



1

2 **SAMLv2: HTTP POST “NoXMLdsig”**
3 **Binding**

4 **Draft, 13 June 2006**

5 **Document identifier:**

6 draft-hodges-saml-binding-noxmldsig-01

7 **Location:**

8 **Editors:**

9 Jeff Hodges, NeuStar

10 Scott Cantor, Internet2

11 **Abstract:**

12 This specification defines a SAML HTTP protocol binding, specifically using the HTTP POST
13 method, and not using XML Digital Signature for SAML message and/or SAML assertion data
14 origination authentication. Rather, a “sign the BLOB” technique is employed wherein a conveyed
15 SAML message, along with any content (e.g. SAML assertions) is treated as a simple octet string
16 if it is signed. Security is optional in this binding.

17 **Status:**

18 This is an individually-authored working draft published to the Security Services Technical
19 Committee (SSTC/SAML), and has no official standing.
20 Committee members should submit comments to the security-services@lists.oasis-open.org list.
21 Non-committee members who wish to comment may do so on the [SAML-dev@lists.oasis-](mailto:SAML-dev@lists.oasis-open.org)
22 [open.org](mailto:SAML-dev@lists.oasis-open.org) mailing list (one must be a list subscriber to post. To subscribe, send mail to
23 <mailto:saml-dev-subscribe@lists.oasis-open.org>).

24 **Table of Contents**

25 1 Introduction..... 4
26 1.1 Protocol Binding Concepts..... 4
27 1.2 Notation..... 4
28 2 HTTP POST Binding - NoXMLdsig..... 6
29 2.0.1 Required Information..... 6
30 2.0.2 Overview..... 6
31 2.0.3 RelayState..... 6
32 2.0.4 Message Encoding..... 6
33 2.0.5 Message Exchange..... 7
34 2.0.5.1 HTTP and Caching Considerations..... 8
35 2.0.5.2 Security Considerations..... 9
36 2.0.6 Error Reporting..... 9
37 2.0.7 Metadata Considerations..... 9
38 2.0.8 Example SAML Message Exchange Using HTTP POST..... 9
39 3 References..... 12
40 Appendix B. Acknowledgments..... 15
41 Appendix C. Notices..... 16

42 1 Introduction

43 This specification defines a SAML HTTP protocol binding, specifically using the HTTP POST method, and
44 which specifically does not use XML Digital Signature [XMLSig] for SAML message and/or SAML
45 assertion data origination authentication. Rather, a “sign the blob” technique is employed wherein a
46 conveyed SAML message, along with any content (e.g. SAML assertion(s)) is treated as a simple octet
47 string if it is signed. Security is optional in this binding. The next subsection gives a general overview of
48 SAML Protocol Binding concepts.

49 1.1 Protocol Binding Concepts

50 Mappings of SAML request-response message exchanges onto standard messaging or communication
51 protocols are called SAML *protocol bindings* (or just *bindings*). An instance of mapping SAML request-
52 response message exchanges into a specific communication protocol <FOO> is termed a <FOO> *binding*
53 *for SAML* or a *SAML <FOO> binding*.

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

56 The intent of this specification is to specify a selected set of bindings in sufficient detail to ensure that
57 independently implemented SAML-conforming software can interoperate when using standard messaging
58 or communication protocols.

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

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

64 1.2 Notation

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

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

69 `Example code listings appear like this.`

70 **Note:** Notes like this are sometimes used to highlight non-normative commentary.

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

Prefix	XML Namespace	Comments
saml:	urn:oasis:names:tc:SAML:2.0:assertion	This is the SAML V2.0 assertion namespace [SAMLCore].
samlp:	urn:oasis:names:tc:SAML:2.0:protocol	This is the SAML V2.0 protocol namespace [SAMLCore].
SOAP-ENV:	http://schemas.xmlsoap.org/soap/envelope	This namespace is defined in SOAP V1.1 .

73 This specification uses the following typographical conventions in text: `<ns:Element>`, `XMLAttribute`,
74 **Datatype**, `OtherKeyword`. In some cases, angle brackets are used to indicate non-terminals, rather than
75 XML elements; the intent will be clear from the context.

76 2 HTTP POST Binding - NoXMLdsig

77 The HTTP POST binding, defined in [SAML20Bind], defines a mechanism by which SAML protocol
78 messages may be transmitted within the base64-encoded content of an HTML form control. When using
79 that binding, SAML protocol messages and/or SAML assertions are signed using [XMLSig], which is an
80 XML-aware, XML-based, invasive digital signature paradigm necessitating canonicalization of the
81 signature target.

82 This document specifies an alternative HTTP POST binding where the conveyed SAML protocol
83 messages, and their content – i.e. any conveyed SAML assertions – are signed as simple “blobs” (“binary
84 large objects”, aka binary octet strings).

85 This binding MAY be composed with the HTTP Redirect binding (see Section 3.4 of [SAML20Bind]) and
86 the HTTP Artifact binding (see Section 3.6 of [SAML20Bind]) to transmit request and response messages
87 in a single protocol exchange using two different bindings.

88 2.0.1 Required Information

89 **Identification:** urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST-noxmlsig

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

91 **Description:** Given below.

92 **Updates:** None. Rather, it provides an alternative to the HTTP POST Binding defined in [SAML20Bind]

93 2.0.2 Overview

94 The HTTP POST binding is intended for cases in which the SAML requester and responder need to
95 communicate using an HTTP user agent (as defined in HTTP 1.1 [RFC2616]) as an intermediary. This
96 may be necessary, for example, if the communicating parties do not share a direct path of communication.
97 It may also be needed if the responder requires an interaction with the user agent in order to fulfill the
98 request, such as when the user agent must authenticate to it.

99 Note that some HTTP user agents may have the capacity to play a more active role in the protocol
100 exchange and may support other bindings that use HTTP, such as the SOAP and Reverse SOAP
101 bindings. This binding assumes nothing apart from the capabilities of a common web browser.

102 2.0.3 RelayState

103 RelayState data MAY be included with a SAML protocol message transmitted with this binding. The value
104 MUST NOT exceed 80 bytes in length and SHOULD be integrity protected by the entity creating the
105 message independent of any other protections that may or may not exist during message transmission.
106 Signing is not realistic given the space limitation, but because the value is exposed to third-party
107 tampering, the entity SHOULD ensure that the value has not been tampered with by using a checksum, a
108 pseudo-random value, or similar means.

109 If a SAML request message is accompanied by RelayState data, then the SAML responder MUST return
110 its SAML protocol response using a binding that also supports a RelayState mechanism, and it MUST
111 place the exact data it received with the request into the corresponding RelayState parameter in the
112 response.

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

116 2.0.4 Message Encoding

117 This section describes how to encode SAML messages, and thus any SAML assertions they may contain,
118 into HTML FORM “controls” [HTML401] (Section 17), thus enabling the SAML messages to be transmitted
119 via the HTTP POST method.

120 A SAML protocol message is form-encoded by:

- 121 1. Applying the base-64 encoding rules to the XML representation of the message. The resulting
122 base64-encoded value MAY be line-wrapped at a reasonable length in accordance with common
123 practice.
- 124 2. Encoding the result from the prior step into a “form data set”, in the same fashion as is specified for
125 “successful controls” in [HTML401] (Section 17.13.3), as a form “control value”. The HTML
126 document also MUST adhere to the XHTML specification, [XHTML].
 - 127 1 If the message is a SAML request, then the form “control name” used to convey the SAML
128 message itself MUST be `SAMLRequest`.
 - 129 2 If the message is a SAML response, then the form “control name” used to convey the SAML
130 message itself MUST be `SAMLResponse`.
 - 131 3 Any additional form controls or presentation, other than those noted below for including a
132 signature, MAY be included but MUST NOT be required in order for the recipient to nominally
133 process the message itself.

134 SAML messages, and any SAML assertions conveyed by said messages in this binding, MUST NOT be
135 signed using [XMLSig].

136 **NOTE:** If they are so signed before being processed as defined herein, their XML digital
137 signatures MUST be removed as described in [SAML20Bind] section 3.4.4.1 step 1.

138 Rather, if a SAML message is to be signed – which this binding leaves as a decision of the implementor
139 and/or deployer – it MUST be signed using the technique given below in section 2.0.5. The resultant
140 signature value is conveyed in a form control value named `MsgSig`, and the signature algorithm is
141 conveyed in a form control value named `SigAlg`. These form control values are included in the form data
142 set constructed in step 2 above.

143 If the message is signed, the `Destination` XML attribute in the root SAML element of the SAML
144 protocol message MUST contain the URL to which the sender has instructed the user agent to deliver the
145 message. The recipient MUST then verify that the value matches the location at which the message has
146 been received.

147 If a “RelayState” value is to accompany the SAML protocol message, it MUST be in a form control named
148 `RelayState`, and included in the form data set constructed in step 2 above, and also included in any
149 signed content if the message is signed.

150 The `action` attribute of the form MUST be the recipient's HTTP endpoint for the protocol or profile using
151 this binding to which the SAML message is to be delivered. The `method` attribute MUST be "POST". The
152 `enctype` attribute specifies the form content type and MUST be `application/x-www-form-`
153 `urlencoded`.

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

158 Note that any form control values included MUST be transformed so as to be safe to include in the
159 XHTML document. This includes transforming characters such as quotes into HTML entities, etc.

160 2.0.5 Signature

161 To construct a signature of a SAML message conveyed by this binding:

162 1. The signature algorithm used MUST be identified by a URI, specified according to [XMLSig] or
163 whatever specification governs the algorithm. The following signature algorithms (see [XMLSig])
164 and their URI representations MUST be supported with this encoding mechanism:

- 165 • DSAwithSHA1 – <http://www.w3.org/2000/09/xmldsig#dsa-sha1>
- 166 • RSAwithSHA1 – <http://www.w3.org/2000/09/xmldsig#rsa-sha1>

167 2. A string consisting of the concatenation of the `RelayState` (if present), `SigAlg`, and
168 `SAMLRequest` (or `SAMLResponse`) values (as appropriate), as defined in section 2.0.4 above, is
169 constructed in one of the following ways (each individually ordered as shown):

```
170 SAMLRequest=value&RelayState=value&SigAlg=value
```

```
171 SAMLResponse=value&RelayState=value&SigAlg=value
```

172 3. The resultant octet string is fed into the signature algorithm.

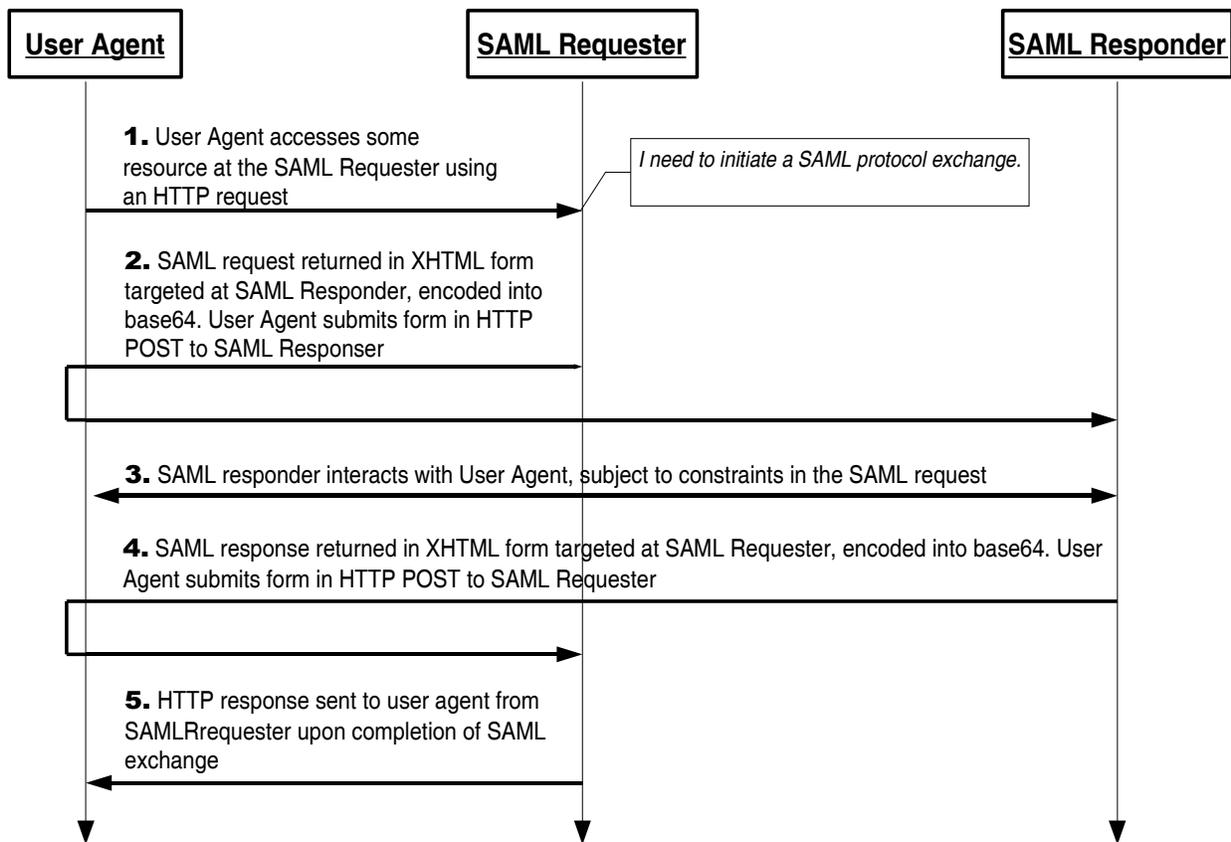
173 4. The value yielded by the signature algorithm is base64 encoded (see RFC 2045 [RFC2045]), and
174 used as the value for the `MsgSig` form control as discussed in section 2.0.4, above.

175 2.0.6 Signature Verification

176 To verify a received signed SAML message conveyed by this binding, the receiver MUST extract the form
177 control values for the `RelayState` (if present), `SigAlg`, and `SAMLRequest` (or `SAMLResponse`) values
178 (as appropriate) from the received HTTP message. Then the receiver reconstructs the string as described
179 in section 2.0.5 step 2, above. The signature value conveyed in the `MsgSig` control value is then checked
180 against this string per the signature algorithm given by the `SigAlg` control value. Error handling and
181 generated messages as a result of the signature not verifying are implementation-dependent.

182 2.0.7 Message Exchange

183 The system model used for SAML conversations via this binding is a request-response model, but these
184 messages are sent to the user agent in an HTTP response and delivered to the message recipient in an
185 HTTP request. The HTTP interactions before, between, and after these exchanges take place is
186 unspecified. Both the SAML requester and responder are assumed to be HTTP responders. See the
187 following diagram illustrating the messages exchanged.



- 188 1. Initially, the user agent makes an arbitrary HTTP request to a system entity. In the course of
 189 processing the request, the system entity decides to initiate a SAML protocol exchange.
- 190 2. The system entity acting as a SAML requester responds to an HTTP request from the user agent by
 191 returning a SAML request. The request is returned in an XHTML document containing the form and
 192 content defined in Section 2.0.4, above. The user agent delivers the SAML request by issuing an
 193 HTTP POST request to the SAML responder.
- 194 3. In general, the SAML responder MAY respond to the SAML request by immediately returning a
 195 SAML response or it MAY return arbitrary content to facilitate subsequent interaction with the user
 196 agent necessary to fulfill the request. Specific protocols and profiles may include mechanisms to
 197 indicate the requester's level of willingness to permit this kind of interaction (for example, the
 198 `IsPassive` attribute in `<samlp:AuthnRequest>`).
- 199 4. Eventually the responder SHOULD return a SAML response to the user agent to be returned to the
 200 SAML requester. The SAML response is returned in the same fashion as described for the SAML
 201 request in step 2.
- 202 5. Upon receiving the SAML response, the SAML requester returns an arbitrary HTTP response to the
 203 user agent.

204 2.0.7.1 HTTP and Caching Considerations

205 HTTP proxies and the user agent intermediary should not cache SAML protocol messages. To ensure
 206 this, the following rules SHOULD be followed.

207 When returning SAML protocol messages using HTTP 1.1, HTTP responders SHOULD:

- 208 • Include a `Cache-Control` header field set to "no-cache, no-store".

209 • Include a `Pragma` header field set to "no-cache".

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

211 2.0.7.2 Security Considerations

212 The presence of the user agent intermediary means that the requester and responder cannot rely on the
213 transport layer for endpoint-to-endpoint (i.e. SAML Requester to/from SAML Responder) authentication,
214 integrity or confidentiality protection. Instead, this binding defines a means for signing the conveyed SAML
215 messages and optional `RelayState` in order to provide endpoint-to-endpoint integrity protection and data
216 origin authentication.

217 This binding SHOULD NOT be used if the content of the request or response should not be exposed to
218 the user agent intermediary. Otherwise, confidentiality of both SAML requests and SAML responses is
219 OPTIONAL and depends on the environment of use. If confidentiality is necessary, SSL 3.0 [SSL3] or TLS
220 1.0 [RFC2246] SHOULD be used to protect the message in transit between the user agent and the SAML
221 requester and responder.

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

224 **NOTE:** cryptographically-based security is entirely OPTIONAL in this binding. If no
225 security mechanisms are employed, then there is essentially no runtime assurance as to
226 who any of the communicating entities, nor the subject (aka user), are in actuality. This
227 may be acceptable in certain deployment scenarios.

228 2.0.8 Error Reporting

229 A SAML responder that refuses to perform a message exchange with the SAML requester SHOULD
230 return a response message with a second-level `<samlp:StatusCode>` value of
231 `urn:oasis:names:tc:SAML:2.0:status:RequestDenied`.

232 HTTP interactions during the message exchange MUST NOT use HTTP error status codes to indicate
233 failures in SAML processing, since the user agent is not a full party to the SAML protocol exchange.

234 For more information about SAML status codes, see the SAML assertions and protocols specification
235 [SAMLCore].

236 2.0.9 Metadata Considerations

237 @@TODO: any fixups needed here?

238 Support for the HTTP POST binding SHOULD be reflected by indicating URL endpoints at which requests
239 and responses for a particular protocol or profile should be sent. Either a single endpoint or distinct
240 request and response endpoints MAY be supplied [SAMLMeta].

241 2.0.10 Example SAML Message Exchange Using HTTP POST - NoXMLdsig

242 In this example, a `<LogoutRequest>` and `<LogoutResponse>` message pair is exchanged using the
243 HTTP POST – NoXMLdsig binding. The messages are signed as described in section 2.0.5, above. If the
244 messages were unsigned, they would be the same as shown below, except that the hidden form controls
245 named `MsgSig` and `SigAlg` would be missing.

246 First, here are the actual SAML protocol messages being exchanged:

```
247 <samlp:LogoutRequest xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"  
248 xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
```

3 References

368

- 369 **[HTML401]** D. Raggett et al. *HTML 4.01 Specification*. World Wide Web Consortium
370 Recommendation, December 1999. See <http://www.w3.org/TR/html4>.
- 371 **[RFC2045]** N. Freed et al. *Multipurpose Internet Mail Extensions (MIME) Part One: Format of*
372 *Internet Message Bodies*, IETF RFC 2045, November 1996. See
373 <http://www.ietf.org/rfc/rfc2045.txt>.
- 374 **[RFC2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF
375 RFC 2119, March 1997. See <http://www.ietf.org/rfc/rfc2119.txt>.
- 376 **[RFC2246]** T. Dierks et al. *The TLS Protocol Version 1.0*. IETF RFC 2246, January 1999.
377 See <http://www.ietf.org/rfc/rfc2246.txt>.
- 378 **[RFC2616]** R. Fielding et al. *Hypertext Transfer Protocol – HTTP/1.1*. IETF RFC 2616, June
379 1999. See <http://www.ietf.org/rfc/rfc2616.txt>.
- 380 **[SAML20Bind]** S. Cantor et al. *Bindings for the OASIS Security Assertion Markup Language*
381 *(SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-bindings-2.0-os.
382 See <http://www.oasis-open.org/committees/security/>.
- 383 **[SAMLCore]** S. Cantor et al. *Assertions and Protocols for the OASIS Security Assertion*
384 *Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-
385 core-2.0-os. See <http://www.oasis-open.org/committees/security/>.
- 386 **[SAMLGloss]** J. Hodges et al. *Glossary for the OASIS Security Assertion Markup Language*
387 *(SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-glossary-2.0-os.
388 See <http://www.oasis-open.org/committees/security/>.
- 389 **[SAMLMeta]** S. Cantor et al. *Metadata for the OASIS Security Assertion Markup Language*
390 *(SAML) V2.0*. OASIS SSTC, March 2005. Document ID saml-metadata-2.0-os.
391 See <http://www.oasis-open.org/committees/security/>.
- 392 **[SAMLSecure]** F. Hirsch et al. *Security and Privacy Considerations for the OASIS Security*
393 *Assertion Markup Language (SAML) V2.0*. OASIS SSTC, March 2005. Document
394 ID saml-sec-consider-2.0-os. See [http://www.oasis-](http://www.oasis-open.org/committees/security/)
395 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 396 **[SSL3]** A. Frier et al. *The SSL 3.0 Protocol*. Netscape Communications Corp, November
397 1996.
- 398 **[SSTCWeb]** OASIS Security Services Technical Committee website, [http://www.oasis-](http://www.oasis-open.org/committees/security/)
399 [open.org/committees/security/](http://www.oasis-open.org/committees/security/).
- 400 **[XHTML]** *XHTML 1.0 The Extensible HyperText Markup Language (Second Edition)*. World
401 Wide Web Consortium Recommendation, August 2002. See
402 <http://www.w3.org/TR/xhtml1/>.
- 403 **[XMLSig]** D. Eastlake et al. *XML-Signature Syntax and Processing*. World Wide Web
404 Consortium Recommendation, February 2002. See
405 <http://www.w3.org/TR/xmlsig-core/>.

406 **Appendix B. Acknowledgments**

407 @@TODO: update as appropriate

408 The editors acknowledge the contributions of the OASIS Security Services Technical Committee, whose
409 voting members at the time of publication were:

- 410 • TBD

411 The editors also acknowledge the following former SSTC members for their contributions to this or
412 previous versions of the OASIS Security Assertions Markup Language Standard:

- 413 • TBD

414 **Appendix C. Notices**

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

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

426 **Copyright © OASIS Open 2005. All Rights Reserved.**

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

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

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