

OASIS SSTC Bindings Model

1

2

3 Prateek Mishra, Netegrity

4 Marlena Erdos, Tivoli

5 Chris Ferris, SUN Microsystems

6 Simon Godik, Crosslogix

7 Jeff Hodges, Oblix

8 **<big><small>**Tim Moses, Entrust

9 Bob Morgan, University of Washington

10 Evan Prodromou, Securant

11 Krishna Sankar, Cisco

12 **</small>**

13 draft-sstc-bindings-model-05.doc

14

15 22 August 2001

16

16		
17	OASIS SSTC Bindings Model.....	1
18	1 Revision History.....	4
19	2 Introduction.....	5
20	2.1 Scope.....	5
21	2.2 Contents.....	6
22	2.3 Guidelines for Specifying Protocol Bindings and Profiles.....	6
23	2.4 Process Framework for Describing and Registering Protocol Bindings and Profiles.....	7
24	3 Protocol Bindings.....	8
25	3.1 HTTP.....	8
26	3.1.1 Introduction.....	8
27	3.1.2 Overview.....	8
28	3.1.3 HTTP Binding.....	9
29	3.1.3.1 Connections.....	9
30	3.1.3.2 Request Messages.....	9
31	3.1.3.3 Response Messages.....	10
32	3.1.3.4 Authentication and Message Integrity.....	10
33	3.1.3.5 Message Confidentiality.....	11
34	3.1.3.6 Errors.....	11
35	3.2 SOAP 1.1.....	12
36	3.2.1 Introduction.....	12
37	3.2.2 Overview.....	13
38	3.2.3 SOAP Binding.....	13
39	3.2.3.1 Namespaces.....	13
40	3.2.3.2 Headers.....	13
41	3.2.3.3 SAML Queries.....	14
42	3.2.3.4 SAML Query Responses.....	14
43	3.2.3.5 Fault Codes.....	14
44	3.2.3.6 Authentication and Message Integrity.....	14
45	3.2.3.7 Confidentiality.....	15
46	4 Profiles.....	15
47	4.1 Web Browser.....	15
48	4.1.1 Background.....	15

49	4.1.2	Relevant Technology.....	16
50	4.1.3	SAML artifact structure	16
51	4.1.4	Profile Overview	17
52	4.1.4.1	SAML Artifact (Pull)	17
53	4.1.4.2	SAML Artifact (Push).....	20
54	4.1.4.3	Form POST	23
55	4.1.5	Threat Model and Counter-Measures.....	26
56	4.1.5.1	Stolen artifact or assertion.....	26
57	4.1.5.2	Forged SAML artifact or Assertion	27
58	4.1.5.3	Browser State Exposure	27
59	4.2	SOAP.....	28
60	4.2.1	Overview	28
61	4.2.2	SOAP Headers and Error Processing.....	28
62	4.2.3	Confidentiality.....	29
63	4.2.4	Example.....	29
64	4.2.5	Integrity of Assertion Attachment.....	30
65	4.2.5.1	Digest of SOAP Message.....	30
66	4.2.5.2	Digital Signature	30
67	5	References	31
68	6	Appendix A	32
69	7	Appendix B	34
70			
71			

71 **1 Revision History**

Revision	Date	Author	1.1.1.1.1 Title
0.5	18 August 2001	Prateek Mishra	Bindings model draft

72

73

74

75

76

77

77 2 Introduction

78 2.1 Scope

79 **Other Oasis Security Services TC subcommittees (e.g. Core Assertions and Protocol) are**
80 **producing a specification of SAML security assertions and one or more SAML**
81 **request-response message exchanges.**
82

83 **The high-level goal of this document is to specify how:**
84

85 **(1) SAML request-response message exchanges are mapped into standard messaging or**
86 **communication protocols. Such mappings are called SAML**
87 **protocol bindings. An instance of mapping SAML request-response**
88 **message exchanges into a specific protocol <FOO> is termed a SAML**
89 **<FOO> binding.**

90

91 Example: A SAML HTTP binding describes how SAML Query and Response message
92 exchanges are mapped into HTTP message exchanges. A SAML SOAP binding describes how
93 SAML Query and Response message exchanges are mapped into SOAP message
94 exchanges.
95

96 **(2) SAML security assertions are embedded in or combined with other objects (e.g. files**
97 **of various types, protocol data units of communication protocols) by an originating party,**
98 **communicated from the originating site to a destination, and**
99 **subsequently processed at the destination. A set of rules describing**
100 **how to embed and extract SAML assertions into a framework or protocol is termed a**
101 **profile for SAML. A set of rules for embedding and extracting SAML**
102 **assertions into a specific class of <FOO> objects is termed a**
103 **<FOO> profile for SAML.**
104

105 Example: A SOAP profile for SAML describes how SAML assertions may be added to SOAP
106 messages, the interaction between SOAP headers and SAML assertions, description of SAML-
107 related error states at the destination.

108

109

110 **(1) and (2) MUST be specified in sufficient detail to yield interoperability when**
111 **independently implemented.**
112

113 2.2 Contents

114 <big>The remainder of this document is in four sections:

115 </big>

116 • <big>Guidelines for the specification of protocol bindings and profiles. The intent here is
117 to provide a checklist that MUST or SHOULD be filled out when developing a protocol
118 binding or profile for a specific protocol or framework.

119 </big>

120 • <big>A process framework for describing and registering proposed and future protocol
121 bindings and profiles.

122 </big>

123 • <big>Protocol bindings for selected protocols. Bindings MUST be specified in enough
124 detail to satisfy the inter-operability requirement.

125 </big>

126 • <big>Profiles for selected protocols and frameworks. Profiles MUST be specified in
127 enough detail to satisfy the inter-operability requirement.

128 </big>

129 2.3 Guidelines for Specifying Protocol Bindings and 130 Profiles<big> </big>

131 <big>Issues that MUST be identified in each protocol binding and profile:</big><big>

132 </big><big></big><big></big><big></big><big>

133 </big><big>(1) Each binding or profile must be characterized as set of interactions between
134 parties. Any restriction on applications used by each party and the protocols involved in each
135 interaction must be explicitly called out.</big><big>

136 </big><big>

137 </big><big>(2) Identification of parties involved in each interaction: how many parties are
138 involved in the interaction? Can intermediaries be involved?

139 </big>

140 </big><big>(3) Authentication of parties involved in each interaction: Is authentication required? What
141 types of authentication are acceptable?</big><big>

142 </big><big>

143 </big><big>(4) Support for message integrity: what mechanisms are used to ensure message
144 integrity?

145 </big>

146 </big><big>(5) Support for Confidentiality: can a third party view the contents of SAML messages and
147 assertions? Does the binding or profile require confidentiality? What mechanisms are
148 recommended for securing confidentiality? </big><big></big><big>

149 </big><big>

150 </big><big>(6) Error states: characterization of error states at each participant, especially those
151 that receive and process SAML assertions or messages.</big>

152 </big>

153

154 (7) Support for *integrity of assertion attachment*. Many profiles consist of a set of rules for
155 adding assertions to an existing protocol or packaging framework. These rules will be used by an
156 originating party (e.g., user, server) to create a *composite package* consisting of assertions and a
157 business payload for delivery to a destination. When the composite package arrives at the
158 destination, the recipient will require proof (1) the originating party is the subject of the
159 assertions contained within the composite package, (2) neither the assertion nor business payload
160 have been altered.

161

162 The term *integrity of assertion attachment* refers to the linkage between the originating party,
163 assertions and business payload, created when an originating party constructs the composite
164 package. Integrity of assertion attachment MUST be verifiable by a recipient. Typically,
165 mechanisms provided to support attachment integrity will be based on some cryptographic
166 techniques (hash or digital signature).

167

168 2.4 Process Framework for Describing and Registering 169 Protocol Bindings and Profiles

170

171 **<big>**When a profile or protocol binding is registered, the following information is
172 supplied:**</big>**

173 **<big>** **</big>**

174 1. **<big>**Identification: specify a URI that authoritatively identifies this profile or protocol
175 binding.

176 **</big>**

177 2. **<big>**Contact information: specify the postal and electronic contact information for the
178 author of the profile or protocol binding.

179 **</big>**

180 3. **<big>**Description: the description MUST follow the guidelines for profiles and protocol
181 bindings given above.

182 **</big>**

183 4. **<big>**Updates: references to previously registered profiles or bindings that the current
184 entry improves or obsoletes.**</big><big>**

185 **</big>**

186 *ISSUE:[BINDINGS-01] Where should this registry be maintained? It has been proposed that*
187 *IANA (<http://www.iana.org>) might provide an appropriate forum. Further investigation is*
188 *required.*

189

190 **<big>***Whe***</big>**

191

192 **3 Protocol Bindings**

193 **3.1 HTTP**

194 ***3.1.1 Introduction***

195 **<big>**
196 **</big><big></big><big>**The following binding description derives from the HTTP binding
197 provided with [AuthXML]. **</big><big></big><big>**Note that this section does not treat the
198 issue of passing SAML assertions or assertion tokens from a standard Web browser to a Web
199 server. Instead, it concentrates on using HTTP as a transport layer for SAML messages, without
200 the restrictions that standard Web browsers impose. In most cases, this binding will be used as a
201 service-to-service binding, rather than a user-to-service binding.

202 **</big>**

203 **<big> </big><big>**Some design goals of this binding are as follows:

204 **</big>**

- 205 • **<big>**Enable using existing HTTP software (Web servers, client libraries) to create SAML
206 services.**</big>**
- 207 • **<big>**Minimize requirements for supporting the somewhat complex HTTP protocol.**</big>**
- 208 • **<big>**Minimize the information carried in HTTP headers and other data. Except in extreme
209 situations, information should be passed as SAML.**</big>**

210

211 **<big> </big><big>**Readers of this document should be familiar with HTTP/1.1, which is
212 described in [RFC2616].**</big>**

213 ***3.1.2 Overview***

214 **<big>**The message protocol for SAML is based on a request-response metaphor. This naturally
215 maps to the HTTP request-response method. So, for most types of interaction between systems, a
216 request message is sent as an HTTP request, and a response message is sent as an HTTP
217 response. There are two parties involved in the interaction: a requester and a responder. There is
218 no provision for intermediaries in the current framework.

219 **</big>**

220 **<big>**In the discussion that follows, the following terms are used:**</big>**

221 **<big>*** request message -or- request: A SAML request XML object.**</big>**

222 **<big>*** response message -or- response: A SAML response XML object.**</big>**

223 **<big>*** HTTP request: An HTTP request, as distinct from a SAML request.**</big>**

224 **<big>*** HTTP response: An HTTP response, as distinct from a SAML response.**</big>**

225 <big>* requester: The party sending the request.</big>
226 <big>* responder: The party sending the response.</big>

227 **3.1.3 HTTP Binding**

228 **3.1.3.1 Connections**

229
230 <big>As with all HTTP connections, the requester will initiate the connection. Connections
231 MUST be one way. Multiple requests and corresponding responses MAY be sent over a single
232 connection, per the HTTP 1.1 specification. The requester MUST only send requests through the
233 connection, and the responder MUST only send responses through the
234 connection.</big><big> </big><big></big>

235 **3.1.3.2 Request Messages**

236 <big>A request message is bound to an HTTP request.
237 </big>

238 <big>The request MUST use the POST method. The HTTP version MUST be one of "1.0" or
239 "1.1".</big>

240
241 <big>The request MUST have a Content-Type of "application/saml+xml".

242 *ISSUE:[BINDINGS-02] We will need to register "application/saml+xml" as a new MIME sub-*
243 *type following RFC3023. Alternatively, can we live with "text/xml" for SAML 1.0?</big>*

244
245 <big>The content of the HTTP request MUST be exactly one request message. Additional
246 content MUST NOT be included in the HTTP request.</big>

247
248 <big>The Host, Date, Content-Type and Content-Length headers MUST be provided in the
249 HTTP request and be correct. A Connection header may be added as is standard in HTTP 1.1.

250
251 <big>Additional HTTP headers MAY be provided, but parties in the conversation MUST ignore
252 those other headers.</big>

253 <big>[Rationale: many existing HTTP libraries will add additional headers to an HTTP request.
254 The intent is to ensure a minimal number of headers required to handle the binding, without
255 requiring that implementations write their own HTTP code.]</big>

256
257 <big>Content-Encoding or Transfer-Encoding schemes MUST NOT be used.</big>

258 <big>[Rationale: SAML messages are relatively small and should not require chunked encoding
259 or compression. Forbidding Content- or Transfer-Encoding will allow implementers to safely
260 ignore these fairly advanced and costly HTTP features.]</big>

3.1.3.3 Response Messages

<big>If a request can be handled and generates a response, the response will be bound to an HTTP response message. If the responder cannot or will not generate a SAML response, the responder MUST send one of the HTTP error responses defined below. The rest of this section will treat only successful responses.

<big>[Note that success, in this context, means that a SAML response was generated. It does not mean that the request was fulfilled or have domain level meaning, such as that authorization was granted, etc. The SAML response may have failure notifications per the SAML protocol.]</big>

<big>The HTTP response MUST have a status code of 200. The HTTP version MUST be one of "1.0", "1.1".</big>

<big>The response MUST have a Content-Type of "application/saml+xml".</big>

<big>The content of the HTTP response MUST be exactly one response message. Additional content MUST NOT be included in the HTTP response.</big>

<big>The Host, Date, Content-Type and Content-Length headers MUST be provided in the HTTP response and be correct. A Connection header may be added as noted above in section 4.1.</big>

<big>Additional HTTP headers MAY be provided, but parties in the conversation MUST ignore those other headers.</big>

<big>Content-Encoding or Transfer-Encoding schemes MUST NOT be used.</big>

3.1.3.4 Authentication and Message Integrity

Authentication of parties and message integrity of both requests and responses is REQUIRED and may be handled in one of two ways.

3.1.3.4.1 XML Signature

If this technique is used, an XML digital signature MUST be added to the entire request or response.

ISSUE:[BINDINGS-3] *We need a SAML Profile of DSIG to characterize acceptable forms of signing (enveloped, enveloping, detached) and acceptable keyinfo contents.*

300 **3.1.3.4.2 HTTP/S with Certificates**

301
302 Alternately, the HTTP conversation may be conducted over a Secure Sockets Layer (SSL)
303 connection. In this case, both parties (requester and responder) MUST provide digital certificates
304 for the SSL layer.
305

306 **3.1.3.5 Message Confidentiality**

307
308 HTTP/S MAY be used preserve message confidentiality. A server-side certificate is required.

309 **3.1.3.6 Errors**

310
311 The following error messages may be sent by the responder for a SAML message. Note that in
312 the following section, the error text is not normative, but gives an indication of what the error
313 code means. Only the error number is normative.
314

315 For all status values besides "200", the "Connection: close" header MUST be sent, and the
316 connection between requester and responder MUST be closed.
317

318 **3.1.3.6.1 200 OK**

319
320 The responder received the request and successfully generated a response. The response may
321 contain a SAML error code or further SAML information. The meaning of the 200 message is
322 "more info in SAML content."
323

324 **3.1.3.6.2 400 Bad Request**

325
326 The responder received the request, but the request was ill-formed at the HTTP level in some
327 way. The content of the Response is undefined, but it SHOULD NOT be a SAML message. The
328 content of the Response MAY be a stock piece of HTML or plain text explaining the nature of
329 the error.

330 [Rationale: Some HTTP server software will add stock explanations for error status codes.]

331 This result code is appropriate for requests with bad HTTP headers, HTTP methods other than
332 "POST", or an ill-formed HTTP request.

333

334 **3.1.3.6.3 403 Forbidden**

335
336 The responder has received the request, but refuses to perform a SAML message exchange with
337 the requestor. The content of the Response is undefined, but it SHOULD NOT be a SAML
338 message. The content of the Response MAY be a stock piece of HTML or plain text explaining
339 the nature of the request.

340

341 ***3.1.3.6.4 500 Internal Server Error***

342

343 The responder has received the request but has failed to produce a response, due to internal
344 error. The content of the Response is undefined, but it
345 SHOULD NOT be a SAML message. The content of the Response MAY be a stock piece of
346 HTML or plain text explaining the nature of the request.

347

348 **3.2 SOAP 1.1**

349 ***3.2.1 Introduction***

350

351 SOAP (Simple Object Access Protocol) 1.1 is a standard proposed by Microsoft, IBM, and other
352 contributors for RPC-like interactions using XML. It defines a mechanism for defining messages
353 in XML, and for sending them over HTTP. Since its introduction, it has had increased attention,
354 and it is expected to provide the foundation for many future Web-based services.

355

356 SOAP 1.1 has three main parts. One is a message format that uses an envelope and body
357 metaphor to wrap XML data for transmission between parties. The second is a restricted
358 definition of XML data for making strict RPC-like calls through SOAP, without using a
359 predefined XML schema. Finally, it provides a binding for SOAP messages to HTTP and
360 enhanced HTTP.

361

362 This document describes how to use SOAP to send and receive SAML messages. An additional
363 section of the SAML specification ("SOAP Profile") defines how to use SAML as an
364 authentication mechanism for SOAP. In other words, this section describes using SAML over
365 SOAP, and that section describes using SAML for SOAP.

366

367 Like SAML, SOAP can be used over multiple underlying transports. This document does not
368 address the use of underlying transports directly, although it makes recommendations for some
369 transports in addressing message integrity and confidentiality concerns.

370 **3.2.2 Overview**

371
372 SOAP messages consist of three elements: an envelope, header data, and a message body. SAML
373 messages (queries and responses) are enclosed in the SOAP message body.

374 SOAP 1.1 also defines an optional data encoding system. This system is not used for the SOAP
375 protocol binding for SAML. This means that SAML messages can be transported using SOAP
376 without re-encoding from "standard" SAML to a SAML-like SOAP encoding.

377 The system model used for SAML conversations over SOAP is a simple request-response model.
378 A sending party sends a SAML query in the body of a SOAP message. The receiving party
379 processes the SAML query and returns a SAML query response in the body of another SOAP
380 message.

381 A brief glossary:

382 SAML conversation: an exchange of a SAML query and a SAML response.

383 sending party: The party sending a message.

384 receiving party: The party receiving a message.

385 querying party: The party sending a query message.

386 responding party: The party sending a response.

387 **3.2.3 SOAP Binding**

388 **3.2.3.1 Namespaces**

389
390 All SAML messages encoded in SOAP MUST include XML namespace qualifiers, as specified
391 by the core assertions and messages definition.

392 [Rationale: Some SOAP message processors require a namespace. Also, the namespace prevents
393 conflicts with other standards and schemata.]

394

395 **3.2.3.2 Headers**

396
397 The sending party in a SAML conversation over SOAP MAY add arbitrary headers to the SOAP
398 message.

399 [Rationale: some SOAP software and libraries may add headers to a SOAP message that are out
400 of the control of the SAML-aware process. Also, some headers may be needed for underlying
401 protocols that require routing of messages.]

402 The receiving party MAY NOT require any headers for the SOAP message.

403 [Rationale: requiring extra headers will cause fragmenting of the standard and will hurt
404 interoperability.]

405 **3.2.3.3 SAML Queries**

406
407 A SAML query is stored as the child of the <SOAP:body> element of a SOAP message. The
408 querying party **MUST** send one SAML query. The querying party **MUST NOT** send more than
409 one SAML query per SOAP message. The querying party **MUST NOT** include any additional
410 XML elements in the SOAP body.

411 On receiving a SAML query as a SOAP message, the receiving party **MUST** return either a
412 SAML query response or a SOAP fault code.

413 **3.2.3.4 SAML Query Responses**

414
415 A SAML query response is stored as the child of the <SOAP:body> element of a SOAP
416 message. The message **MUST** contain exactly one SAML query response. The querying party
417 **MUST NOT** include any additional XML elements in the SOAP body.

418 On receiving a SAML query response in a SOAP message, the querying party **MUST NOT** send
419 a fault code or other error messages to the sending party.

420 [Rationale: The format for the message interchange is a simple request-response. Adding
421 additional error conditions, notifications, etc. would needlessly complicate the protocol.]

422 **3.2.3.5 Fault Codes**

423
424 If a responding party cannot, for some reason, process a SAML query, it should return a SOAP
425 fault code. Fault codes **MUST NOT** be sent for errors within the SAML problem domain, e.g. as
426 a signal that the subject is not authorized to access an object in an authorization query.

427 The four fault codes (VersionMismatch, MustUnderstand, Client, Server) defined by SOAP 1.1
428 are sufficient to define any SOAP-related errors. Responding parties **MUST NOT** use any
429 additional fault codes, or sub-defined fault codes, in a fault response.

430 Responding parties **MAY** provide additional fault information, such as descriptions and details,
431 as defined by SOAP.

432 [Rationale: some SOAP processors may add fault information automatically.]

433 **3.2.3.6 Authentication and Message Integrity**

434 Authentication of parties and message integrity of both requests and responses is **REQUIRED**
435 and may be handled in one of the following ways.

436 ***3.2.3.6.1 XML Digital Signature***

437
438 To ensure authentication and message integrity, the parties in a SAML conversation **MAY** add a
439 XML Digital Signature to the SAML query and SAML response.

440 3.2.3.6.2 HTTP/S with Certificates

441
442 Alternately, the parties MAY use the underlying transport of the SOAP conversation to ensure
443 authentication and message integrity. For SOAP messages sent over HTTP, this would be
444 HTTP/S with client and server certificates.

445 3.2.3.7 Confidentiality

446 Unfortunately, at the SOAP level itself there is no standard message oriented technique for
447 confidentiality. This will only be possible when XML-ENCRYPTION standard becomes
448 available. So for the near future, we have to depend on facilities provided by the substrate
449 protocol over which SOAP is layered.

450 For the case where SOAP messages are used over HTTP, this would be HTTP/S with the use of
451 a server-side certificate.</big>

452 4 Profiles</big>

453 4.1 Web Browser

454 4.1.1 Background

455
456 The web browser profile utilizes terminology taken from Use Case 1, Scenario 1-1 and Scenario
457 1-2 of [OASIS-Use-Case]. This material should be reviewed at this point.

458 <big>The user is utilizing a standard commercial browser and has authenticated to a *source site*.
459 We assume that the source site has some form of security engine in place that can track locally
460 authenticated users [websso]. Typically, this takes the form of a session which may be
461 represented by an encrypted cookie or an encoded URL or by the use of some other technology
462 [session]. This is a strong assumption but one which is met by a large class of security engines.

463 At some point, the user attempts to access a *target* resource available from the *destination site*
464 and subsequently through one or more steps (e.g., re-direction) arrives at an *inter-site transfer*
465 *URL* at the source site. Starting at this point, the SAML web browser profiles describe a
466 canonical sequence of HTTP protocol exchanges that transit the user browser to a distinguished
467 *assertion consumer URL* at the destination site. Information about *SAML assertions* associated
468 with the user and the desired target are conveyed, from the source to the destination site, by the
469 protocol exchange.

470 The destination site can examine both the assertions and target information and determine
471 whether to allow access to the target resource, thereby achieving web single sign-on for
472 authenticated users originating from the source site. Often, the destination site also utilizes a
473 standard security engine that will create and maintain a session, possibly utilizing information
474 contained in the source site assertions, for the user at the destination site.

475 *ISSUE:[BINDINGS-4] The use-case document and those given above, do not describe the case*
476 *where the user contacts the destination site first and is then sent to the source site. Do we need to*
477 *include this step in SAML 1.0 web browser profiles?*

478 **4.1.2 Relevant Technology**

479 We describe two HTTP-based techniques available for conveying information from one site to
480 another via a stock commercial browser. We do not discuss the use of cookies, as these impose
481 the limitation that both the source and destination site belong to the same "cookie domain".

- 482 • *Form POST*: SAML assertions are uploaded to the user browser within a HTML Form
483 [HTML] and conveyed to the destination site as part of a HTTP POST payload when the user
484 "submits" the form,
- 485 • *SAML Artifact*: A "small", bounded-size SAML artifact, which unambiguously identifies an
486 assertion, is carried as part of a URL query string and conveyed via re-direction to the
487 destination site; the destination site must acquire the referenced assertion by some further
488 steps. Typically, this involves the use of a registered SAML protocol binding.

489
490 The need for a "small" SAML artifact is motivated by restrictions on URL size imposed by
491 commercial web browsers. While [RFC2616] does not specify any restrictions on URL length, in
492 practice commercial web browsers and **</big><big></big><big>**application servers impose size
493 constraints on URLs (maximum size of 2000 characters [Appendix A]). Further, as developers
494 will need to estimate and set aside URL "real-estate" for the artifact, it is important that the
495 artifact have a bounded size (predefined maximum size). These measures ensure that the artifact
496 can be reliably carried as part of the URL query string and thereby transferred from source to
497 destination site.

498 **4.1.3 SAML artifact structure**

499
500 Depending on upon the level of security desired and associated profile protocol steps, many
501 viable architectures may be proposed for the SAML artifact ([Core-Assertions-Examples, Shib-
502 Marlena]. We accommodate variability in SAML artifact architecture by a mandatory two byte
503 artifact type code in the representation:

```
504  
505 <SAML_artifact> :=  
506     B64 representation of <TypeCode> <artifact contents>  
507     <TypeCode> := Byte1Byte2
```

508
509 We describe one specific architecture with the property that it is simple to implement but at the
510 same time its use has adequate safeguards against attacks such as artifact forgery, browser state
511 exposure and impersonation.

```
512  
513 <TypeCode> := 0x0001  
514 <RemainingArtifact> := <PartnerID> <AssertionHandle>  
515 <PartnerID> := byte1byte2byte3byte4
```

516 **<AssertionHandle> := byte1byte2byte3byte4byte5byte6byte7byte8**

517

518 <PartnerID> is a four byte value used by the destination site to determine source site identity as
519 well as the URL (or address) for the “assertion lookup” service. This information needs to have
520 been agreed upon between the source and destination site using an out-of-band technique. On
521 receiving the SAML artifact, the destination site determines if the <PartnerID> belongs to a
522 valid partner, accesses the “assertion lookup” service URL and invokes it with the
523 <AssertionHandle> value as an argument.

524

525 <AssertionHandle> is an eight byte value which MUST be drawn from a random number
526 sequence [RFC1750] generated at the source site and serves to identify the assertion to the
527 source site. The <AssertionHandle> value is completely opaque to the destination site; further,
528 its construction ensures that it has no predictable relationship to the contents of the referenced
529 assertion at the source site.

530

531 **4.1.4 Profile Overview**

532 In this section, we describe two distinct web browser profiles: one based on a SAML artifact and
533 one based on form POST. The SAML artifact profile involves two sub-cases: a pull case
534 (corresponds to Scenario 1-1 from [OASIS-Use-Case]) and a push case (corresponds to Scenario
535 1-2 from [OASIS-Use-Case]). For each type of profile, a section describing the threat model and
536 relevant counter-measures is also included.

537 Two types of information are communicated through the web browser profiles:

538

539 (1) information about the “target” of interest to the user. This is essentially some contextual
540 information originating from the source web site. Typically, this takes the form of a URL at the
541 destination web site but more generally it could take the form of a category or resource name.
542 The destination site may use the target information to present an appropriate category of
543 resources to the user (e.g., redirect to the target URL) once sign-on has been completed.

544

545 (2) information describing one or more SAML assertions. There are two restrictions here. First,
546 each such assertion MUST be a “bearer” assertion. Second, one (and only one) assertion MUST
547 take the form of an authentication assertion.

548 **4.1.4.1 SAML Artifact (Pull)**

549 **</big>**

550 **<big>**This profile consists of a single interaction between three parties (source site, user
551 equipped with a browser, destination site), with a nested sub-interaction between two parties
552 (source site, destination site). We refer to the sub-interaction as an *assertion pull* interaction. The
553 interaction sequence is diagrammed in Figure 1.

554 The user has authenticated to the source web site and subsequently visits an inter-site transfer
555 URL with information about the desired target on the URL query string (step (1)). As this step is

556 over the open internet, confidentiality is required, and the inter-site transfer URL MUST be
557 exposed over HTTPS (HTTP over server-side SSL). Otherwise, the artifact(s) returned on (step
558 (2)) will be available in plain text to any attacker.

559
560 The inter-site transfer URL redirects the user (step (2) to the destination URL with target and
561 one or more SAML artifacts carried on the URL query string.

562 In response, the user browser attempts to access the destination URL (step (3)) and delivers both
563 the destination URL, the SAML artifact(s) and target to (a web server at) the destination site. As
564 this step is over the open internet, confidentiality is required, and the destination URL MUST be
565 exposed over HTTPS (HTTP over server-side SSL). This is because a SAML artifact represents
566 a bearer token, and its disclosure may allow an adversary to impersonate the user.

567 If the destination site is unable to process this information it MUST return a HTTP "400 Bad
568 Request" error code to the browser (step 6)). Otherwise, it MUST carry out the *assertion pull*
569 interaction (steps (4) and (5)) described below, and obtain assertions from the source site.

570
571 Thereafter, the destination site may utilize communicated assertions and target information,
572 further interaction with the user and other information and make an access control judgement. If
573 the user is refused access to the desired resource, the destination site MUST return a HTTP "403
574 Forbidden" error code to the browser (step (6)).

575 The assertion pull interaction consists of a SAML message exchange between source and
576 destination site (steps (4) and (5)) utilizing a registered SAML protocol binding. The destination
577 site sends a *<samlp:Request>* message to the source site, containing SAML artifacts which
578 identify SAML assertions at the source site. If the source site can find the required assertions it
579 responds with a *<samlp:Response>* message with the desired assertion. Otherwise, it returns an
580 "assertion not found" error to destination site.

581 The selected SAML protocol binding for assertion pull MUST support confidentiality and bi-
582 lateral authentication. The source site MUST implement a SAML HTTP binding with support for
583 confidentiality (HTTPS); support for other protocol bindings is not mandatory.

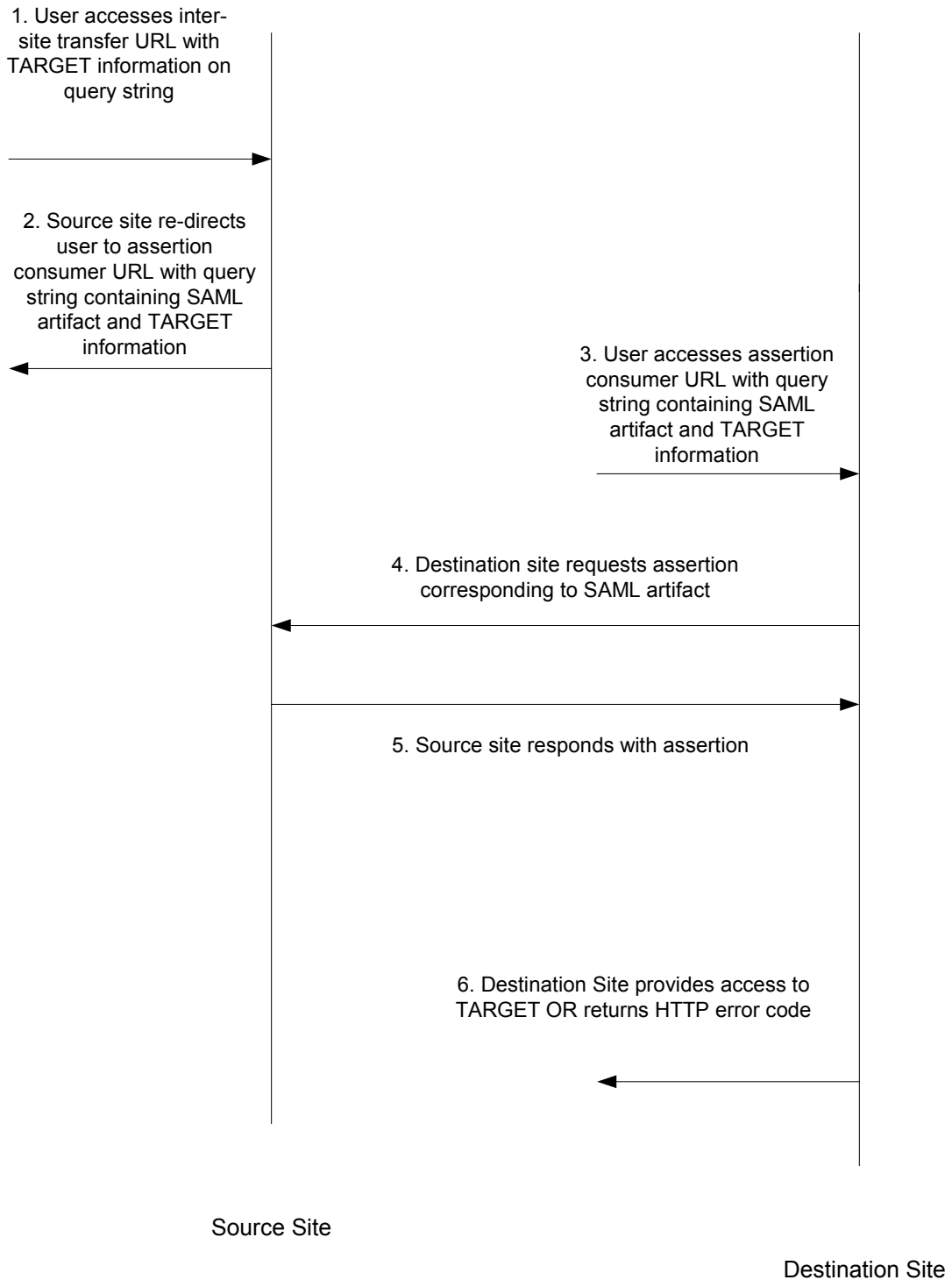


Figure 1: web Browser Profile: SAML Artifact (Pull)

Action	HTTP
(1)	GET https://www.example.com/<inter-site-transfer URL>?TARGET=<target>..
(2)	HTTP 1.1 301 GET https://destination_URL?SAMLart=<artifact body>?TARGET=<target>..
(3)	GET https://destination_URL?SAMLart=<artifact body>?TARGET=<target>..
(4)	<samlp:Request> message is sent to source site with artifact information over selected protocol bindings.
(5)	<samlp:Response> message with an assertions is returned to destination site over selected protocol binding.
(6)	User is given access to TARGET OR “400 Bad Request” is returned OR “403 Forbidden” is returned

585
586

587 The source and destination sites MUST implement the following additional restrictions when
588 processing SAML artifacts:

589

- 590 1. The SAML artifact corresponding to an Authentication Assertion MUST be "one-time use";
591 once the user completes step (6) above, any repetition of step (3) MUST fail with the
592 destination site returning HTTP code “403 Forbidden”.
593
- 594 2. The destination site MUST implement a “one-time lookup” property for any authentication
595 assertion exposed via a SAML artifact. Many simple implementations meet this requirement:
596 for example deleting the relevant authentication assertion from persistent storage at the
597 source site after first successful lookup
- 598 3. A successful <samlp:Response> message is returned from the source site only if the
599 <samlp:Request> message originates from the destination site to whom the artifact was
600 issued. Thus, step (4) above would complete successfully at most once and only if originating
601 from the (unique) destination site.
602

603 4.1.4.2 SAML Artifact (Push)

604 Figure 2 describes the interaction sequence for the “push” case of the SAML artifact profile. The
605 number of parties and interactions is very similar to the “pull” case. An authenticated user visits
606 an inter-site transfer URL with information about the destination site target resource as part of
607 the URL query string (step (1)). The source site and destination site participate in a message
608 exchange (steps (2) and (3)) and determine whether the user has the right to access the desired
609 target resource. The source site sends the destination site a <samlp:Query> message of type

610 AuthorizationQueryType and the destination site returns a <samlp:Response> message.
 611 The response message carries either SAML artifact(s) or an authorization failure code.

612 The message exchange utilizes a SAML protocol binding which MUST support bilateral
 613 authentication and confidentiality. The destination site MUST implement a SAML HTTP
 614 binding with support for confidentiality (HTTPS); support for other protocol bindings is not
 615 mandatory.

616 The destination site (step (4)) either returns an HTTP error-code to the user, or, re-directs the
 617 user to the assertion consumer URL at the destination site. Target and SAML artifact information
 618 are carried as part of the URL query string. The user browser accesses the assertion consumer
 619 URL at the destination site (step(5)) delivering target and artifact information. The destination
 620 site “looks up” the assertions corresponding to the delivered artifact and provides access to the
 621 desired target (step (6)).

622 As interactions with the inter-site transfer and assertion consumer URLs is over the open
 623 internet, confidentiality is required, and both URLs MUST be exposed over HTTPS (HTTP over
 624 server-side SSL).

625

Action	HTTP
(1)	GET https://www.example.com/<inter-site-transfer URL>?TARGET=<target>..
(2)	<samlp:Request> message of AuthorizationQueryType is sent to destination site utilizing a protocol binding supporting confidentiality.
(3)	<samlp:Response> message with a SAML artifact(s) or an access denied response is returned to destination site utilizing a protocol binding supporting confidentiality.
(4)	HTTP 1.1 301 GET https://assertion_consumer_URL?SAMLart=<artifact body>?TARGET=<target>.. OR “403 Forbidden” returned to browser.
(5)	GET https://assertion_consumer_URL?SAMLart=<artifact body>?TARGET=<target>..
(6)	User is given access to TARGET OR “400 Bad Request” is returned

626

627 The destination site MUST implement the following restrictions when processing SAML
 628 artifacts:

629

630 The SAML artifact corresponding to an authentication assertion MUST be "one-time use"; once
 631 the user completes step (6) above, any repetition of step (5) MUST fail with the destination site
 632 returning HTTP code “403 Forbidden”.

633

634

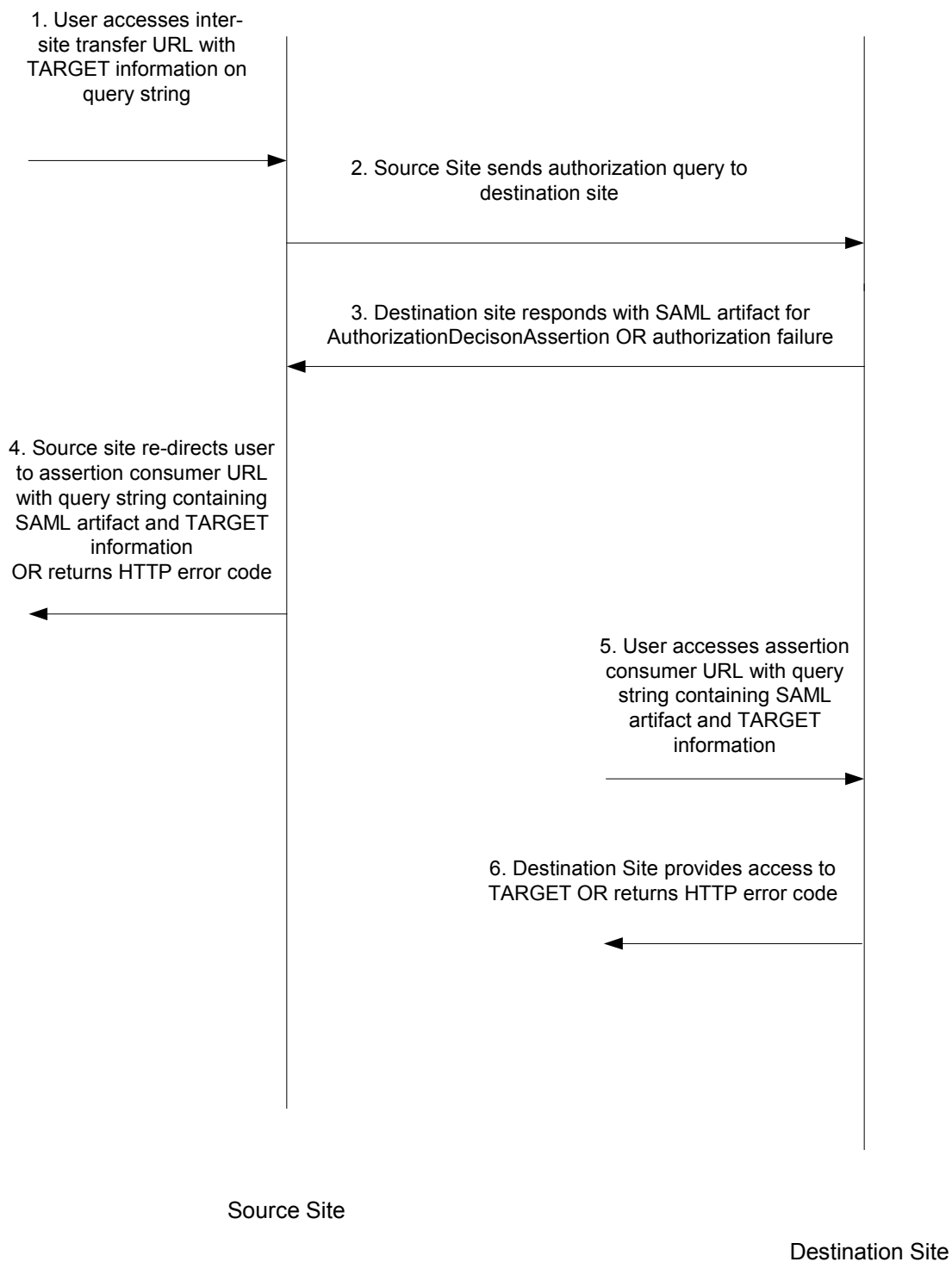


Figure 2: web Browser Profile: SAML Artifact (Push)

636 **4.1.4.3 Form POST**

637 Figure 3 provides a description of a web browser profile based upon the use of “POST” to
 638 convey SAML assertions from source to destination site [S2ML, Anders-Browser-Profile]. An
 639 authenticated user visits an inter-site transfer URL with information about the target as part of
 640 the URL query string (step (1)). The source site generates an HTML page containing a form with
 641 one or more embedded SAML assertions and target information (step (3)). The user browser
 642 “clicks on” the form SUBMIT button and navigates to the assertion consumer URL at the
 643 destination site (step (4)). The destination site scrutinizes the posted assertion and target
 644 information and determines whether to allow the user access to the target resource (step (5)).

645
 646 As interactions with the inter-site transfer and assertion consumer URLs is over the open
 647 internet, confidentiality is required, and both URLs MUST be exposed over HTTPS (HTTP over
 648 server-side SSL).

649
 650

Action	HTTP
(1)	<code>GET</code> <code>https://www.example.com/<inter-site-transfer URL>?TARGET=<target>..</code>
(2)	<code>HTTP 1.1</code> <code>Content-Type: application/x-www-form-urlencoded</code> <code>Content-length:...</code> <code><BODY></code> <code><FORM METHOD="post" ACTION="assertion_consumer_URL"></code> <code><INPUT TYPE="submit" NAME="button" VALUE="submit"></code> <code><INPUT TYPE="hidden" NAME="SAMLAssertion" VALUE="B64 (SAML Assertion)"></code> <code><INPUT TYPE="hiddent" NAME="TARGET" VALUE="<target>"></code> <code></FORM></code> <code></BODY></code>
(3)	This step may be eliminated in a Javascript-enabled browser. See Appendix B.
(4)	<code>POST assertion_consumer_URL</code> [standard POST payload corresponding to form in (2)]
(5)	User is given access to TARGET OR “403 Forbidden” is returned

- 651
- 652 Notes:
- 653
- 654 1. All SAML assertions communicated to the destination site using the POST web browser
 655 profile MUST be digitally signed by the issuing party.
 656
 - 657 2. The destination site MUST ensure a “single use” policy for authentication assertions
 658 communicated using form POST. The implication here is that the destination site will need to

659 be stateful. A simple implementation maintains a table of pairs:

660

661 Assertion Id, Time at which entry is to be deleted

662

663 The time at which an entry is to be deleted is based upon the authentication assertion life-

664 time. As authentication assertions are recommended to have short life-times in the web

665 browser context, such a table would be of manageable size.

666

667 3. Privacy reasons may require that SAML assertions be encrypted. This is an area that requires
668 further investigation.

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

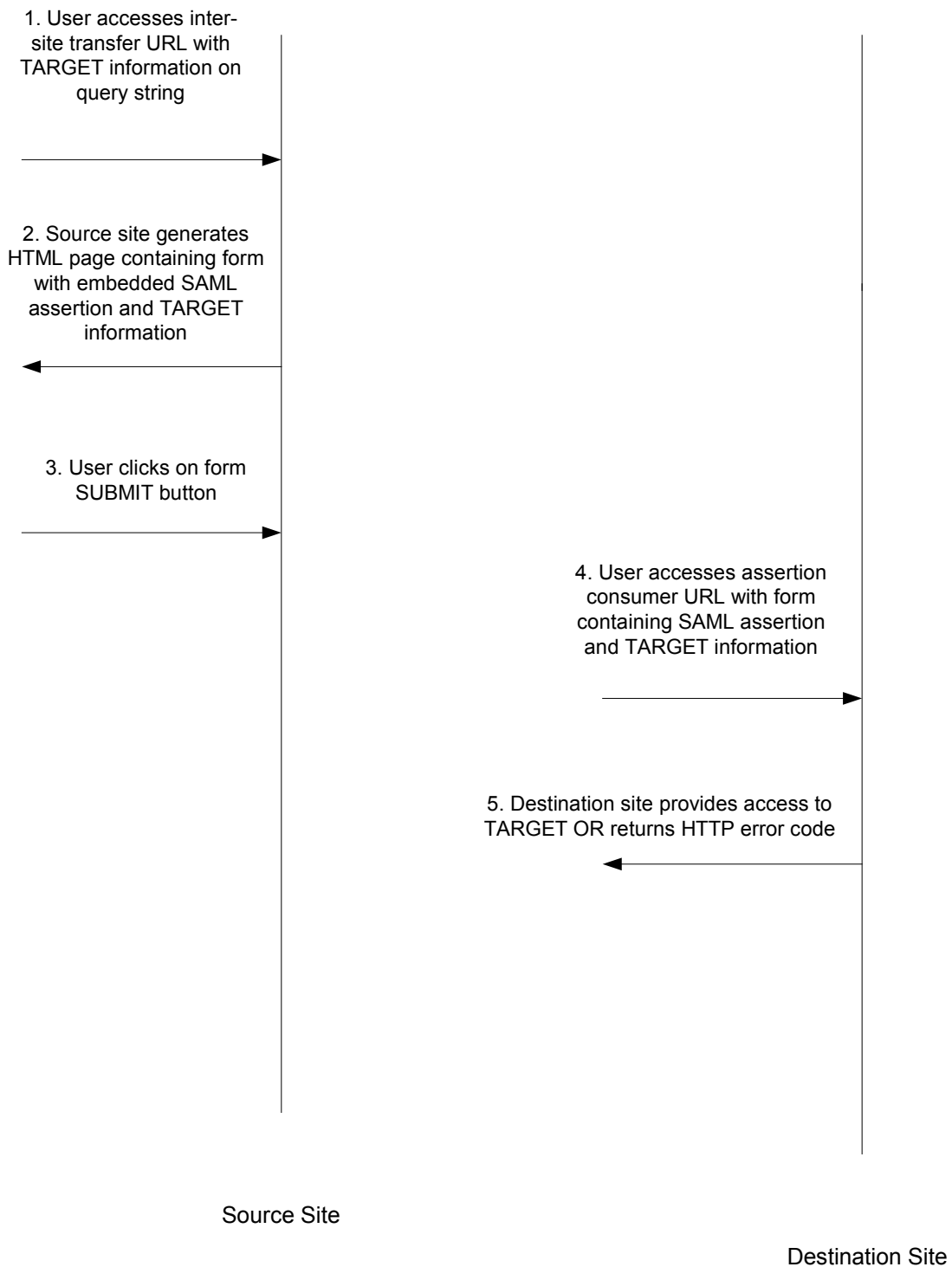


Figure 3: Web Browser Profile: POST

694 **4.1.5 Threat Model and Counter-Measures**

695

696 This section utilizes materials from [Shib-Marlena].

697 **4.1.5.1 Stolen artifact or assertion**

698 1. If a malicious user (MAL) can copy the real user’s SAML artifact or SAML assertion (Form
699 POST), then the MAL could construct a URL with the real user’s SAML artifact or POST
700 body and be able to impersonate the user at the destination site.

701

702 Counter-Measure:

703 (a) SAML artifact: SAML artifacts communicated through a web browser profile must
704 always reference a SAML authentication assertion. An authentication assertion
705 communicated through a web browser profile **MUST** include (1) issue instant and (2)
706 validity period. It **MAY** include the IP address of the user. It is recommended that a SAML
707 authentication assertion communicated through a web browser profile have the shortest
708 possible validity period consistent with successful functioning of the profile. This is
709 typically of the order of a few minutes.

710

711 The destination site should check the browser IP address against the IP address contained in
712 the authentication assertion (if available) and also ensure that the validity period of the
713 assertion has not expired.

714

715 (b) Form POST: As above.

716

717 2. Since the destination site obtains “bearer” SAML artifacts or SAML assertions from the user
718 via a web browser profile, a malicious site could impersonate the user at some “new”
719 destination site. The new destination site would believe the malicious site to be the user.

720

721 Counter-Measure:

722 (a) SAML artifact: The new destination site must obtain the SAML assertions
723 corresponding to the SAML artifacts from the source site through a bilaterally authenticated
724 channel. The SAML artifact profile requires that the source site only allow access to
725 assertions to those sites to which it has directly provided the corresponding SAML artifacts.

726

727 (b) Form POST: A SAML authentication assertion communicated through a form POST is
728 always digitally signed and **MUST** include the <AudienceRestrictionCondition> element.
729 The destination site must check the <AudienceRestrictionCondition> element to ensure that
730 its value matches the destination site expectations. It is strongly recommended that
731 assertions communicated through the web browser profile have extremely “narrow” values
732 for this field (e.g., each destination site has a unique <AudienceRestrictionCondition>
733 value).

734

735 **4.1.5.2 Forged SAML artifact or Assertion**

736 A MAL could forge a SAML artifact (SAML artifact) or SAML assertion (form POST).

737 Counter-Measure: The POST browser profile requires SAML assertions to be signed, thus
738 providing both message integrity and authentication. The destination site must always verify the
739 signature and ensure that it corresponds to the assertion issuer.

740 A SAML artifact is a eight byte (sixty-four bit) number drawn from a random sequence. A MAL
741 could attempt to repeatedly “guess” a valid SAML artifact value (one that corresponds to an
742 existing assertion at a source site) but given the size of the value space (2^{64} possible values)
743 would likely require a very large number of failed attempts.

744 **4.1.5.3 Browser State Exposure**

745 Both the SAML artifact profile and the POST browser profile involve upload of SAML artifacts
746 or assertions to the web browser from a source site. This information is available as part of the
747 web browser state and is usually stored in persistent storage on the user system in a completely
748 unsecured fashion. The threat here is that the assertion or artifact may be “re-used” at some later
749 point in time.

750

751 Counter-Measure: The “one-use” property of SAML artifacts corresponding to authentication
752 assertion presentation and lookup ensures that an authentication assertion artifact may not be re-
753 used from a browser. The form POST case similarly includes a requirement that an
754 authentication assertion cannot be re-presented at the destination site. The web browser profile
755 always requires an authentication assertion.

756

757

758

759

760

761

SOAP Profile (Use-Case 3)



Sending Party attaches SAML assertions to

763 {PRIV
764 ATE "TYPE=PICT;ALT=Figure 2: SOAP Message Transfer"}
765

766 **4.2.1 Overview**

767
768 The SOAP profile for SAML is based on a single interaction between a sending party and a
769 receiving party. The sending party adds with one or more SAML assertions to a SOAP document
770 and sends the message to the receiving party. The receiving party processes the SAML assertion
771 and either returns an error or goes on to process the message in the standard way. The message
772 may be sent over any protocol for which a SOAP protocol binding is available [SOAP].

773 SOAP provides a flexible header mechanism, which may be (optionally) used for extending
774 SOAP payloads with additional information. A header entry is identified by its fully qualified
775 element name, which consists of the namespace URI and the local name. All immediate child
776 elements of the SOAP Header element MUST be namespace-qualified.

777

778 **4.2.2 SOAP Headers and Error Processing**

779

780 SAML assertions MUST be contained within the SOAP <Header> element contained within the
781 SOAP <Envelope> element. Two standard SOAP attributes are available for use with header
782 elements: actor and mustUnderstand. Use of the actor attribute is application dependent and
783 no normative use is specified herein.

784

785 The SOAP mustUnderstand global attribute can be used to indicate whether a header entry
786 is mandatory or optional for the recipient to process. SAML assertions MUST have the
787 mustUnderstand attribute set to 1; this ensures that a SOAP processor to which the message is
788 directed must be able to successfully process the SAML assertions or return a SOAP message
789 with <Fault> element as the message body. The returned <Fault> element takes the form:

790

```
791 <Fault>  
792   <Faultcode>mustUnderstand</Faultcode>  
793   <Faultstring>...</Faultstring>  
794 </Fault>
```

795

796

797 If the receiving party is able to successfully process the attached SAML assertions, and based on
798 their contents does not further process the body of the SOAP message, it MUST return a SOAP
799 message with <Fault> element as the message body. The returned <Fault> element takes the
800 form:

801

802

```
803 <Fault>  
804   <Faultcode>Client.SAML</Faultcode>  
805   <Faultstring>Subject not authorized</Faultstring>  
806 </Fault>
```

807

808 SAML assertions contained with a SOAP message MUST be digitally signed. This ensures that
809 the receiving party can authenticate the issuer and ensure that the assertion hasn't been tampered
810 with.

811 **4.2.3 Confidentiality**

812 In the absence of a mature [XML-Encryption] specification, confidentiality has to be ensured by
813 selection of a "substrate" SOAP protocol binding which preserves confidentiality. This would
814 include, for example, HTTPS or S/MIME. MANDATORY TO IMPLEMENT: HTTPS with
815 server-side certificates.

816 **4.2.4 Example**

817

818 The following example illustrates the addition of SAML assertions to a SOAP message:

819 {PRIVATE "TYPE=PICT;ALT=Figure 3: SOAP document with inserted assertions"}

```

820 <SOAP-ENV:Envelope xmlns:SOAP-ENV=http://schema.xmlsoap.org/soap/envelope/>
821
822 <SOAP-ENV:Header xmlns:SAML="..." >
823     <SAML:Assertion mustUnderstand=1>...</SAML:Assertion>
824     <SAML:Assertion mustUnderstand=1>...</SAML:Assertion>
825 </SOAP-ENV:Header>
826 ...
827 <SOAP-ENV:Body>
828     <message_payload/>
829 </SOAP-ENV:Body>
830 </SOAP-ENV:Envelope>
831
832
833

```

834 **4.2.5 Integrity of Assertion Attachment**

835 When processing an assertion, the receiving party MUST check the integrity of assertion
836 attachment to the SOAP message. In general, the mandatory `<SubjectConfirmation>` element
837 contained within assertions the may be used to specify this information. Two specific techniques
838 are called as out as mandatory to implement.

839 **4.2.5.1 Digest of SOAP Message**

```

840 <SubjectConfirmation>
841     <AuthenticationMethod>DSIG-DIGEST</AuthenticationMethod>
842     <SubjectConfirmationData>
843         <dsig:CanonicalizationMethod>...</dsig:CanonicalizationMethod>
844         <dsig:Reference>
845             <dsig:Transforms> . . . </Transforms>
846             <dsig:DigestMethod> . . . </DigestMethod>
847             <dsig:DigestValue> . . . </DigestValue>
848         </Reference>
849     </SubjectConfirmationData>
850 </SubjectConfirmation>

```

851 We plan to re-use elements from [XML-DSIG] to represent the hash of SOAP message within
852 the assertion. The hash value should be computed by EXCLUDING the SAML assertion within
853 which the hash needs to be placed. The `<Transforms>` element is provided for this purpose. We
854 need to (1) ensure that there is DSIG-DIGEST authentication method, and (2) profile the use of
855 the required elements from [XML-DSIG].

856 **4.2.5.2 Digital Signature**

857 Using this technique, the `<SubjectConfirmation>` element carries the sender's public key or
858 X509 certificate within the `<dsig:KeyInfo>` element. The signature itself is carried separately as

859 a <dsig:signature> element as part of the SOAP envelope. This ensures that an assertion can
860 re-used with many different SOAP messages. Note that the signature element is used only for
861 checking integrity of assertion attachment (message integrity). Therefore, there is no requirement
862 that the receiving party validate the key or certificate. This also suggests that servers can
863 generate public/private key pairs and utilize them for this purpose.

```
864 <SubjectConfirmation>  
865   <AuthenticationMethod>DSIG-SIGNATURE</AuthenticationMethod>  
866   <dsig:KeyInfo>...<dsig:KeyInfo>  
867 </SubjectConfirmation>
```

868

869 5 References

870

871 [Anders-Browser-Profile] A suggestion on how to implement SAML browser bindings without
872 using “Artifacts”, <http://www.x-obi.com/OBI400/andersr-browser-artifact.ppt>

873

874 [AuthXML] AuthXML: A Specification for Authentication Information in XML.
875 <http://www.oasis-open.org/committees/security/docs/draft-authxml-v2.pdf>

876

877 [Glossary] OASIS Security Services TC: Glossary.
878 <http://www.oasis-open.org/committees/security/docs/draft-sstc-hodges-glossary-02.html>

879

880 [S2ML] S2ML: Security Services Markup Language, Version 0.8a, January 8, 2001.
881 <http://www.oasis-open.org/committees/security/docs/draft-s2ml-v08a.pdf>

882

883 [draft-sstc-core-07.doc] Security Assertions Markup Language, Version 0.7, May 14th, 2001.
884 <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-07.pdf>

885

886 [Shib] Shibboleth Overview and Requirements
887 [http://middleware.internet2.edu/shibboleth/docs/draft-internet2-shibboleth-requirements-](http://middleware.internet2.edu/shibboleth/docs/draft-internet2-shibboleth-requirements-00.html)
888 [00.html](http://middleware.internet2.edu/shibboleth/docs/draft-internet2-shibboleth-requirements-00.html)[http://middleware.internet2.edu/shibboleth/docs/draft-internet2-shibboleth-requirements-](http://middleware.internet2.edu/shibboleth/docs/draft-internet2-shibboleth-requirements-00.html)
889 [00.html](http://middleware.internet2.edu/shibboleth/docs/draft-internet2-shibboleth-requirements-00.html)

890

891 [Shib-Marlena] Marlena Erdos, Shibboleth Architecture DRAFT v1.1,
892 <http://middleware.internet2.edu/shibboleth/docs/draft-erdos-shibboleth-architecture-00.pdf>

893

894 [RFC2616] Hypertext Transfer Protocol -- HTTP/1.1

895

896 [RFC1750] Randomness Recommendations for Security.
897
898 [SOAP] Simple Object Access Protocol (SOAP) 1.1 , W3C Note 08 May 2000
899
900 [Core-Assertions-Examples] Core Assertions Architecture, Examples and Explanations,
901 <http://www.oasis-open.org/committees/security/docs/draft-sstc-core-phill-07.pdf>
902
903 [XML-DSIG] XML – Signature Syntax and Processing, available from <http://www.w3.org>
904
905 [websso] RL “Bob” Morgan, Interactions between Shibboleth and local-site web sign-on
906 services, [http://middleware.internet2.edu/shibboleth/docs/draft-morgan-](http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-websso-00.txt)
907 [shibboleth-websso-00.txt](http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-websso-00.txt)
908
909 [session] RL “Bob” Morgan, Support of target web server sessions in Shibboleth,
910 [http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-](http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-session-00.txt)
911 [session-00.txt](http://middleware.internet2.edu/shibboleth/docs/draft-morgan-shibboleth-session-00.txt)
912

913 **6 Appendix A**

914 <http://support.microsoft.com/support/kb/articles/Q208/4/27.ASP>
915

916
917 The information in this article applies to:

918 Microsoft Internet Explorer (Programming) versions 4.0, 4.01, 4.01 SP1, 4.01 SP2, 5, 5.01, 5.5
919

920 SUMMARY

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

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

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

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

930 REFERENCES 931

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

934 Additional query words: POST GET URL length

935 Keywords : kbIE kbIE400 kbie401 kbGrpDSInet kbie500 kbDSupport kbie501 kbie550
936 kbieFAQ

937 Issue type : kbinfo

938 Technology :

939 -----

940 Issue: 19971110-3 Product: Enterprise Server

941

942 Created: 11/10/1997 Version: 2.01

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

944 Does this article answer your question?

945 Please let us know!

946

947 Question:

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

950 Answer:

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

952 -----

953 issue: 19971015-8 Product: Communicator, Netcaster

954 Created: 10/15/1997 Version: all

955 Last Updated: 08/10/1998 OS: All

956 Does this article answer your question?

957 Please let us know!

958

959 Question:

960 Is there a limit on the length of the URL string?

961 Answer:

962 Netscape Communicator and Navigator do not have any limit. Windows 3.1 has a restriction of
963 32kb (characters). (Note that this is operating system limitation.) See this article for information
964 about Netscape Enterprise Server.

965 -----

966 `<map></map>`

967 **7 Appendix B**

968
969 Javascript may be used to avoid an additional “submit” step from the user. This material is taken
970 from [Anders-Browser-Profile].

```
971 <HTML>
972 <BODY Onload="javascript:document.forms[0].submit ()">
973 <FORM METHOD="POST" ACTION="Destination-site URL">
974 ...
975 <INPUT TYPE="HIDDEN" NAME="SAMLAssertion" VALUE="Assertion in Base64-
976 coding">
977 </FORM>
978 </BODY>
979 </HTML>
```

980