Web-services policy language use-cases and requirements

Working draft 04, 16 April 2003

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

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

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

Editors:

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

Contributors:

Anne Anderson, Sun Microsystems
Frank Siebenlist, Argonne National Labs
Frederick Hirsch, Nokia Mobile Phone
Ron Monzillo, Sun Microsystems
Simon Godik, Overxeer

Abstract:

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

Status:

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

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

Table of contents

65

66

1. Introduction

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

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

2. Use-cases

2.1. Use-case 1: Submit request

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

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

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

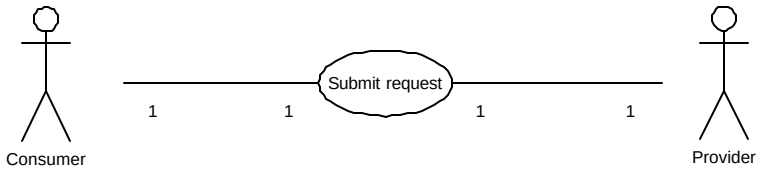


Figure 1 - Use-case 1

The corresponding sequence diagram is shown in Figure 2.

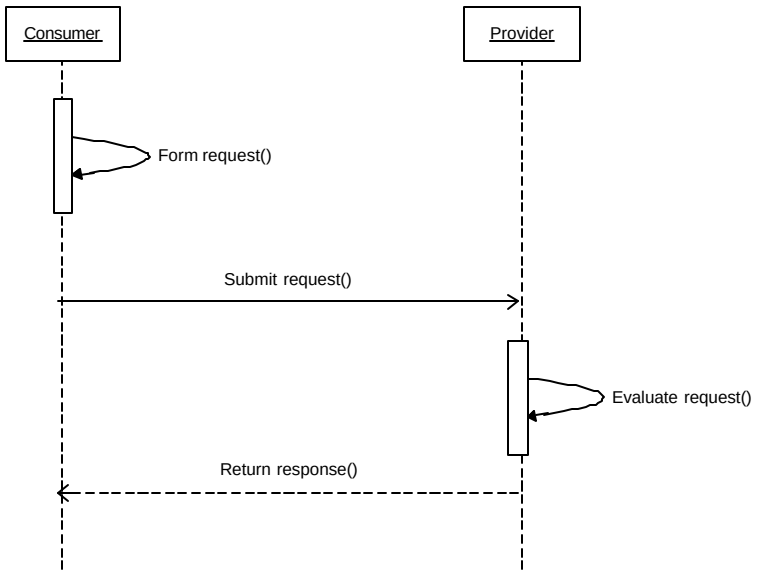


Figure 2 - Use-case 1 sequence

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

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

2.2. Use-case 2: Return response

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

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

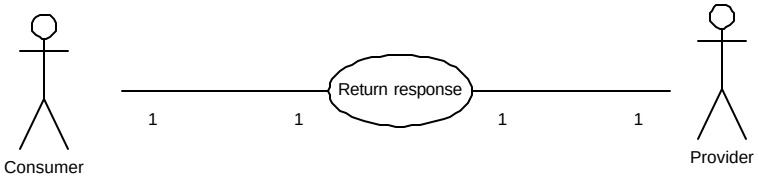


Figure 3 - Use-case 2

The corresponding sequence diagram is shown in Figure 4.

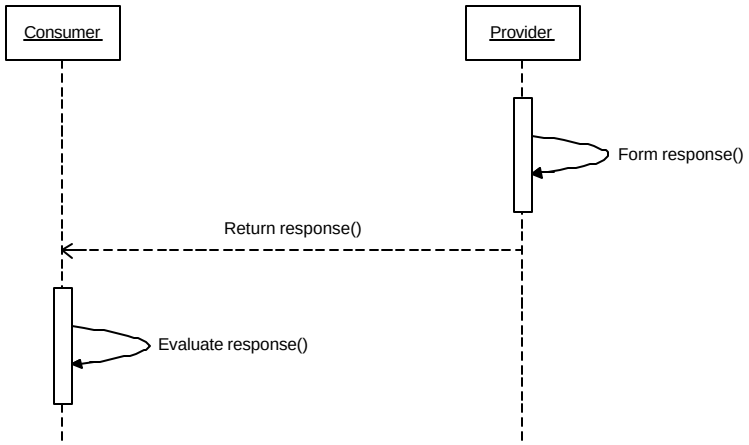


Figure 4 - Use-case 2 sequence

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

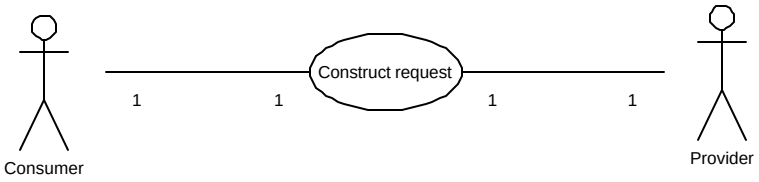
2.3. Use-case 3: Construct request

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

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

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

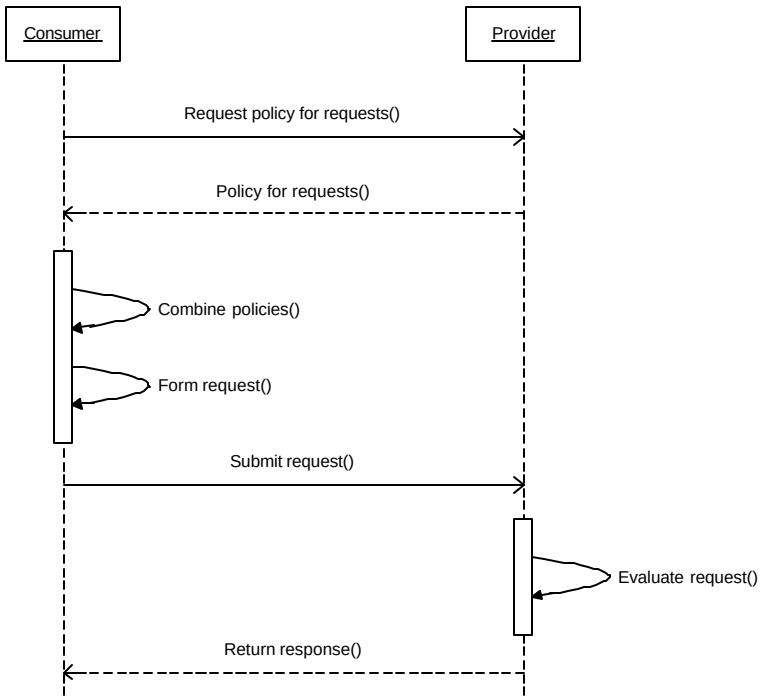
This situation exists, for instance, where Provider imposes an upper limit on the "time to live" of a WS-Reliability **[WS-Rel]** message. In the event that Consumer chooses a value that exceeds this upper limit, its request will be rejected.



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.

2.4. Use-case 4: Construct response

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

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

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

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

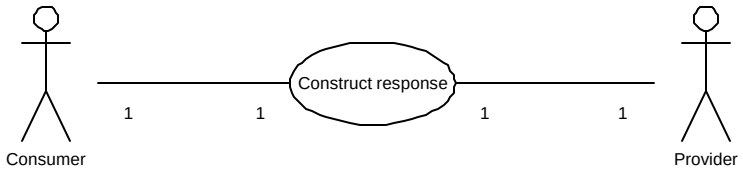


Figure 7 - Use-case 4

The corresponding sequence diagram is shown in Figure 8.

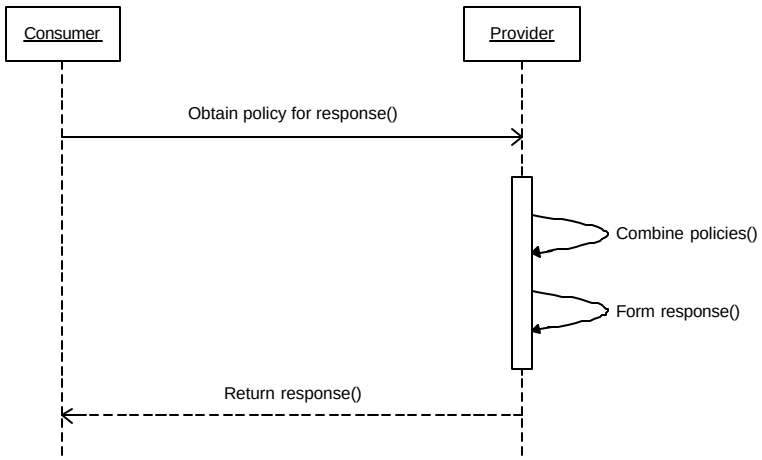


Figure 8 - Use-case 4 sequence

1. Provider obtains Consumer's policy for responses.

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

3. Provider forms a response in conformance with the combined policy for responses.

4. Provider returns the response to Consumer.

Note: Steps 2 and 3 may be accomplished by trial and error.

2.5. Use-case 5: Control usage

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

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

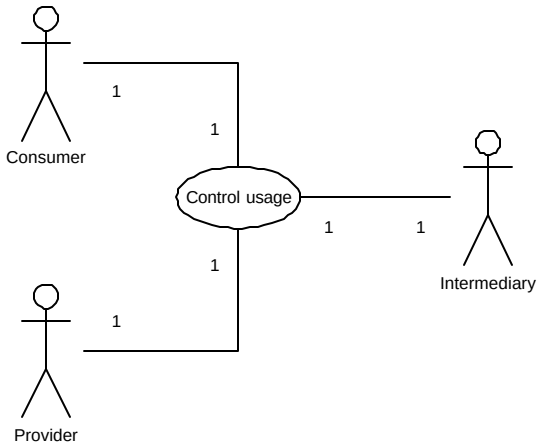
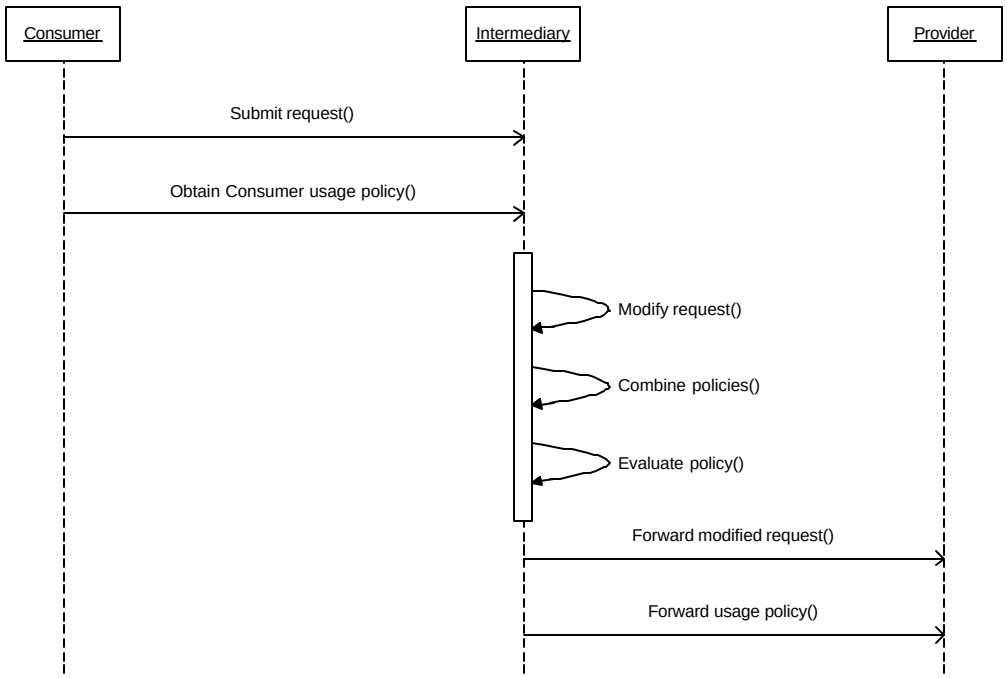


Figure 9 - Use-case 5

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.

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

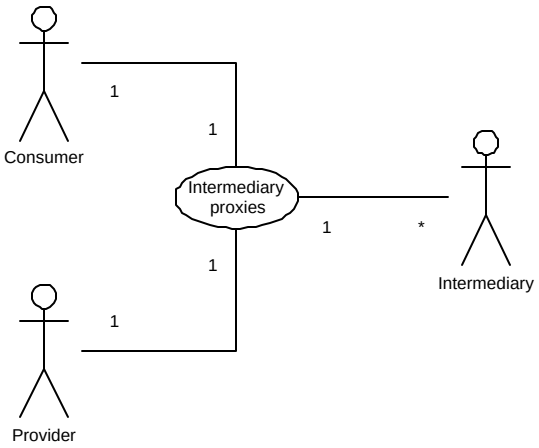
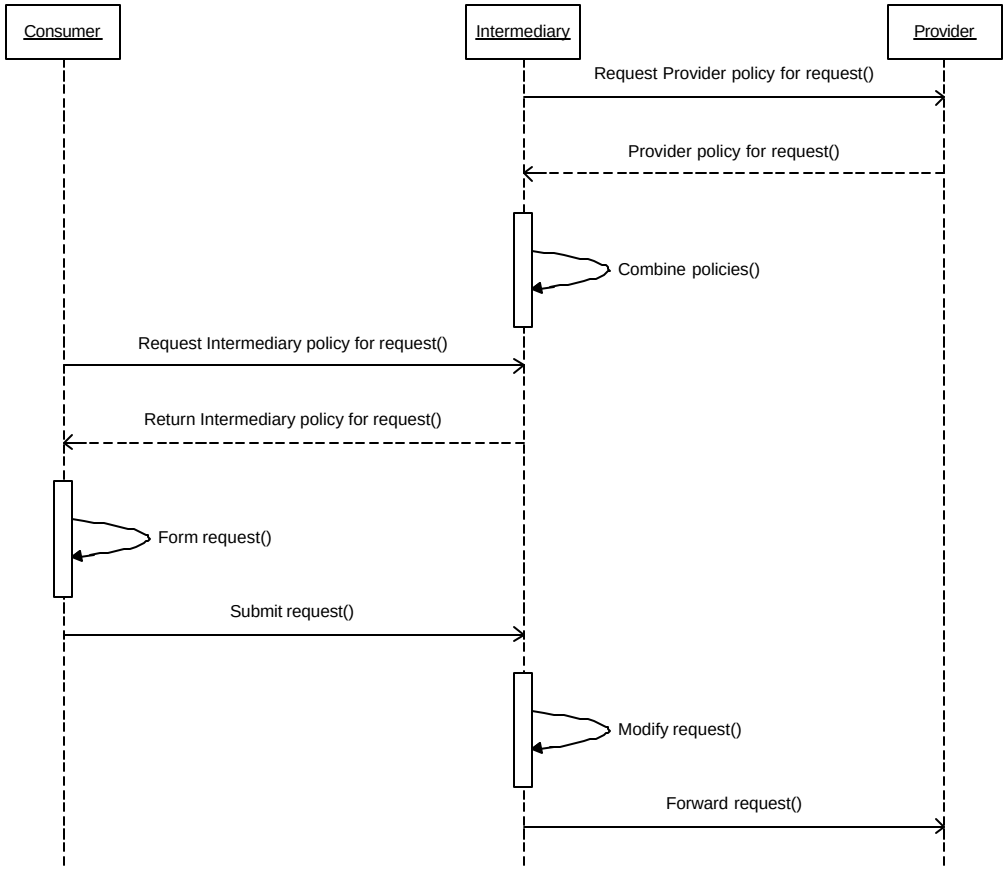


Figure 11 - Use-case 6

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.

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

2.7. Use-case 7: Intermediary intercepts

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

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

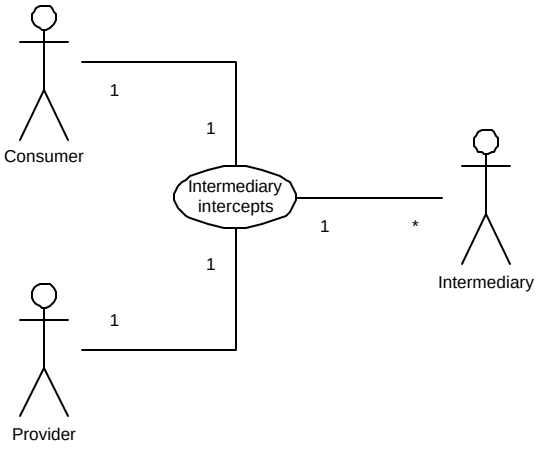


Figure 13 - Use-case 7

The corresponding sequence diagram is shown in Figure 14.

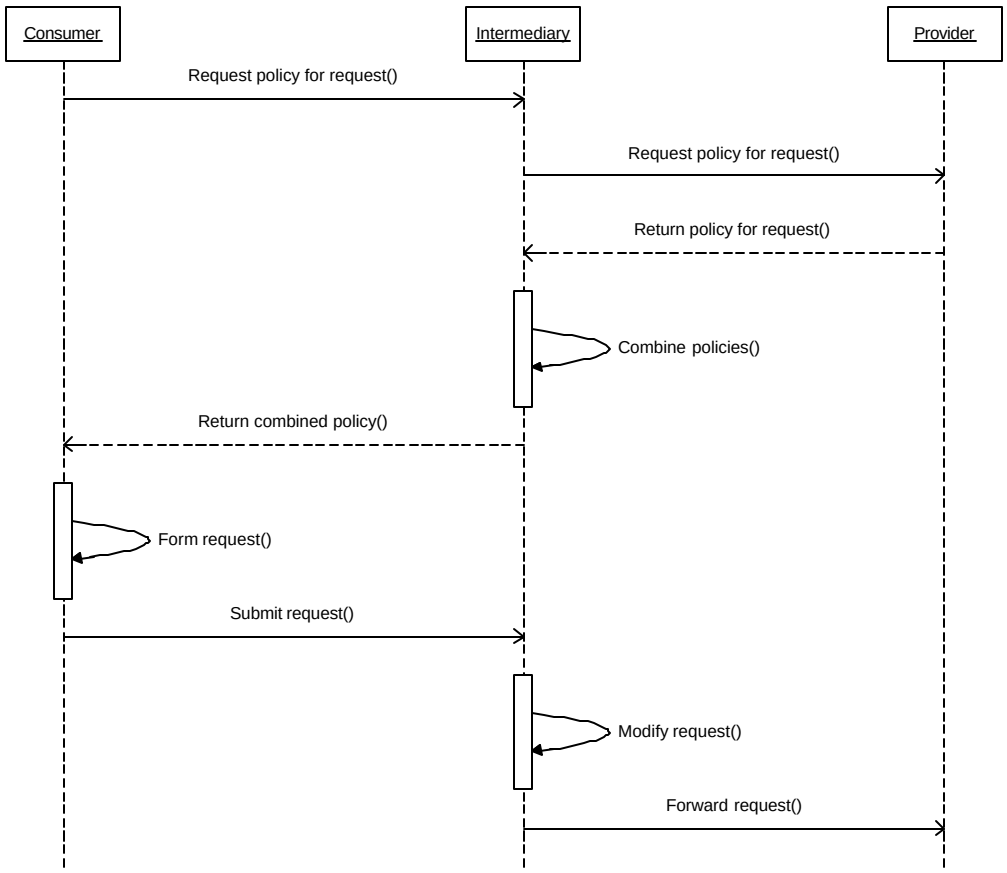


Figure 14 - Use-case 7 sequence

1. Consumer requests policy for requests from Provider. The request is intercepted by Intermediary.
2. Intermediary requests policy for requests from Provider.
3. Provider returns policy for requests to Intermediary.
4. Intermediary combines Provider's policy with its own.
5. Intermediary returns combined policy to Consumer.
6. Consumer forms a request in conformance with policy.
7. Consumer submits the request to Provider. The request is intercepted by Intermediary.
8. Intermediary modifies the request.
9. Intermediary forwards the modified request to Provider.

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

2.8. Use-case 8: Multiple sources

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

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

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

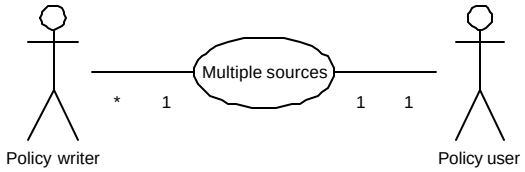


Figure 15 - Use-case 8

The corresponding sequence diagram is shown in Figure 16.

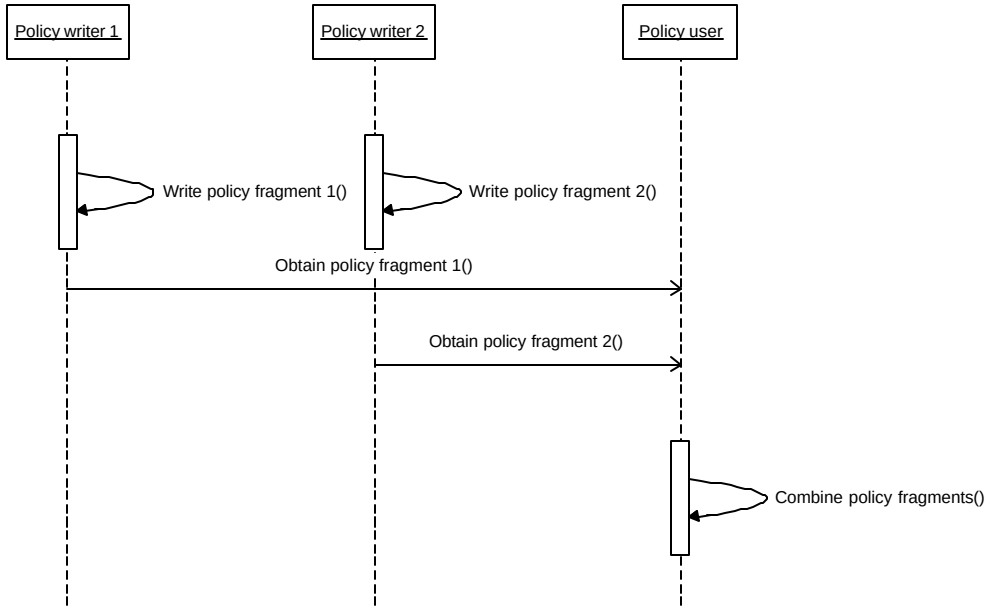


Figure 16 - Use-case 8 sequence

1. Policy writer 1 prepares policy fragment 1.
2. Policy writer 2 prepares policy fragment 2.
3. Policy user obtains policy fragment 1.
4. Policy user obtains policy fragment 2.
5. Policy user combines policy fragment 1 and policy fragment 2.

2.9. Use-case 9: Second party combines

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.

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.

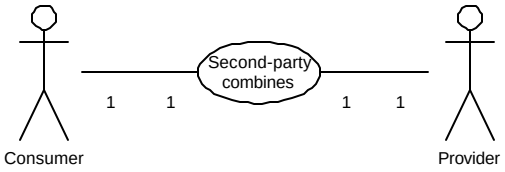


Figure 17 - Use-case 9

The corresponding sequence diagram is shown in Figure 18.

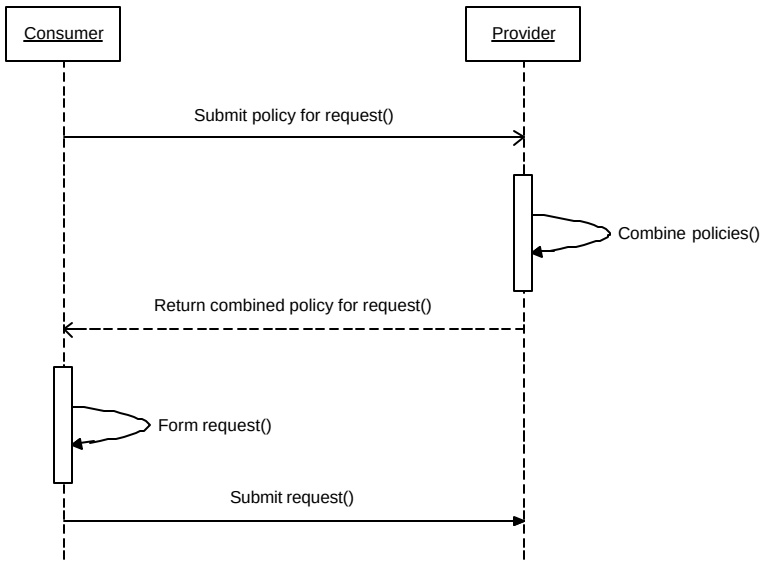


Figure 18 - Use-case 9 sequence

1. Consumer sends policy for request to Provider.
2. Provider combines Consumer's policy for request with its own.
3. Provider returns the combined policy to Consumer.
4. Consumer forms a request in conformance with the combined policy.
5. Consumer submits a request that conforms with the combined policy.

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

2.10. Use-case 10: Third party combines

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

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

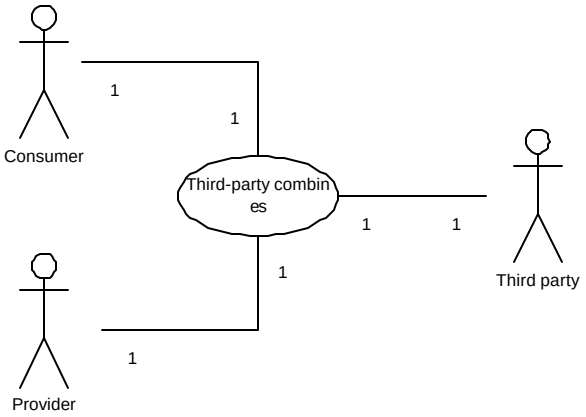


Figure 19 - Use-case 10

The corresponding sequence diagram is shown in Figure 20.

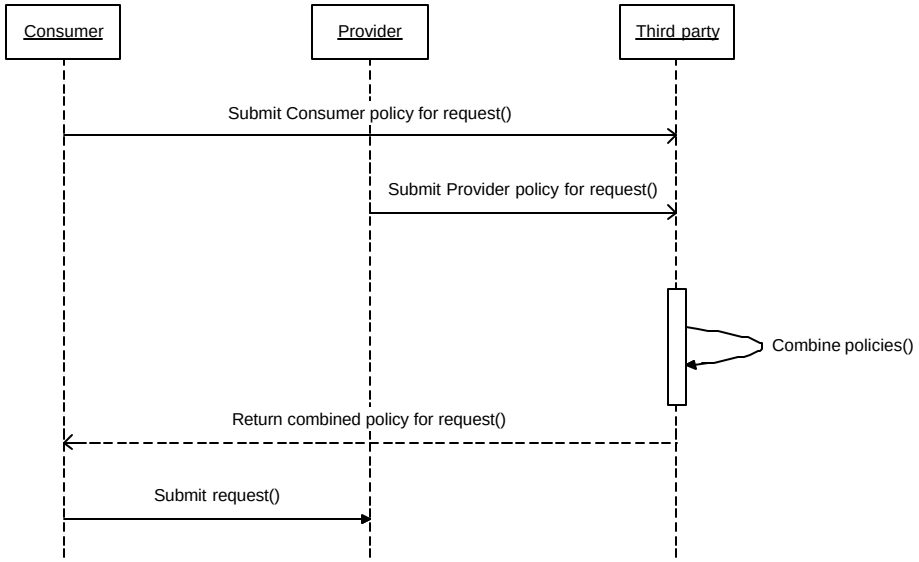


Figure 20 - Use-case 10 sequence

1. Consumer sends policy for request to Third party.
2. Provider sends policy for request to Third party.
3. Third party combines Consumer's policy for request with Provider's policy for request.
4. Third party returns the combined policy to Consumer.

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

There is a corresponding use-case for responses, in which Third party returns the combined policy to Provider in step 4, and, in step 5, Provider returns the response to Consumer.

2.11. Use-case 11: Third-party translates

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

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

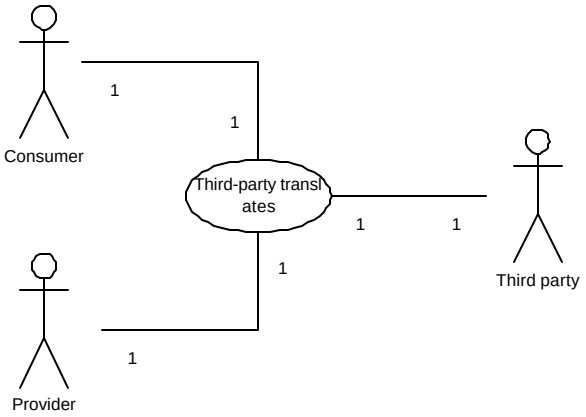


Figure 21 - Use-case 11

The corresponding sequence diagram is shown in Figure 22.

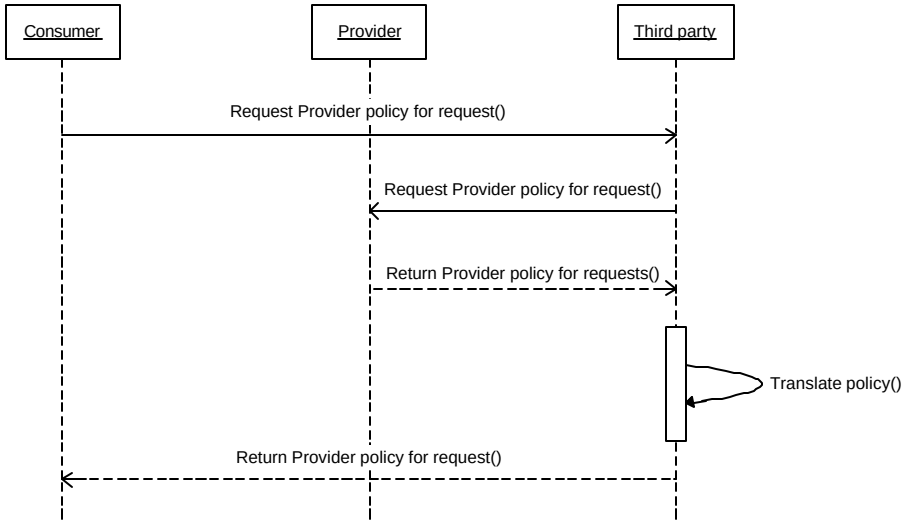


Figure 22 - Use-case 11 sequence

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

2. Third party requests the policy for requests from Provider.

3. Provider returns its policy for requests to Third party in its chosen syntax.

4. Third party translates the policy to the syntax chosen by Consumer.

5. Third party returns the policy to Consumer.

There is a corresponding use-case for responses, in which Third party translates Consumer policy in step 4 and returns it to Provider in step 5.

3. Policy communication

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

In the case of simple service provision, where Consumer sends an isolated service request to Provider, Provider may publish its policy in one or more of a number of ways, including: UDDI, WSDL, HTTP, LDAP, DNS or in SQL or SAML request/response messages.

In the case of complex service provision, the Provider and Consumer may communicate their policies to one another in-band, for instance, by including them as SOAP headers.

4. Language support

The policy language has to support alternative combinations of requirements, which gives rise to the need for logical combining operations, such as OR and AND. Support for reliable-messaging requirements gives rise to the need for integer comparison operations, such as greater-than and less-than, and support for cryptographic-security requirements gives rise to the need for set operations, such as subset and superset, over XML nodes and resource identifiers.

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

In some application domains, policies may be expressed as a set of independent ***objectives***, each of which may be achieved by any one of a number of alternative ***strategies***. Each strategy comprises a number of mandatory ***predicates***. There should be a suitable way of expressing policies of this form.

5. Requirements

5.1. Three-value logic

In order to support use-cases 1,2 and 6, it must be possible to evaluate an instance of policy to produce a Boolean result. A “True” result indicates that the requested action conforms with policy. A “False” result indicates that it does not. In the case that necessary information is unavailable, an “Indeterminate” result should be returned.

5.2. Amenable to combining

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

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

5.3. Interpretation as instructions

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

5.4. Common data-types

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

5.5. Extensible data-types

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

5.6. Common operators

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

5.7. Extensible operators

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

5.8. Multiple enforcement points

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

5.9. Multiple bindings

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

5.10. Preferences

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

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

5.11. Suppressed disclosure

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

5.12. Supported functions

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

5.13. Specified order

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

5.14. Policy identified by name

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

5.15. Attributes identified by name

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

5.16. Attributes identified by location

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

5.17. Behaviour in event attributes are unavailable

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

5.18. Version control

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

6. References

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

Appendix A. Notices

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

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

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