



1

2 Web-services policy language use- 3 cases and requirements

4 Working draft 04, 16 April 2003

5 Document identifier: wd-xacml-wspl-use-cases-04.pdf

6 Location: http://www.oasis-open.org/committees/documents.php?wg_abbrev=xacml

7 Send comments to: xacml-comment@lists.oasis-open.org

8 Editors:

9 Tim Moses, Entrust (tim.moses@entrust.com)

10 Contributors:

11 Anne Anderson, Sun Microsystems

12 Frank Siebenlist, Argonne National Labs

13 Frederick Hirsch, Nokia Mobile Phone

14 Ron Monzillo, Sun Microsystems

15 Simon Godik, Overxeer

16 Abstract:

17 This working draft defines use-cases and requirements for negotiating a variety of forms of
18 policy in the Web-services architecture.

19 Status:

20 This version of the specification is a working draft of the committee. As such, it is expected
21 to change prior to adoption as an OASIS standard.

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

26

27 Copyright (C) OASIS Open 2003 All Rights Reserved.

28 Table of contents

29	1.	Introduction	4
30	2.	Use-cases	4
31	2.1.	Use-case 1: Submit request	4
32	2.2.	Use-case 2: Return response	5
33	2.3.	Use-case 3: Construct request	6
34	2.4.	Use-case 4: Construct response	8
35	2.5.	Use-case 5: Control usage	9
36	2.6.	Use-case 6: Intermediary proxies	10
37	2.7.	Use-case 7: Intermediary intercepts	13
38	2.8.	Use-case 8: Multiple sources	15
39	2.9.	Use-case 9: Second party combines	16
40	2.10.	Use-case 10: Third party combines	17
41	2.11.	Use-case 11: Third-party translates	18
42	3.	Policy communication	19
43	4.	Language support	19
44	5.	Requirements	19
45	5.1.	Three-value logic	19
46	5.2.	Amenable to combining	20
47	5.3.	Interpretation as instructions	20
48	5.4.	Common data-types	20
49	5.5.	Extensible data-types	20
50	5.6.	Common operators	20
51	5.7.	Extensible operators	20
52	5.8.	Multiple enforcement points	20
53	5.9.	Multiple bindings	20
54	5.10.	Preferences	21
55	5.11.	Suppressed disclosure	21
56	5.12.	Supported functions	21
57	5.13.	Specified order	21
58	5.14.	Policy identified by name	21
59	5.15.	Attributes identified by name	21
60	5.16.	Attributes identified by location	21
61	5.17.	Behaviour in event attributes are unavailable	21
62	5.18.	Version control	21
63	6.	References	22

64 Appendix A. Notices
65
66

23

67 1. Introduction

68 This document explores the requirements for policy expression in the Web-services application
69 domain.

70 Several applications of policy were considered in preparing this analysis, including: cryptographic-
71 security policy, authentication policy, authorization policy, privacy policy, reliable-messaging policy
72 transaction-processing policy and trust policy.

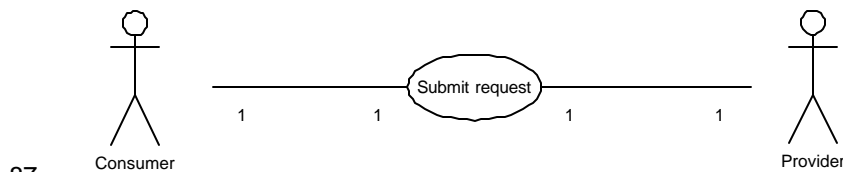
73 2. Use-cases

74 2.1. Use-case 1: Submit request

75 Use-case 1 is shown in Figure 1. In this case, Consumer submits a service request to Provider. If
76 the service request conforms with Provider's policy for requests, then Provider accepts the request.
77 Otherwise, it returns a fault status. Optionally, in the fault case, it returns its policy for requests of
78 the type.

79 Consumer may not wish to disclose information in a genuine service request until it can be certain
80 that its request will be acceptable to Provider, by virtue of the fact that it conforms with Provider's
81 policy.

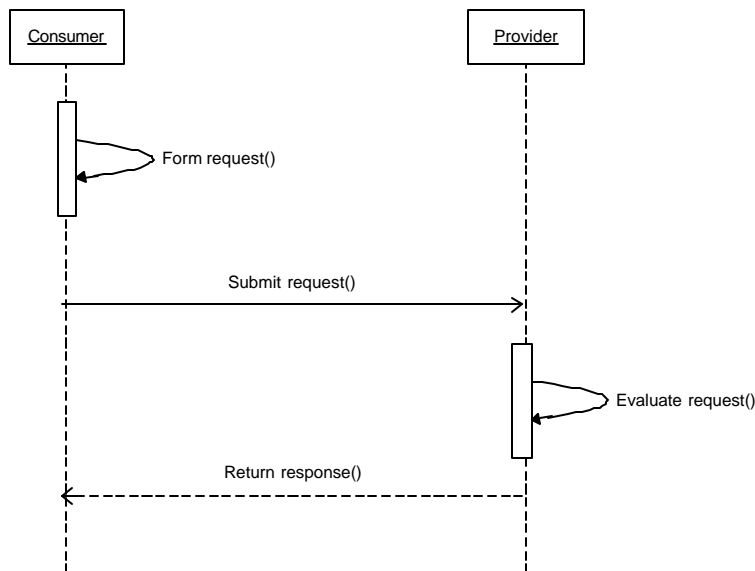
82 This use-case applies to situations in which Provider imposes requirements on the form of
83 acceptable service requests and/or is willing to accept service requests of a certain form. This
84 situation exists, for instance, where Provider requires Consumer to assign a unique identifier to its
85 request, in accordance with WS-Reliability [WS-Rel]. If it receives a request with no suitable
86 identifier, then it will return a fault status.



87

88 **Figure 1 - Use-case 1**

89 The corresponding sequence diagram is shown in Figure 2.



90

91 **Figure 2 - Use-case 1 sequence**

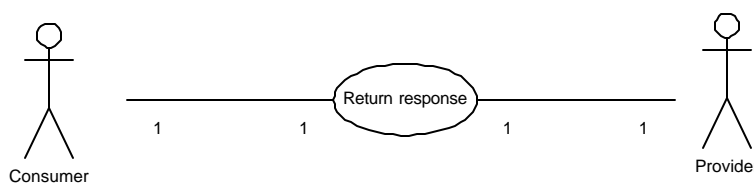
- 92 1. Consumer forms a service request in compliance with its own policy for the request type.
- 93 2. Consumer sends the request to Provider.
- 94 3. Provider tests the request against its policy for the request type.
- 95 4. If the request satisfies Provider's policy, then Provider accepts the request and (optionally)
- 96 returns a response. If the request does not satisfy Provider's policy, then Provider returns a
- 97 fault status and, optionally, its policy for requests of the type.

98 Note: Consumer may send an empty service request so that it can obtain Provider's policy without
99 disclosing information.

100 2.2. Use-case 2: Return response

101 Use-case 2 is shown in Figure 3. In this case, Provider returns a service response to Consumer. If
102 the service response conforms with Consumer's policy for responses, then it accepts the response.
103 Otherwise, it discards the response.

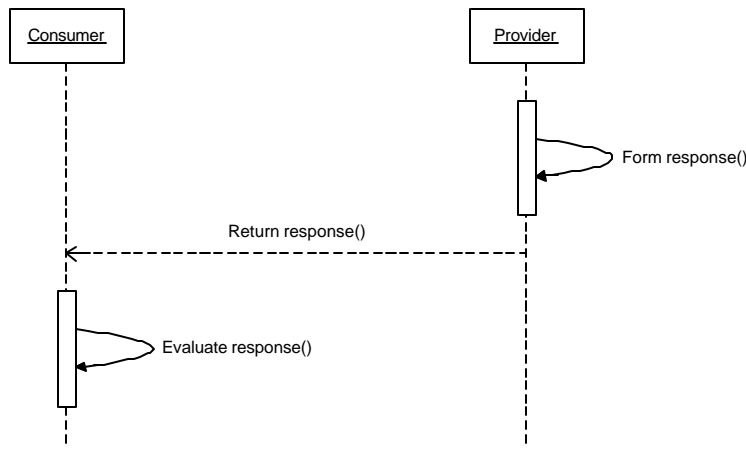
104 This use-case applies to situations in which Consumer imposes requirements on the form of
105 acceptable service responses and/or is willing to accept service responses of a certain form. This
106 situation exists, for instance, where Consumer requires Provider to certify certain contents of the
107 response by signing them.



108

109 **Figure 3 - Use-case 2**

110 The corresponding sequence diagram is shown in Figure 4.



111

112 **Figure 4 - Use-case 2 sequence**

- 113 1. Provider forms a service response in compliance with its own policy for the response type.
- 114 2. Provider returns the response.
- 115 3. Consumer tests the response against its policy for responses of the type. If the response
- 116 satisfies its policy, then it accepts the response. Otherwise, Consumer discards the response.

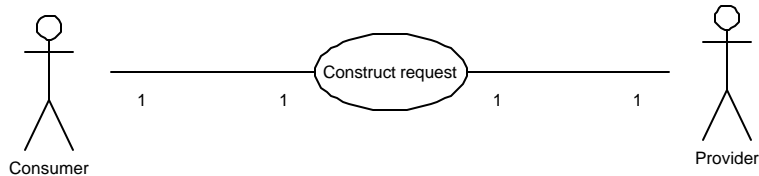
117 **2.3. Use-case 3: Construct request**

118 Use-case 3 is shown in Figure 5. In this case, Consumer forms a request that it knows will be
119 accepted by Provider because it conforms with Provider's policy for requests of the type.

120 This use-case applies to situations in which Consumer cannot form an acceptable service request
121 by repeatedly submitting and modifying requests until one is accepted. Rather it must form a
122 service request that it can be certain is acceptable to Provider. Therefore, Provider describes in its
123 policy the functions that it insists on performing and the functions that it is willing and able to
124 perform. This description may include acceptable alternative functions. There may be differential
125 costs associated with the alternative functions. Therefore, Provider may wish to indicate which of
126 the alternative functions it prefers to perform. Likewise, Consumer may have preferences amongst
127 the alternative functions. Consumer's preferences may not necessarily align with Provider's
128 preferences.

129 Consumer may construct the request directly, by examining Provider's policy, or by testing
130 candidate requests against Provider's policy.

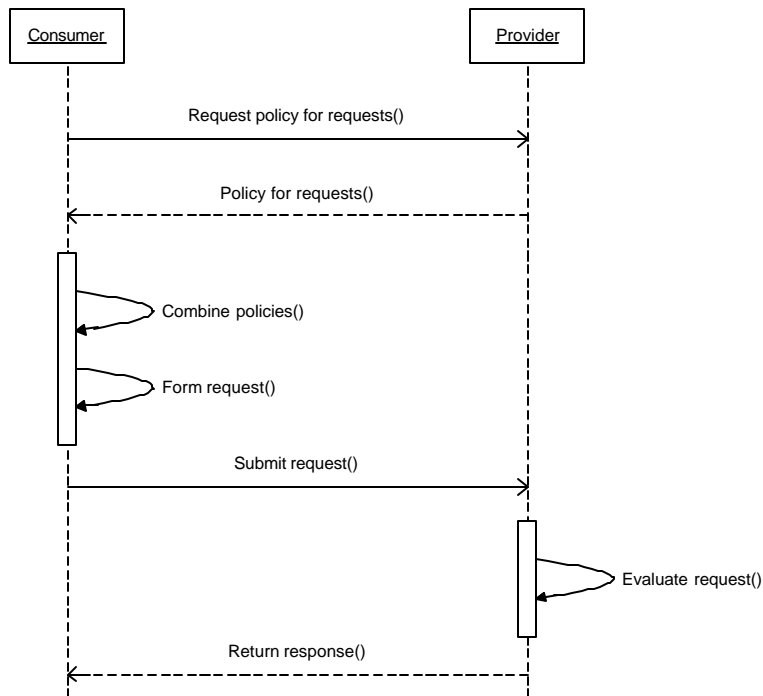
131 This situation exists, for instance, where Provider imposes an upper limit on the "time to live" of a
132 WS-Reliability [WS-Rel] message. In the event that Consumer chooses a value that exceeds this
133 upper limit, its request will be rejected.



134

135 **Figure 5 - Use-case 3**

136 The corresponding sequence diagram is shown in Figure 6.



137

138 **Figure 6 - Use-case 3 sequence**

- 139 1. Consumer requests Provider's policy for requests.
- 140 2. Consumer obtains Provider's policy for requests.
- 141 3. Consumer combines Provider's policy for requests with its own.
- 142 4. Consumer forms the request in conformance with the combined policy for requests.
- 143 5. Consumer sends the request for service to Provider.
- 144 6. Provider verifies that the request satisfies its policy for requests.
- 145 7. If it does, then it accepts the request and (optionally) returns a response. Otherwise, it returns
- 146 a fault status.
- 147 Note: Steps 3 and 4 may be accomplished by trial and error.

148

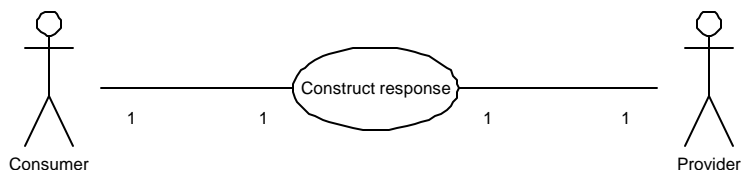
2.4. Use-case 4: Construct response

149 Use-case 4 is shown in Figure 7. In this case, Provider forms a response that it knows will be
150 accepted by Consumer, because it conforms with Consumer's policy for responses.

151 This use-case applies to situations in which Provider cannot form an acceptable response by
152 repeatedly returning and modifying responses until one is accepted. Rather it must form a service
153 response that it can be certain is acceptable to Consumer. Therefore, Consumer describes in its
154 policy the functions that it insists on performing and the functions that it is willing and able to
155 perform. As in use-case 4, the description may include acceptable alternative functions. There
156 may be differential costs associated with the alternative functions. Therefore, Consumer may wish
157 to indicate which of the alternative functions it prefers to perform. Likewise, Provider may have
158 preferences amongst the alternative functions. Provider's preferences may not necessarily align
159 with Consumer's preferences.

160 Provider may construct the response directly, by examining Consumer's policy, or by testing
161 candidate responses against Consumer's policy.

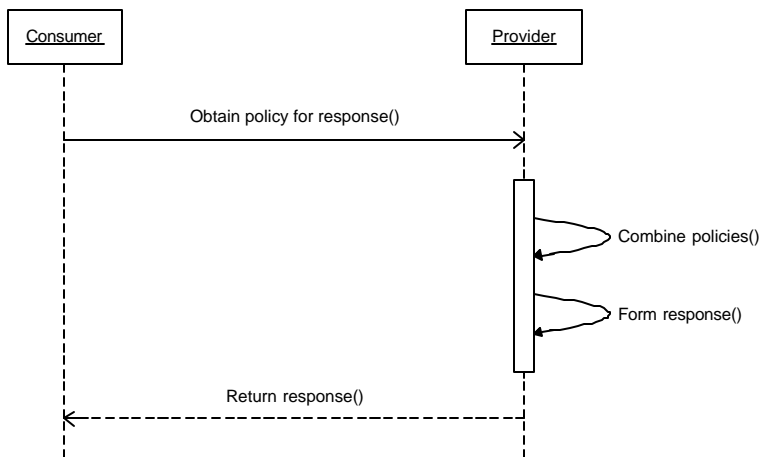
162 This situation exists, for instance, where Provider's policy requires that certain contents be
163 encrypted, while Consumer's policy requires that certain other contents be "in the clear". Provider
164 is able to form a response in which information that is required to be encrypted is encrypted, and
165 information that is required to be "in the clear" is "in the clear".



166

167 **Figure 7 - Use-case 4**

168 The corresponding sequence diagram is shown in Figure 8.



169

170 **Figure 8 - Use-case 4 sequence**

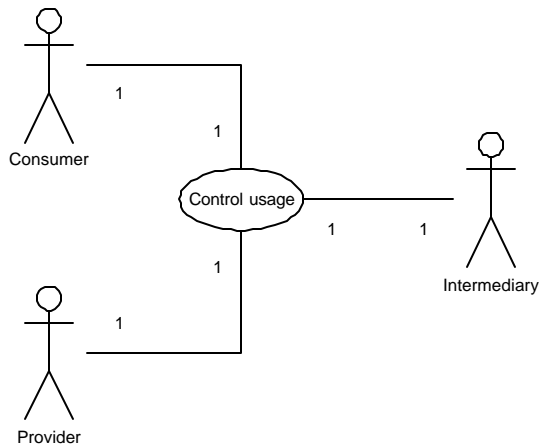
171 1. Provider obtains Consumer's policy for responses.

- 172 2. Provider combines Consumer's policy for responses with its own.
173 3. Provider forms a response in conformance with the combined policy for responses.
174 4. Provider returns the response to Consumer.
175 Note: Steps 2 and 3 may be accomplished by trial and error.

2.5. Use-case 5: Control usage

177 Use-case 5 is shown in Figure 9. In this case, Consumer's policy places limits on Intermediary's
178 use of Consumer's request. Intermediary forwards Consumer's modified request to Provider, only
179 in conformance with its own and with Consumer's usage policy. Intermediary may also forward
180 Consumer's usage policy to Provider.

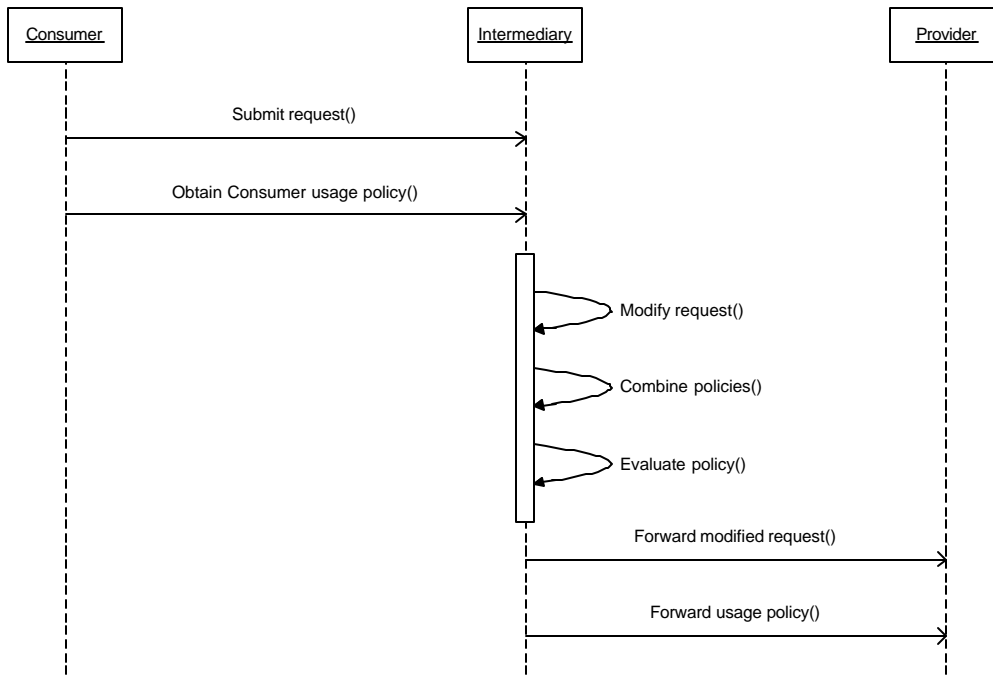
181 This use-case applies, for instance, when Consumer provides confidential information, including
182 (but not limited to) personal information, and Intermediary has to pass certain parts of the
183 confidential information to Provider, an actor not governed by Intermediary.



184

185 **Figure 9 - Use-case 5**

186 The corresponding sequence diagram is shown in Figure 10.



187

188 **Figure 10 - Use-case 5 sequence**

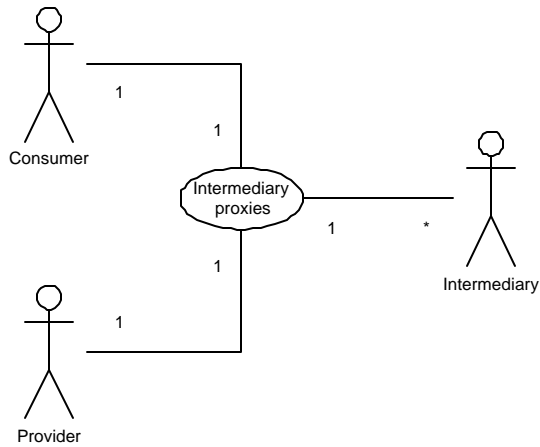
- 189 1. Consumer submits request to Intermediary.
190 2. Intermediary obtains Consumer's usage policy.
191 3. Intermediary processes Consumer's request.
192 4. Intermediary combines Consumer's usage policy with its own.
193 5. Intermediary evaluates its own and Consumer's usage policy.
194 6. If the combined policy is satisfied, then Intermediary sends the modified request to Provider.
195 Otherwise, it does not.
196 7. Optionally, Provider obtains the usage policy for the modified request.

197 **2.6. Use-case 6: Intermediary proxies**

198 Use-case 6 is shown in Figure 11. In this case, Intermediary acts as a proxy for Provider.
199 Intermediary combines Provider's policy for requests with its own to express the effective policy for
200 Consumer's request. There may be a chain of intermediaries in the path between Consumer and
201 Provider; each outputs its own policy as a modified version of the policy obtained from the next
202 "upstream" actor. Consumer sends a service request to Intermediary. Intermediary forwards a
203 modified request to Provider.

204 In this use case, an intermediary serves as a proxy for a single service provider.

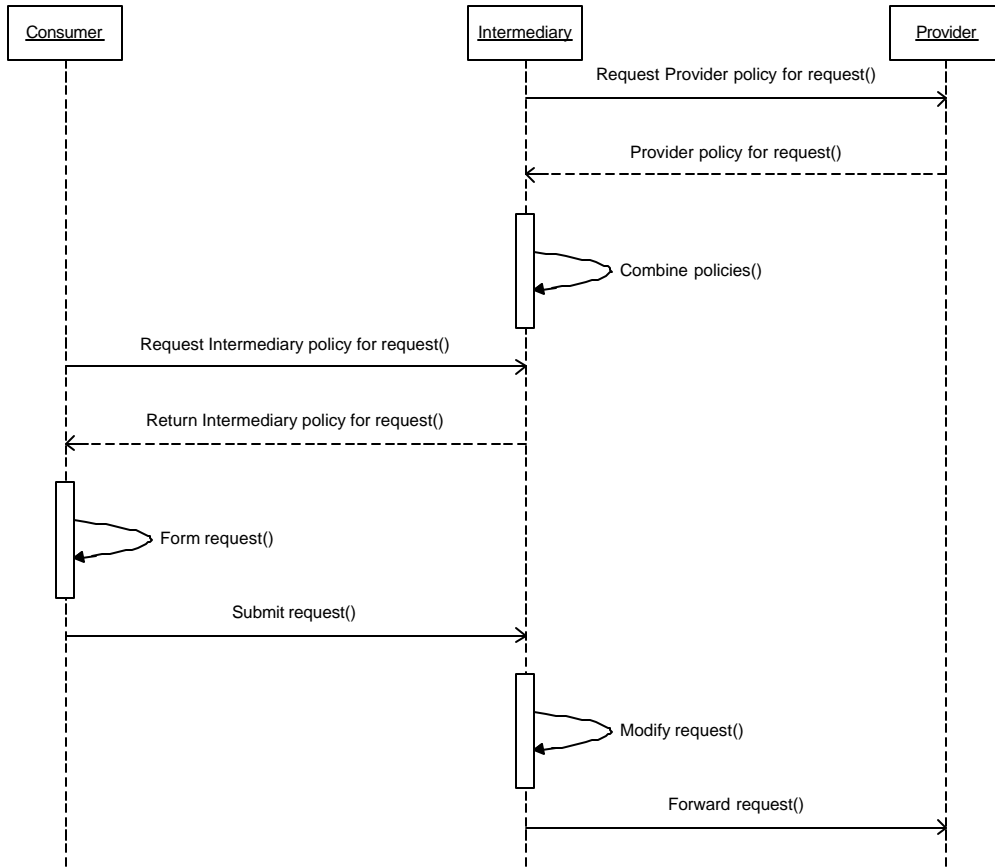
205 This use-case applies when Provider imposes policy requirements that affect the request submitted
206 by Consumer, although Consumer is unaware of the role played by Provider in the request. A
207 firewall that performs address translation may act in this way: taking a Provider's policy and
208 modifying it to include its own requirements.



209

210 **Figure 11 - Use-case 6**

211 The corresponding sequence diagram is shown in Figure 12.



212

213 **Figure 12 - Use-case 6 sequence**

214 1. Intermediary requests policy for requests from Provider.

215 2. Provider returns policy for requests to Intermediary.

216 3. Intermediary combines Provider's policy with its own.

217 4. Consumer requests policy from Intermediary.

218 5. Intermediary returns policy to Consumer.

219 6. Consumer forms a request in conformance with policy.

220 7. Consumer submits a conformant request to Intermediary.

221 8. Intermediary modifies the request.

222 9. Intermediary forwards the request to Provider.

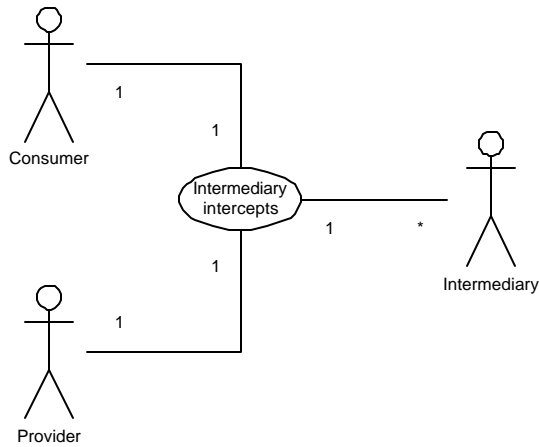
223 Note: Consumer does not have to be aware that the policy provided by Intermediary is the result of
224 combining Intermediary's policy with that of Provider.

225 There is a corresponding use-case for responses, in which Consumer sends its policy for
226 responses to Intermediary, Intermediary combines it with its own and passes the result to Provider.
227 Provider then forms the response in conformance with the combined policy.

228 2.7. Use-case 7: Intermediary intercepts

229 Use-case 7 is shown in Figure 13. In this case, an Intermediary places itself in the path between
230 Consumer and Provider, without the knowledge of either actor. There may be a chain of
231 intermediaries in the path between Consumer and Provider; each outputs its own policy as a
232 modified version of the policy obtained from the next “upstream” actor. Intermediary imposes policy
233 requirements on requests and responses exchanged between Consumer and Provider.

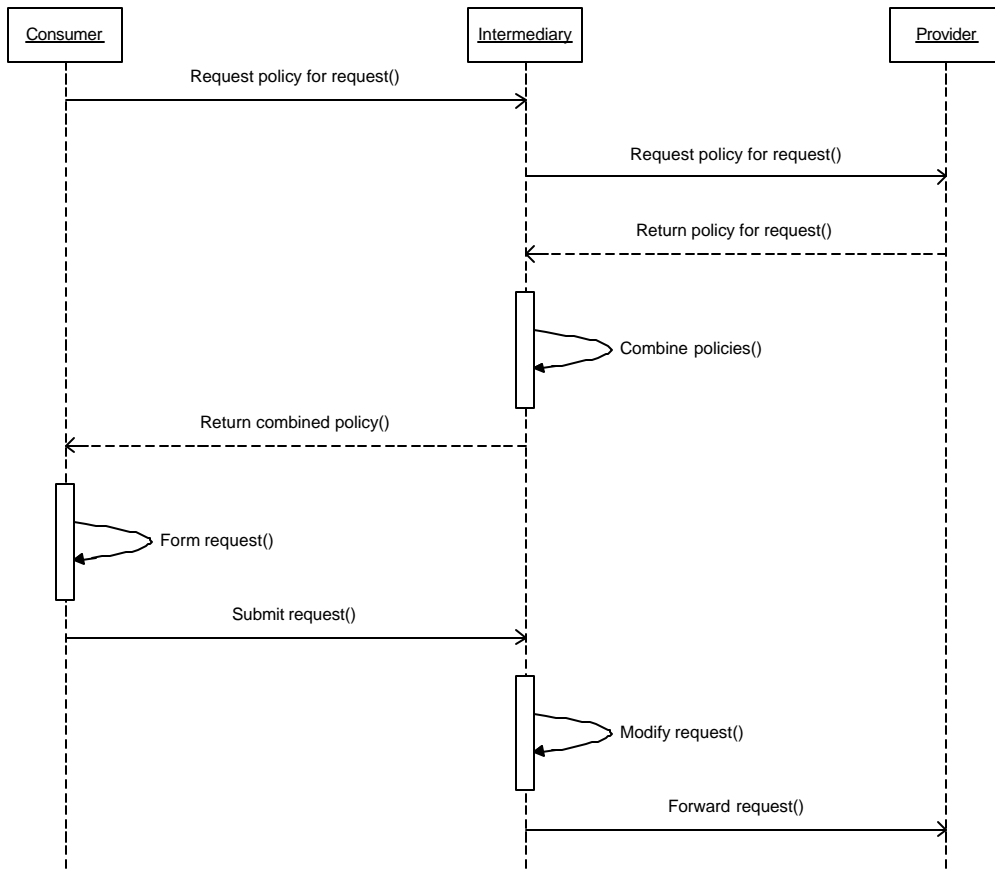
234 This use-case applies for instance when security functions are performed by an interceptor.



235

236 **Figure 13 - Use-case 7**

237 The corresponding sequence diagram is shown in Figure 14.



238

239 **Figure 14 - Use-case 7 sequence**

- 240 1. Consumer requests policy for requests from Provider. The request is intercepted by
- 241 Intermediary.
- 242 2. Intermediary requests policy for requests from Provider.
- 243 3. Provider returns policy for requests to Intermediary.
- 244 4. Intermediary combines Provider's policy with its own.
- 245 5. Intermediary returns combined policy to Consumer.
- 246 6. Consumer forms a request in conformance with policy.
- 247 7. Consumer submits the request to Provider. The request is intercepted by Intermediary.
- 248 8. Intermediary modifies the request.
- 249 9. Intermediary forwards the modified request to Provider.

250 There is a corresponding use-case for responses, in which Consumer sends its policy for
251 responses to Intermediary, Intermediary combines it with its own and passes the result to Provider.
252 Provider then forms the response in conformance with the combined policy.

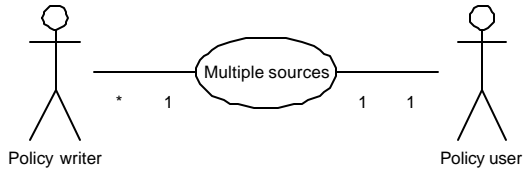
253

2.8. Use-case 8: Multiple sources

254 Use-case 8 is shown in Figure 15. In this case, the complete policy associated with a particular
255 operation (whether request or response) is formed by combining policies from a number of sources.

256 This use-case applies, for instance, when the policy applicable to a request is defined at both the
257 departmental and corporate levels of an enterprise. Either the policies may be combined or the
258 evaluation results may be combined. Combination may be performed by the policy user or by
259 another actor.

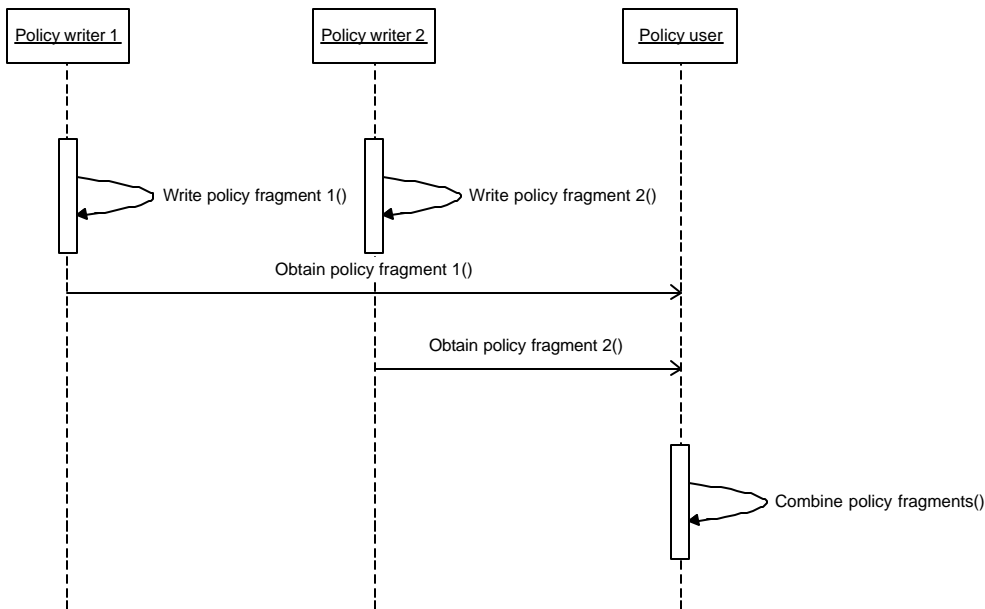
260 Policy fragments may be referenced by name for the purpose of location and retrieval.



261

262 **Figure 15 - Use-case 8**

263 The corresponding sequence diagram is shown in Figure 16.



264

265 **Figure 16 - Use-case 8 sequence**

- 266 1. Policy writer 1 prepares policy fragment 1.
- 267 2. Policy writer 2 prepares policy fragment 2.
- 268 3. Policy user obtains policy fragment 1.
- 269 4. Policy user obtains policy fragment 2.
- 270 5. Policy user combines policy fragment 1 and policy fragment 2.

271

2.9. Use-case 9: Second party combines

272

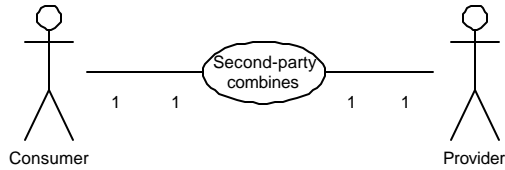
Use-case 9 is shown in Figure 17. In this case, the combined policy associated with a service request is formed by Provider and then returned to Consumer.

273

274

This use-case applies when Provider is unwilling to reveal its policy, for instance, if it wishes to ensure that Consumer uses Provider's preferred options, rather than its own preferred options.

275



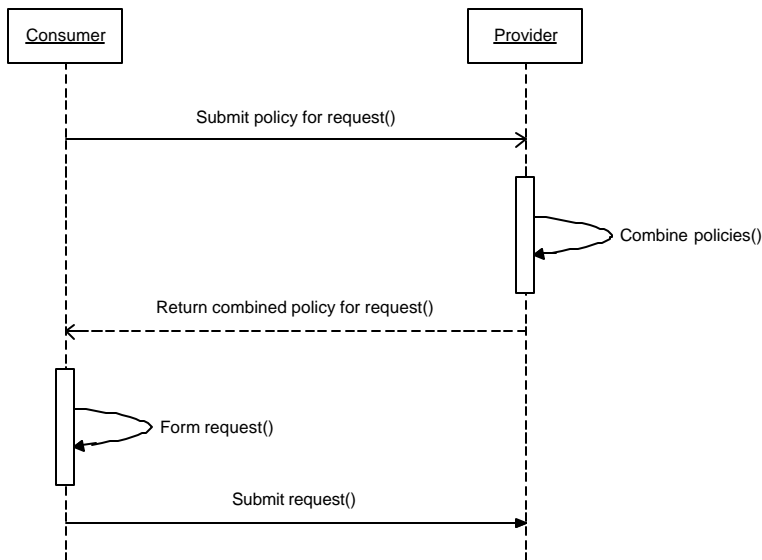
276

277

Figure 17 - Use-case 9

278

The corresponding sequence diagram is shown in Figure 18.



279

280

Figure 18 - Use-case 9 sequence

281

1. Consumer sends policy for request to Provider.

282

2. Provider combines Consumer's policy for request with its own.

283

3. Provider returns the combined policy to Consumer.

284

4. Consumer forms a request in conformance with the combined policy.

285

5. Consumer submits a request that conforms with the combined policy.

286

There is a corresponding use-case for responses, in which Provider sends its policy for responses

287

to Consumer, Consumer combines it with its own and returns the result to Provider. Provider then

288

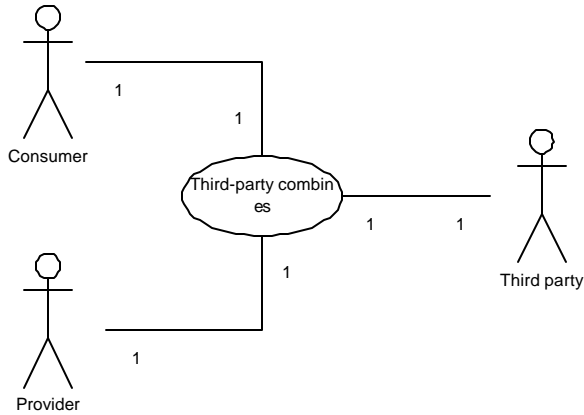
forms the response in conformance with the combined policy.

289

2.10. Use-case 10: Third party combines

290 Use-case 10 is shown in Figure 19. In this case, the combined policy associated with a service
291 request is formed by a third party and then returned to Consumer.

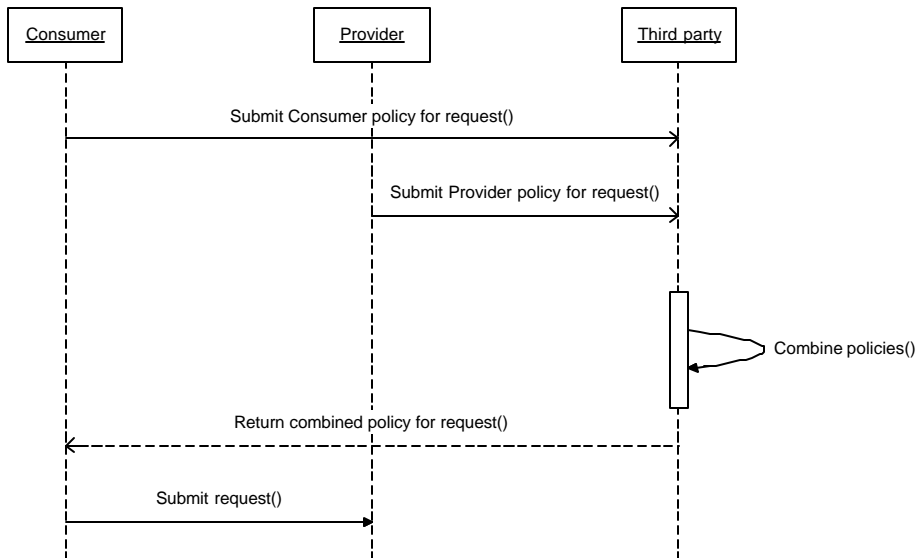
292 This situation exists when neither Consumer nor Provider wishes to reveal its policy to the other.



293

294 **Figure 19 - Use-case 10**

295 The corresponding sequence diagram is shown in Figure 20.



296

297 **Figure 20 - Use-case 10 sequence**

298 1. Consumer sends policy for request to Third party.

299 2. Provider sends policy for request to Third party.

300 3. Third party combines Consumer's policy for request with Provider's policy for request.

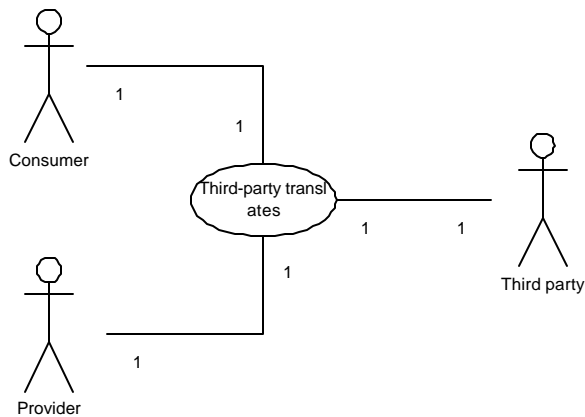
301 4. Third party returns the combined policy to Consumer.

302 5. Consumer submits a request that conforms with the combined policy.
 303 There is a corresponding use-case for responses, in which Third party returns the combined policy
 304 to Provider in step 4, and, in step 5, Provider returns the response to Consumer.

305 2.11. Use-case 11: Third-party translates

306 Use-case 11 is shown in Figure 21. In this case, the Provider policy associated with a service
 307 request is translated into a form that is acceptable to Consumer by a third party.

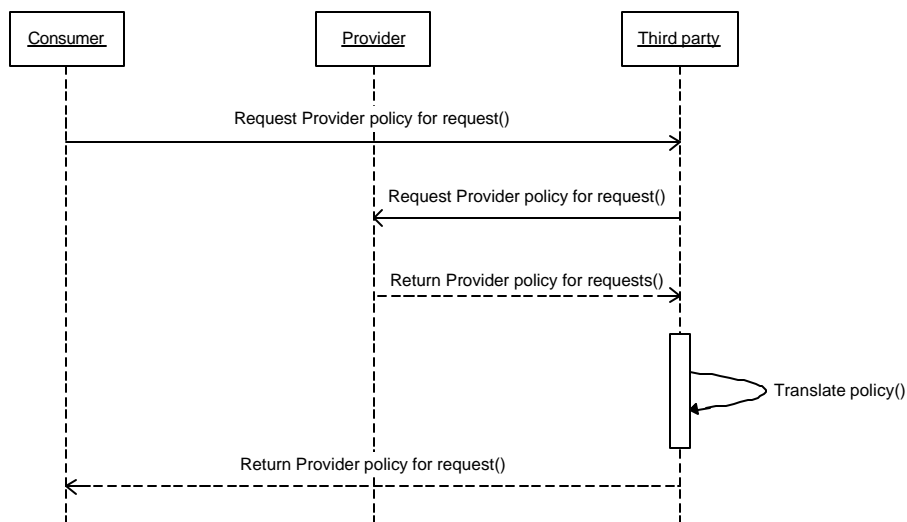
308 This situation exists when there is no single policy syntax understood by both Consumer and
 309 Provider.



310

311 **Figure 21 - Use-case 11**

312 The corresponding sequence diagram is shown in Figure 22.



313

314 **Figure 22 - Use-case 11 sequence**

315 1. Consumer requests Provider's policy for requests from Third party.

- 316 2. Third party requests the policy for requests from Provider.
- 317 3. Provider returns its policy for requests to Third party in its chosen syntax.
- 318 4. Third party translates the policy to the syntax chosen by Consumer.
- 319 5. Third party returns the policy to Consumer.
- 320 There is a corresponding use-case for responses, in which Third party translates Consumer policy
- 321 in step 4 and returns it to Provider in step 5.

322 3. Policy communication

323 In all use-cases, policy instances may be communicated in any one of a number of ways. For

324 instance:

325 In the case of simple service provision, where Consumer sends an isolated service request to

326 Provider, Provider may publish its policy in one or more of a number of ways, including: UDDI,

327 WSDL, HTTP, LDAP, DNS or in SQL or SAML request/response messages.

328 In the case of complex service provision, the Provider and Consumer may communicate their

329 policies to one another in-band, for instance, by including them as SOAP headers.

330 4. Language support

331 The policy language has to support alternative combinations of requirements, which gives rise to

332 the need for logical combining operations, such as OR and AND. Support for reliable-messaging

333 requirements gives rise to the need for integer comparison operations, such as greater-than and

334 less-than, and support for cryptographic-security requirements gives rise to the need for set

335 operations, such as subset and superset, over XML nodes and resource identifiers.

336 It must also be possible to indicate operations that must not be performed.

337 In some application domains, policies may be expressed as a set of independent **objectives**, each

338 of which may be achieved by any one of a number of alternative **strategies**. Each strategy

339 comprises a number of mandatory **predicates**. There should be a suitable way of expressing

340 policies of this form.

341 5. Requirements

342 5.1. Three-value logic

343 In order to support use-cases 1,2 and 6, it must be possible to evaluate an instance of policy to

344 produce a Boolean result. A “True” result indicates that the requested action conforms with policy.

345 A “False” result indicates that it does not. In the case that necessary information is unavailable, an

346 “Indeterminate” result should be returned.

347 **5.2. Amenable to combining**

348 In order to support use-case 6, it must be possible to combine the results of evaluation of two or
349 more policies. In order to support use-cases 4, 5, 7, 8, 9, 10 and 11, it must be possible to combine
350 and reduce two or more policies to derive a set of instructions (see Section 5.3).

351 Note: an acceptable approach is to evaluate the candidate service messages, in turn, against each
352 of the policies, until one is found to conform.

353 **5.3. Interpretation as instructions**

354 In order to support use-cases 4 and 5, it must be possible to derive from a policy instance a set of
355 instructions for producing a request that conforms with the policy.

356 **5.4. Common data-types**

357 In order to support multiple policy types in an efficient and interoperable manner, a common set of
358 data-types must be defined. This must include integers, XML nodes and resource identifiers.

359 **5.5. Extensible data-types**

360 In order to address unforeseen applications, it must be possible to extend the set of built-in data-
361 types.

362 **5.6. Common operators**

363 In order to support multiple policy types in an efficient and interoperable manner, a common set of
364 operators must be defined. These must include logical operators (including NOT), integer
365 comparison operators and set operators.

366 **5.7. Extensible operators**

367 In order to address unforeseen applications, it must be possible to extend the set of built-in
368 operators.

369 **5.8. Multiple enforcement points**

370 In order to support multiple policy types, each with a distinct enforcement point, it must be possible
371 to target a policy instance at a specific enforcement point and message type, and for that
372 enforcement point to be able to identify and extract the piece of a policy instance that is appropriate
373 to it. Enforcement points must, at least, include: cryptographic-security, authentication,
374 authorization, privacy, reliable-messaging, transaction-processing and trust. Likewise, actors
375 responsible for particular aspects of message preparation must be able to identify and extract the
376 components of policy that are applicable to that aspect.

377 **5.9. Multiple bindings**

378 It must be possible to convey policy instances in a number of different protocols, including: UDDI,
379 WSDL, SOAP, LDAP, DNS, HTTP and in SQL and SAML attribute request/response messages.

380 **5.10. Preferences**

381 It must be possible for a Web-services end-point to indicate its order of preference amongst a
382 mutually-acceptable set of optional functions.

383 Note: consideration should be given to the practicality of identifying the preferred option when the
384 parties' preferences fail to align.

385 **5.11. Suppressed disclosure**

386 End-points must be able to defer disclosure of message payload data until such time as they know
387 that their request will be accepted by the destination end-point.

388 **5.12. Supported functions**

389 It must be possible for a Web-services end-point to indicate operations that it is capable of
390 performing, as well as operations that it insists upon performing.

391 **5.13. Specified order**

392 It must be possible for a Web-services end-point to indicate the order in which it will perform
393 operations, and thereby, the order in which operations must be performed on a message intended
394 to conform with that end-point's policy.

395 **5.14. Policy identified by name**

396 It must be possible to reference a policy instance by an identifier of various types.

397 **5.15. Attributes identified by name**

398 It must be possible to reference attributes in a policy instance by an identifier of various types.

399 **5.16. Attributes identified by location**

400 It must be possible to reference attributes in a policy instance by their location within a message.

401 **5.17. Behaviour in event attributes are unavailable**

402 It must be possible to specify in a policy instance behaviour in the event that referenced attributes
403 cannot be evaluated.

404 **5.18. Version control**

405 From time to time, policy instances may have to be withdrawn and replaced. Mechanisms are
406 required to identify the version of a policy that is currently in effect.

407

6. References

408

WS-Rel: Web Services Reliability (WS-Reliability) Ver1.0, January 8, 2003. [http://www.oasis-](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrm)

409

[open.org/committees/tc_home.php?wg_abbrev=wsrm](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrm)

410 Appendix A. Notices

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

420 OASIS has been notified of intellectual property rights claimed in regard to some or all of the
421 contents of this specification. For more information consult the online list of claimed rights.

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

425 Copyright (C) OASIS Open 2003. All Rights Reserved.

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

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

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