 Message Integrity**

796 This binding uses a combination of indirect transmission of a message reference followed by a direct
797 exchange to return the actual message. As a result, the message reference (artifact) need not itself be
798 integrity protected, but the callback request/response exchange that returns the actual message MAY be
799 protected, depending on the environment of use.

800 **3.5.5.4 Confidentiality**

801 The transmission of an artifact to and from the user agent MUST be protected with confidentiality; SSL 3.0
802 or TLS 1.0 SHOULD be used. The callback request/response exchange that returns the actual message
803 MAY be protected, depending on the environment of use.

804 **3.5.6 Security Considerations**

805 Before deployment, each combination of authentication, message integrity, and confidentiality
806 mechanisms SHOULD be analyzed for vulnerability in the context of the specific protocol exchange, and
807 the deployment environment. See specific protocol processing rules in [SAMLCore], and the SAML
808 security considerations document [SAMLSec] for a detailed discussion.

809 In general, this binding relies on the artifact as a hard-to-forge short-term reference and applies other
810 security measures to the callback request/response that returns the actual message.

811 **3.5.7 Error Reporting**

812 A SAML responder that refuses to perform a message exchange with the SAML requester SHOULD
813 return a <StatusResponse> [response](#) message with a second level <StatusCode> value of
814 RequestDenied.

815 HTTP interactions during the message exchange MUST NOT use HTTP error status codes to indicate
816 failures in SAML processing, since the user agent is not a full party to the SAML protocol [exchange](#).

817 If the issuer of an artifact receives an <ArtifactRequest> message that it can understand, it MUST
818 return an <ArtifactResponse> with a <StatusCode> value of Success, even if it does not return the
819 corresponding message ([for example](#) because the artifact requester is not authorized to receive the
820 message or the artifact is no longer valid).

821 For more information about SAML status codes, see the SAML assertion and protocol specification
822 [SAMLCore].

823 **3.5.8 Metadata Considerations**

824 Support for the HTTP Artifact binding SHOULD be reflected by indicating URL endpoints at which
825 requests and responses for a particular protocol or profile should be sent. Either a single endpoint or
826 distinct request and response endpoints should be possible. An endpoint for processing

827 <ArtifactRequest> messages SHOULD also be described.

828 3.5.9 Example SAML Message Exchange Using HTTP Artifact

829 TBD

830 3.6 SAML URI Binding

831 URIs are a protocol-independent means of referring to a resource. This binding is not a general SAML
832 request/response binding, but [rather](#) supports the encapsulation of an <AssertionIDRequest>
833 message with a single <AssertionIDReference> into the resolution of a URI. The result of a
834 successful request is a SAML <Assertion> element.

835 Like SOAP, URI resolution can occur over multiple underlying transports. This binding has
836 [protocol/transport](#)-independent aspects, but also calls out the use of HTTP with SSL 3.0 or TLS 1.0 as
837 REQUIRED (mandatory to implement).

838 3.6.1 Required Information

839 **Identification:** urn:oasis:names:tc:SAML:2.0:bindings:URI-binding

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

841 **Description:** Given below.

842 **Updates:** None

843 3.6.2 Protocol-Independent Aspects of the SAML URI Binding

844 The following sections define aspects of the SAML URI binding that are independent of the underlying
845 [transport](#) protocol of the URI resolution process.

846 3.6.2.1 Basic Operation

847 A SAML URI reference identifies a specific SAML assertion. The result of resolving the URI MUST be a
848 message containing the assertion, or a [transport/protocol](#)-specific error. If the [resolution/transport](#) protocol
849 permits the returned content to be described, such as HTTP 1.1- [RFC2616], then the assertion MAY be
850 encoded in whatever format is [permitted/allowed by that protocol](#). If not, the assertion MUST be returned in
851 a form which can be unambiguously interpreted as or transformed into an XML serialization of the
852 assertion.

853 It MUST be the case that if the same URI reference is resolved in the future, then either the same SAML
854 assertion, or an error, is returned. That is, the reference MAY be persistent but MUST consistently
855 reference the same assertion, if any.

856 3.6.2.2 Authentication

857 Authentication of the requester, responder, and the assertion itself are OPTIONAL and depend on the
858 environment of use. Authentication mechanisms available from the underlying substrate protocol MAY be
859 utilized to provide authentication. The resulting assertion MAY also be signed. Section 3.6.5.3 describes
860 authentication during HTTP URI resolution.

861 3.6.2.3 Message Integrity

862 Message integrity of the request, response, and assertion are OPTIONAL and depend on the environment

863 of use. The security layer in the underlying substrate protocol MAY be used to ensure message integrity.
864 Section 3.6.5.4 describes support for message integrity during HTTP URI resolution.

865 **3.6.2.4 Confidentiality**

866 Confidentiality of the request, response, and the assertion itself are OPTIONAL and depend on the
867 environment of use. The security layer in the underlying substrate protocol MAY be used to ensure
868 message confidentiality. Section 3.6.5.5 describes support for confidentiality during HTTP URI resolution.

869 **3.6.3 General Security Considerations**

870 Before deployment, each combination of authentication, message integrity, and confidentiality
871 mechanisms SHOULD be analyzed for vulnerability in the context of the specific protocol exchange, and
872 the deployment environment. See specific protocol processing rules in [SAMLCore], and the SAML
873 security considerations document [SAMLSec] for a detailed discussion.

874 Indirect use of a SAML assertion presents dangers if the binding of the reference to the result ~~of the~~
875 [indirection](#) is not secure. The particular threats and their severity depend on the use to which the assertion
876 is being put. In general, the result of resolving a URI reference to a SAML assertion SHOULD only be
877 trusted if the requester can be certain of the identity of the responder and that the contents have not been
878 modified in transit.

879 It is often not sufficient that the assertion itself be signed, because URI references are by their nature
880 somewhat opaque to the requester. The requester SHOULD have independent means to insure that the
881 assertion returned is actually the one that is represented by the URI; this is accomplished by [both](#)
882 authenticating the responder and relying on the integrity of the response.

883 **3.6.4 MIME Encapsulation**

884 For resolution protocols that support MIME as a content description and packaging mechanism, the
885 resulting assertion SHOULD be returned as a MIME entity of type "application/saml+xml", as defined by
886 XX.

887 **3.6.5 Use of HTTP URIs**

888 A SAML [processor authority](#) that claims conformance to the SAML URI binding MUST implement support
889 for HTTP. This section describes certain specifics of using HTTP URIs, including URI syntax, HTTP
890 headers, error reporting, authentication, message integrity, and confidentiality.

891 **3.6.5.1 URI Syntax**

892 In general, there are no restrictions on the permissible syntax of a SAML URI reference as long as the
893 SAML authority responsible for the reference creates the message containing it. However, authorities
894 MUST support a URL endpoint at which an HTTP request can be sent with a single query string
895 parameter named "[AssertionID](#)". There MUST be no query string in the endpoint URL itself
896 independent of the parameter.

897 For example, if the documented endpoint at an authority is "<https://saml.example.edu/assertions>", a
898 request for an assertion with `AssertionID` of abcde can be sent to
899 "<https://saml.example.edu/assertions?AssertionID=abcde>".

900 [Note that the use of wildcards is not allowed for such AssertionID queries.](#)

901 **3.6.5.2 HTTP Headers**

902 [HTTP proxies MUST NOT cache SAML assertions. HTTP headers SHOULD be used to protect against](#)

903 | [such caching behavior.](#)

904 | ~~HTTP header should have a cache-control header indicating that HTTP proxies MUST NOT cache~~
905 | ~~responses carrying SAML assertions.~~

906 | ~~Both of the following conditions apply when using HTTP 1.1:~~

907 | 1. ~~If the value of the Cache-Control header field is not set to no-store, then the SAML responder~~
908 | ~~MUST NOT include the Cache-Control header field in the response.~~

909 | 2. ~~If the Expires response header field is not disabled by a Cache-Control header field with a~~
910 | ~~value of no-store, then the Expires field SHOULD NOT be included.~~

911 | ~~There are no other restrictions on HTTP headers.~~

912 | **3.6.5.3 Authentication**

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

914 | 1. No client or server authentication.

915 | 2. HTTP basic client authentication [RFC2617] with and without SSL 3.0 or TLS 1.0.

916 | 3. HTTP over SSL 3.0 or TLS 1.0 (see Section 3.1.1) server authentication with a server-side
917 | certificate.

918 | 4. HTTP over SSL 3.0 or TLS 1.0 mutual authentication with both server-side and a client-side
919 | certificate.

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

921 | **3.6.5.4 Message Integrity**

922 | When message integrity needs to be guaranteed, SAML responders MUST implement HTTP over SSL
923 | 3.0 or TLS 1.0 (see Section 3.1.1) with a server-side certificate.

924 | **3.6.5.5 Message Confidentiality**

925 | When message confidentiality is required, SAML responders MUST implement HTTP over SSL 3.0 or
926 | TLS 1.0 (see Section 3.1.1) with a server-side certificate.

927 | **3.6.5.6 Security Considerations**

928 | [RFC2617] describes possible attacks in the HTTP environment when basic or message-digest
929 | authentication schemes are used.

930 | **3.6.5.7 Error Reporting**

931 | As an HTTP protocol exchange, the appropriate HTTP status code SHOULD be used to indicate the result
932 | of a request. For example, a SAML responder that refuses to perform a message exchange with the
933 | SAML requester SHOULD return a "403 Forbidden" response. If the assertion specified is unknown to
934 | the responder, then a "404 Not Found" response SHOULD be returned. In these cases, the content of
935 | the HTTP body is not significant.

936 | **3.6.5.8 Metadata Considerations**

937 | Support for the URI binding over HTTP SHOULD be reflected by indicating a URL endpoint at which
938 | requests for arbitrary assertions are to be sent.

939 **3.6.5.9 Example SAML Message Exchange Using an HTTP URI**

940 Following is an example of a request for an assertion.

```
941 GET /SamlService?AssertionID=abcde HTTP/1.1  
942 Host: www.example.com
```

943 Following is an example of the corresponding response, which supplies the requested assertion.

```
944 HTTP/1.1 200 OK  
945 Content-Type: application/saml+xml  
946 Content-Length: nnnn  
  
947 <saml:Assertion AssertionID="abcde" ...>  
948 ...  
949 </saml:Assertion>
```

4 SAML Artifact Formats

[TODO: Move one or more formats into the definition of the artifact binding.](#)

The general format of an artifact includes a mandatory two-byte artifact type code, as follows:

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

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

The notation `B64(TypeCode RemainingArtifact)` stands for the application of the base64 [RFC2045] transformation to the catenation of the `TypeCode` and `RemainingArtifact`.

4.1 SAML 1.x Format

4.1.1 Required Information

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

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

Description: Given below.

Updates: None.

4.1.2 Format Details

This profile defines an artifact type of type code 0x0001. This artifact type is defined as follows:

```
TypeCode           := 0x0001
RemainingArtifact  := SourceID AssertionHandle
SourceID           := 20-byte_sequence
AssertionHandle    := 20-byte_sequence
```

`SourceID` is a 20-byte sequence used by the artifact receiver to determine artifact issuer identity and location. It is assumed that the destination site will maintain a table of `SourceID` values as well as the URL (or address) for the corresponding SAML responder, or alternatively metadata containing it. This information is communicated between the parties out-of-band. On receiving the SAML artifact, the receiver determines if the `SourceID` belongs to a known artifact issuer and obtains the location of the SAML responder before sending a SAML artifact request.

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

The following practices are RECOMMENDED for the creation of SAML artifacts:

- Each issuer selects a single identification URL. The domain name used within this URL is registered with an appropriate authority and administered by the issuer.
- The issuer constructs the `SourceID` component of the artifact by taking the SHA-1 hash of the identification URL.
- The `AssertionHandle` value is constructed from a cryptographically strong random or pseudorandom number sequence [RFC1750] generated by the issuer. The sequence consists of

989 values of at least eight bytes in size. These values should be padded to a total length of 20 bytes.

990 4.2 SAML 1.x Supplemental Format

991 4.2.1 Required Information

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

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

994 **Description:** Given below.

995 **Updates:** None.

996 4.2.2 Format Details

997 An alternative artifact format is described here:

998	TypeCode	:=	0x0002
999	RemainingArtifact	:=	AssertionHandle SourceLocation
1000	AssertionHandle	:=	20-byte_sequence
1001	SourceLocation	:=	URI

1002 The `SourceLocation` URI is the address of the SAML responder associated with the artifact issuer. The
1003 `AssertionHandle` is as described in Section 4.1.2, and governed by the same requirements. The
1004 `SourceLocation` URI is mapped to a sequence of bytes based on use of the UTF-8 [RFC2279]
1005 encoding. The receiver MUST process the artifact in a manner identical to that described in Section 4.1.2,
1006 with the exception that the location of the SAML responder of the issuer MAY be obtained directly from
1007 the artifact, rather than by look-up, based on `SourceID`.

1008 **Note:** the receiver MUST confirm that the corresponding message is issued by an acceptable issuer, not
1009 relying merely on the fact that it was returned in response to a `<samlp:ArtifactRequest>` message.

1010 5 URL Size Restriction (Non-Normative)

1011 This section describes the URL size restrictions that have been documented for widely used commercial
1012 products.

1013 A Microsoft technical support article [MSURL] provides the following information:

1014 The information in this article applies to:

1015 Microsoft Internet Explorer (Programming) versions 4.0, 4.01, 4.01 SP1, 4.01 SP2, 5, 5.01, 5.5

1016 SUMMARY

1017 Internet Explorer has a maximum uniform resource locator (URL) length of 2,083 characters, with a
1018 maximum path length of 2,048 characters. This limit applies to both POST and GET request URLs.

1019 If you are using the GET method, you are limited to a maximum of 2,048 characters (minus the
1020 number of characters in the actual path, of course).

1021 POST, however, is not limited by the size of the URL for submitting name/value pairs, because they
1022 are transferred in the header and not the URL.

1023 RFC 2616, Hypertext Transfer Protocol -- HTTP/1.1, does not specify any requirement for URL length.

1024 REFERENCES

1025 Further breakdown of the components can be found in the Wininet header file. Hypertext Transfer
1026 Protocol -- HTTP/1.1 General Syntax, section 3.2.1

1027 Last Reviewed: 9/13/2001

1028 Keywords: kbDSupport kbFAQ kbinfo KB208427

1029 An article about Netscape Enterprise Server provides the following information:

1030 Issue: 19971110-3 Product: Enterprise Server

1031 Created: 11/10/1997 Version: 2.01

1032 Last Updated: 08/10/1998 OS: AIX, Irix, Solaris

1033 Does this article answer your question?

1034 Please let us know!

1035 Question:

1036 How can I determine the maximum URL length that the Enterprise server will accept? Is this
1037 configurable and, if so, how?

1038 Answer:

1039 Any single line in the headers has a limit of 4096 chars; it is not configurable.

6 References

1040

- 1041 **[AES]** FIPS-197, Advanced Encryption Standard (AES), available from <http://www.nist.gov/>.
- 1042 **[Anders]** A suggestion on how to implement SAML browser bindings without using “Artifacts”,
1043 <http://www.x-obi.com/OBI400/andersr-browser-artifact.ppt>.
- 1044 **[CoreAssnEx]** Core Assertions Architecture, Examples and Explanations, [http://www.oasis-](http://www.oasis-open.org/committees/security/docs/draft-sstc-core-phill-07.pdf)
1045 [open.org/committees/security/docs/draft-sstc-core-phill-07.pdf](http://www.oasis-open.org/committees/security/docs/draft-sstc-core-phill-07.pdf).
- 1046 **[HTML401]** HTML 4.01 Specification, W3C Recommendation 24 December 1999,
1047 <http://www.w3.org/TR/html4>.
- 1048 **[XHTML]** XHTML 1.0 The Extensible HyperText Markup Language (Second Edition),
1049 <http://www.w3.org/TR/xhtml1/>.
- 1050 **[Liberty]** The Liberty Alliance Project, <http://www.projectliberty.org>.
- 1051 **[MSURL]** Microsoft technical support article,
1052 <http://support.microsoft.com/support/kb/articles/Q208/4/27.ASP>.
- 1053 **[PAOS]** Aarts, R., “Liberty Reverse HTTP Binding for SOAP Specification”, Version: 1.0,
1054 <https://www.projectliberty.org/specs/liberty-paos-v1.0.pdf>
- 1055 **[Rescorla-Sec]** E. Rescorla et al., *Guidelines for Writing RFC Text on Security Considerations*,
1056 <http://www.ietf.org/internet-drafts/draft-iab-sec-cons-03.txt>.
- 1057 **[RFC1952]** GZIP file format specification version 4.3,
1058 <http://www.ietf.org/rfc/rfc1952.txt>
- 1059 **[RFC1738]** Uniform Resource Locators (URL), <http://www.ietf.org/rfc/rfc1738.txt>
- 1060 **[RFC1750]** Randomness Recommendations for Security. <http://www.ietf.org/rfc/rfc1750.txt>
- 1061 **[RFC1945]** Hypertext Transfer Protocol -- HTTP/1.0, <http://www.ietf.org/rfc/rfc1945.txt>.
- 1062 **[RFC2045]** Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message
1063 Bodies, <http://www.ietf.org/rfc/rfc2045.txt>
- 1064 **[RFC2119]** S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, IETF RFC 2119,
1065 March 1997, <http://www.ietf.org/rfc/rfc2119.txt>.
- 1066 **[RFC2246]** The TLS Protocol Version 1.0, <http://www.ietf.org/rfc/rfc2246.txt>.
- 1067 **[RFC2279]** UTF-8, a transformation format of ISO 10646, <http://www.ietf.org/rfc/rfc2279.txt>.
- 1068 **[RFC2616]** Hypertext Transfer Protocol -- HTTP/1.1, <http://www.ietf.org/rfc/rfc2616.txt>.
- 1069 **[RFC2617]** *HTTP Authentication: Basic and Digest Access Authentication*, IETF RFC 2617,
1070 <http://www.ietf.org/rfc/rfc2617.txt>.
- 1071 **[SAMLCore]** E. Maler et al. *Assertions and Protocol for the OASIS Security Assertion Markup*
1072 *Language (SAML)*. OASIS, September 2003. Document ID oasis-sstc-saml-core-1.1.
1073 <http://www.oasis-open.org/committees/security/>.
- 1074 **[SAMLGloss]** E. Maler et al. *Glossary for the OASIS Security Assertion Markup Language (SAML)*.
1075 OASIS, September 2003. Document ID oasis-sstc-saml-glossary-1.1. [http://www.oasis-](http://www.oasis-open.org/committees/security/)
1076 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 1077 **[SAMLProfile]** *Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0, DRAFT*
- 1078 **[SAMLSec]** E. Maler et al. *Security Considerations for the OASIS Security Assertion Markup*
1079 *Language (SAML)*, OASIS, September 2003, Document ID oasis-sstc-saml-sec-consider-1.1.
1080 <http://www.oasis-open.org/committees/security/>.
- 1081 **[SAMLReqs]** Darren Platt et al., *SAML Requirements and Use Cases*, OASIS, April 2002,
1082 <http://www.oasis-open.org/committees/security/>.
- 1083 **[SAMLWeb]** OASIS Security Services Technical Committee website, [sstc-saml-bindings-2.0-draft-098
Copyright © OASIS Open 2003-2004. All Rights Reserved.](http://www.oasis-</p></div><div data-bbox=)

1084 open.org/committees/security.

1085 **[SESSION]** RL “Bob” Morgan, Support of target web server sessions in Shibboleth,
1086 <http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-session-00.txt>

1087 **[ShibMarlena]** Marlena Erdos, Shibboleth Architecture DRAFT v1.1,
1088 <http://shibboleth.internet2.edu/draft-internet2-shibboleth-arch-v05.html> .

1089 **[SOAP1.1]** D. Box et al., *Simple Object Access Protocol (SOAP) 1.1*, World Wide Web Consortium
1090 Note, May 2000, <http://www.w3.org/TR/SOAP>.

1091 **[SSL3]** A. Frier et al., *The SSL 3.0 Protocol*, Netscape Communications Corp, November 1996.

1092 **[WEBSSO]** RL “Bob” Morgan, Interactions between Shibboleth and local-site web sign-on services,
1093 <http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-websso-00.txt>

1094 **[WSS-SAML]** P. Hallam-Baker et al., *Web Services Security: SAML Token Profile*, OASIS, March 2003,
1095 <http://www.oasis-open.org/committees/wss>.

1096 **[XMLSig]** D. Eastlake et al., *XML-Signature Syntax and Processing*, World Wide Web Consortium,
1097 <http://www.w3.org/TR/xmlsig-core/>.

1098 **A. Acknowledgments**

1099 The editors would like to acknowledge the contributions of the OASIS Security Services Technical
1100 Committee, whose voting members at the time of publication were:

- 1101 • TBD

B. Revision History

Rev	Date	By Whom	What
1	02/16/04	Frederick Hirsch	Split Bindings and Profiles into two documents. Removed profiles from this document, added PAOS reverse SOAP binding
4	02/25/04	Scott Cantor	General language changes in introduction to reflect binding support of any SAML protocol Updated SOAP binding to reflect intended expansion of use across other protocols and SOAP-specific security mechanisms Removed confirmation methods, they don't belong in binding discussions.
5	02/26/04	Scott Cantor	Revised arrangement of SSL discussion Added HTTP-Redirect/POST binding Added metadata considerations to SOAP binding
6	03/01/04	John Kemp	Added 3 sections of URL-encoding.
7	03/14/04	Scott Cantor	Added HTTP Artifact and URI bindings
8	03/27/04	Frederick Hirsch	Updates for core 8, review comments and corrections.
9	04/09/04	Scott Cantor	Tightened URL encoding/signing rules Added text around PAOS Incorporated feedback from FTF Placeholders for some additional work items.

C. Notices

1104 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
1105 might be claimed to pertain to the implementation or use of the technology described in this document or
1106 the extent to which any license under such rights might or might not be available; neither does it represent
1107 that it has made any effort to identify any such rights. Information on OASIS's procedures with respect to
1108 rights in OASIS specifications can be found at the OASIS website. Copies of claims of rights made
1109 available for publication and any assurances of licenses to be made available, or the result of an attempt
1110 made to obtain a general license or permission for the use of such proprietary rights by implementors or
1111 users of this specification, can be obtained from the OASIS Executive Director.

1112 OASIS invites any interested party to bring to its attention any copyrights, patents or patent applications, or
1113 other proprietary rights which may cover technology that may be required to implement this specification.
1114 Please address the information to the OASIS Executive Director.

1115 **Copyright © OASIS Open 2003-2004. All Rights Reserved.**

1116 This document and translations of it may be copied and furnished to others, and derivative works that
1117 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and
1118 distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and
1119 this paragraph are included on all such copies and derivative works. However, this document itself may
1120 not be modified in any way, such as by removing the copyright notice or references to OASIS, except as
1121 needed for the purpose of developing OASIS specifications, in which case the procedures for copyrights
1122 defined in the OASIS Intellectual Property Rights document must be followed, or as required to translate it
1123 into languages other than English.

1124 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
1125 or assigns.

1126 This document and the information contained herein is provided on an "AS IS" basis and OASIS
1127 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY
1128 WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR
1129 ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

1130 JavaScript is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and
1131 other countries.