



# XACML v3.0 Administration and Delegation Profile Version 1.0

## Committee Draft 03

11 March 2010

### Specification URIs:

#### This Version:

<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-administration-v1-spec-cd-03-en.html>  
<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-administration-v1-spec-cd-03-en.doc>  
(Authoritative)  
<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-administration-v1-spec-cd-03-en.pdf>

#### Previous Version:

<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-administration-v1-spec-cd-1-en.html>  
<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-administration-v1-spec-cd-1-en.doc> (Authoritative)  
<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-administration-v1-spec-cd-1-en.pdf>

#### Latest Version:

<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-administration-v1-spec-en.html>  
<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-administration-v1-spec-en.doc> (Authoritative)  
<http://docs.oasis-open.org/xacml/3.0/xacml-3.0-administration-v1-spec-en.pdf>

### Technical Committee:

OASIS eXtensible Access Control Markup Language (XACML) TC

### Chair(s):

Bill Parducci, <[bill@parducci.net](mailto:bill@parducci.net)>  
Hal Lockhart, Oracle <[hal.lockhart@oracle.com](mailto:hal.lockhart@oracle.com)>

### Editor(s):

Erik Rissanen, Axiomatics AB <[erik@axiomatics.com](mailto:erik@axiomatics.com)>

### Related work:

This specification is related to:

- [eXtensible Access Control Markup Language \(XACML\) Version 3.0 Committee Draft 03](#)

### Declared XML Namespace(s):

None

### Abstract:

This specification describes a profile for XACML 3.0 to enable it to express administration and delegation policies.

### Status:

This document was last revised or approved by the eXtensible Access Control Markup Language (XACML) TC on the above date. The level of approval is also listed above. Check the "Latest Version" or "Latest Approved Version" location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the

“Send A Comment” button on the Technical Committee’s web page at <http://www.oasis-open.org/committees/xacml/>.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (<http://www.oasis-open.org/committees/xacml/ipr.php>).

The non-normative errata page for this specification is located at <http://www.oasis-open.org/committees/xacml/>.

---

## Notices

Copyright © OASIS® 2010. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

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

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

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

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' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on 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 implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The names "OASIS" and "XACML" are trademarks of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see <http://www.oasis-open.org/who/trademark.php> for above guidance.

---

# Table of Contents

1	Introduction.....	6
1.1	Terminology.....	6
1.2	Glossary.....	6
1.3	Normative References.....	6
1.4	Non-Normative References.....	7
2	Use Cases (non-normative).....	8
2.1	Administration/Delegation.....	8
2.1.1	Use case 1: Policy Administration.....	8
2.1.2	Use case 2: Dynamic Delegation.....	8
2.1.3	Discussion.....	8
2.2	Only if X is permitted to do it.....	8
3	Solution Overview and Semantics (non-normative).....	10
4	Processing Model.....	11
4.1	URIs.....	11
4.2	Reserved Attribute Categories.....	11
4.3	Trusted policies.....	11
4.4	The context handler.....	11
4.5	Administrative request generation during reduction.....	12
4.6	Policy set evaluation.....	12
4.7	Forming the reduction graph.....	13
4.8	Reduction of “Permit”.....	13
4.9	Reduction of “Deny”.....	13
4.10	Reduction of “Indeterminate”.....	14
4.11	Maximum delegation depth.....	14
4.12	Obligations.....	14
5	Example (non-normative).....	15
6	Optimization (non-normative).....	25
6.1	Optimization of Reduction.....	25
6.2	Alternative forms of delegation.....	25
7	Actions Other Than Create.....	26
7.1	Revocation by the issuer.....	26
7.2	Revocation by super administrators.....	26
7.3	Revocation as an action under access control.....	26
8	Security and Privacy Considerations (non-normative).....	27
8.1	Dynamic Issuer Attributes.....	27
8.2	Enforcing Constraints on Delegation.....	27
8.3	Issuer and delegate attributes.....	28
8.4	Denial of Service.....	28
8.5	Obligations.....	28
9	Conformance.....	29
9.1	Delegation by reduction.....	29
A.	Acknowledgements.....	30
B.	Revision History.....	31



---

# 1 Introduction

## 1.1 Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [RFC2119].

## 1.2 Glossary

For simplicity, this document uses the term **policy** to include the XACML definitions for both **policy** and **policy set**.

The following terms are defined.

### Access policy

A **policy** that governs access.

### Access request

A request to determine whether access to a resource should be granted.

### Administrative policy

A **policy** that authorizes a **delegate** to issue **policies** about constrained **situations**.

### Administrative request

A request to determine whether a **policy** was issued by an authorized source.

### Backward Chaining

Finding a chain of administrative and **access policies** beginning with an **access policy**, such that each **policy** is authorized by the next one.

### Delegate

Someone authorized by an **administrative policy** to issue **policies**.

### Forward Chaining

Finding a chain of administrative and **access policies** beginning at a **trusted policy**, such that each **policy** authorizes the next one.

### Issuer

A set of attributes describing the source of a **policy**.

### Reduction

the process by which the authority of a **policy** associated with an **issuer** is verified or discarded.

### Situation

A set of properties delineated by the <Attributes> elements of an **access request** context.

### Trusted policy

A **policy** without a <PolicyIssuer> element.

## 1.3 Normative References

- [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.

37        **[XACML]**            OASIS Committee Draft 03, *eXtensible access control markup language*  
38                            (*XACML*) *Version 3.0*. 11 March 2010. [http://docs.oasis-](http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cd-03-en.doc)  
39                            [open.org/xacml/3.0/xacml-3.0-core-spec-cd-03-en.doc](http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cd-03-en.doc)

40        **1.4 Non-Normative References**

41        **None**

---

## 2 Use Cases (non-normative)

This specification is intended to support the following use cases.

### 2.1 Administration/Delegation

#### 2.1.1 Use case 1: Policy Administration

**Policy** administration controls the types of **policies** that individuals can create and modify. Typically, different individuals would be allowed to create **policies** about certain sets of resources. Alternatively, administration might be divided up by action type, subject or some other properties.

In XACML 2.0 the question of the circumstances under which **policies** can be created is out of scope. It essentially says that some **policies** exist which the PDP will use.

#### 2.1.2 Use case 2: Dynamic Delegation

Dynamic delegation permits some users to create **policies** of limited duration to delegate certain capabilities to others. XACML 2.0 allows **policies** that say, "Mary can do something on behalf of Jack" by means of different subject-categories. But, it would be useful to allow people to generate **policies** on the fly that say such things as "while I am on vacation, Mary can approve requests." This requires the ability to create **policies** that control the **policies** that can be created.

#### 2.1.3 Discussion

In meeting these two use cases, it is NOT desirable to require either of the following to always be true:

1. Anything you can do, you can delegate to someone else to do.
2. If you can delegate something, you can always do it yourself by generating the necessary **policy** that applies to you.

It should be possible to create **policies** that enable #1 and/or #2, but they should not be "wired in."

The main difference between use cases #1 and #2 is how **policies** are accessed. In #1, most likely **policies** will be found in some repository or set of repositories. There will be some simple enforcement mechanism that says that the **issuer** of one **policy** must correspond to the person who created or modified the other **policy**. In #2, **policies** might need to be carried in application requests or accessed dynamically via some back channel. In this case, signatures, or some other such mechanism, would be used to verify the **issuer's** identity.

Note that in both cases, having a **policy** from Fred, signed by Fred does not mean the **policy** will be enforced. It merely means that it will be considered as a candidate. It is still necessary to authorize Fred's **policy** for it to be enforced.

It is also desirable to arrange for **policy** evaluation to be optimized by doing as much work prior to access time as possible. It should be possible to "flatten" **policy** chains to an equivalent form using whatever **policies** are at hand.

Support for administration/delegation should not reduce the existing functionality of XACML 2.0

### 2.2 Only if X is permitted to do it

Consider the common use case: Mary is the manager and approves expense reports for her department. When she is on vacation, Jack can approve expense reports.

We need a convenient way to say "Jack is allowed to do such and such, but only if Mary is allowed to do it". Mary might or might not be the **issuer** of this **policy**. In plain XACML, there is no way to do this except by duplicating the rules that apply to Mary.

82 In other words, we need a way to replace the access-subject in the request context with a specified  
83 subject, call the entire **policy** evaluation process and if the result is "Permit", then return a value of "True."

---

### 84 3 Solution Overview and Semantics (non-normative)

85 The purpose of the delegation model is to make it possible to express permissions about the right to issue  
86 **policies** and to verify issued **policies** against these permissions.

87 A **policy** may contain a <PolicyIssuer> element that describes the source of the **policy**. A missing  
88 <PolicyIssuer> element means that the **policy** is trusted.

89 A **trusted policy** is considered valid and its origin is not verified by the PDP.

90 **Policies** which have an **issuer** need to have their authority verified. The essence of the verification is that  
91 the **issuer** of the **policy** is checked against the **trusted policies**, directly or through other **policies** with  
92 **issuers**. During this check the right of the **issuer** to issue a **policy** about the current **access request** is  
93 verified.

94 If the authority of the **policy issuer** can be traced back to the **trusted policies**, the **policy** is used by the  
95 PDP, otherwise it is discarded as an unauthorized **policy**. The authority of the **issuer** depends on which  
96 access **situation** the current **access request** applies to, so a **policy** can be both valid and invalid  
97 depending on the **access request**.

98 Steps in the validation process are performed using a special case XACML requests, called  
99 **administrative requests**, which contain information about the **policy issuers** and the access **situation**.

---

## 100 4 Processing Model

### 101 4.1 URIs

102 urn:oasis:names:tc:xacml:3.0:delegation:decision

103 The identifier which MUST be used for the attribute indicating which type of decision is being  
104 reduced.

### 105 4.2 Reserved Attribute Categories

106 urn:oasis:names:tc:xacml:3.0:attribute-category:delegate

107 This attribute category MUST be used in **administrative requests** to carry the attributes of the  
108 **issuer** of the **policy** which is being reduced.

109 urn:oasis:names:tc:xacml:3.0:attribute-category:delegation-info

110 This attribute category MUST be used in **administrative requests** to carry information about the  
111 **reduction** in progress, such as the decision being reduced.

112 urn:oasis:names:tc:xacml:3.0:attribute-category:delegated:<anyURI>

113 Categories starting with this and ending with any URI MUST be used to carry information about  
114 the **situation** which is being reduced.

### 115 4.3 Trusted policies

116 In case there is no <PolicyIssuer> element in the **policy** or **policy set**, the **policy** or **policy set**  
117 MUST be trusted and no **reduction** of the **policy** will be performed.

### 118 4.4 The context handler

119 The attributes contained in an explicit <Attributes> element with Category  
120 “urn:oasis:names:tc:xacml:3.0:attribute-category:delegate” MAY be complemented with additional  
121 attributes by the context handler, as is the case with the other elements in the request context.

122 A dynamic **issuer** attribute is an attribute of an **issuer/delegate** such that the attribute value may have  
123 changed since the **policy** was issued. The time at which attributes are resolved is important for dynamic  
124 **delegate** attributes. The PDP and context handler MUST operate in either “current **issuer/delegate**  
125 attribute mode” or “historic **issuer/delegate** attribute mode” but not in both.

- 126 • Current attributes mode

127 In current attribute mode, when a **delegate** attribute is dynamic, the value of the attribute MUST  
128 be used as it is at the time of the **access request** being processed.

- 129 • Historic attributes mode

130 In historic attribute mode, when a **delegate** attribute is dynamic, the value of the attribute MUST  
131 be used as it was at the time when the **policy**, from which the **delegate** was derived, was issued.

132 These rules MUST apply to both attributes that appear in the <PolicyIssuer> element and the  
133 attributes that are retrieved by the context handler, which means that in case of the current attribute mode  
134 dynamic **issuer** attributes MUST NOT be present in the <PolicyIssuer> element.

135 See also the security considerations discussion related to this in section 8.1.

136 **4.5 Administrative request generation during reduction**

137 **Reduction** is the process by which the authority of **policies** is established. **Reduction** is performed as a  
 138 search in a graph. This section explains how a single **administrative request** is created to determine an  
 139 edge in the **reduction** graph. **Reduction** is always performed in the context of a request *R*, which is  
 140 being evaluated against a **policy set**.

141 Given a potentially supported **policy**, *P*, and the request *R*, an **administrative request**, *A*, is generated  
 142 based on *R* by the following steps:

- 143 1. The <Attributes> elements of *R* are mapped to <Attributes> elements in *A* according to  
 144 the following:
  - 145 a. An <Attributes> element with Category equal to  
 146 "urn:oasis:names:tc:xacml:3.0:attribute-category:delegate" in *R* has no corresponding  
 147 part in *A*.
  - 148 b. An <Attributes> element with Category which starts with the prefix  
 149 "urn:oasis:names:tc:xacml:3.0:attribute-category:delegated:" maps to an identical  
 150 <Attributes> element.
  - 151 c. An <Attributes> element with Category equal to  
 152 "urn:oasis:names:tc:xacml:3.0:attribute-category:delegation-info" in *R* has no  
 153 corresponding part in *A*. (Note, a new delegation-info category is created, see point 3  
 154 below.)
  - 155 d. An <Attributes> element with any other Category maps to an <Attributes>  
 156 element with the Category prefixed with "urn:oasis:names:tc:xacml:3.0:attribute-  
 157 category:delegated:" and identical contents.
- 158 2. *A* contains an <Attributes> element with Category equal to  
 159 "urn:oasis:names:tc:xacml:3.0:attribute-category:delegate" and contents identical to the  
 160 <PolicyIssuer> element from *P*.
- 161 3. *A* contains an <Attributes> element with Category equal to  
 162 "urn:oasis:names:tc:xacml:3.0:attribute-category:delegation-info" and the following contents:
  - 163 a. An <Attribute> element with AttributeId equal to  
 164 "urn:oasis:names:tc:xacml:3.0:delegation:decision", DataType equal to  
 165 "http://www.w3.org/2001/XMLSchema#string", and the value equal to the decision which  
 166 is being reduced, that is either "Permit" or "Deny". (See section 4.7 for explanation on  
 167 how this value is set.)

168 **4.6 Policy set evaluation**

169 This delegation profile defines how **policy sets** are evaluated in the presence of **policies** with **issuers**. A  
 170 PDP implementing this profile MUST perform **policy set** evaluation according the following process or a  
 171 process that produces an identical result in all cases. Note that the regular **policy set** evaluation  
 172 according to [XACML] is a special case of this process as long as no **policy** has an **issuer**.

173 The evaluation of a **policy set** is done as in [XACML], with the exception that the contained **policies** are  
 174 possibly reduced and/or discarded, before combination, as defined by the following table.

175

Value of evaluated policy	Policy Issuer	Action
Don't care	Absent	The value is combined as it is.
"Permit", "Deny" or "Indeterminate"	Present	The value is reduced as defined in sections 4.8, 4.9 and 4.10 respectively and possibly discarded before combination.

"Not applicable"	Present	The value is discarded.
------------------	---------	-------------------------

176

177 After the above actions have been performed, the remaining **trusted policy** values determine the value  
 178 of the **policy set** as defined in [XACML].

## 179 4.7 Forming the reduction graph

180 The **reduction** process is a graph search where the nodes of the graph are the **policies** in a **policy set**  
 181 and the edges represent how the **policies** authorize each other.

182 The nodes of the **reduction** graph are the **policies** of the **policy set**.

183 There are four kinds of directed edges in the graph: Types PP, PI, DP and DI.

184 Note (non-normative): Informally, the PP and DP edges are used to indicate whether a  
 185 **policy** authorizes delegation of "Permit" and "Deny" respectively. The PI and DI edges  
 186 are used to propagate "Indeterminate" results from **administrative policies** into the final  
 187 result. It is important to propagate "Indeterminate" results since failing to detect an error  
 188 can result in the wrong decision being implemented by the PEP.

189 To generate the edges of the **reduction** graph

- 190 1. For each ordered pair of **policies** in the **policy set** ( $P1, P2$ ), generate an **administrative request**  
 191  $A$  reducing "Permit" based on  $P1$  and the request being evaluated against the **policy set**.  
 192 a. Evaluate  $A$  against  $P2$ .  
 193 b. If and only if the result is "Permit", there is a PP edge from  $P1$  to  $P2$ .  
 194 c. If and only if the result is "Indeterminate", there is a PI edge from  $P1$  to  $P2$ .
- 195 2. For each ordered pair of **policies** in the **policy set** ( $P1, P2$ ), generate an **administrative request**  
 196  $A$  reducing "Deny" based on  $P1$  and the request being evaluated against the **policy set**.  
 197 a. Evaluate  $A$  against  $P2$ .  
 198 b. If and only if the result is "Permit", there is a DP edge from  $P1$  to  $P2$ .  
 199 c. If and only if the result is "Indeterminate", there is a DI edge from  $P1$  to  $P2$ .

## 200 4.8 Reduction of "Permit"

201 A **policy**,  $P$ , which evaluated to "Permit" in the **policy set**, MUST be reduced as follows in this section.

202 Form a reduction graph as described in section 4.7.

203 Start a graph search from the node corresponding to the **policy** to be reduced. Follow only PP edges. If it  
 204 is possible to reach a node which corresponds to a **trusted policy**, the **policy**  $P$  is treated as "Permit" in  
 205 combination of the **policy set**.

206 If it was not possible to reach a **trusted policy**, do a second graph search, following PP and PI edges. If  
 207 it possible to reach a **trusted policy** in this manner, the **policy**  $P$  is treated as "Indeterminate" in  
 208 combination of the **policy set**.

209 If it was not possible to reach a **trusted policy** with either search, the **policy**  $P$  is discarded and not  
 210 combined in the **policy set**.

211 In all graph searches, the maximum delegation depth limit MUST be checked as described in section  
 212 4.11.

213 In all graph searches obligations must be collected as described in section 4.12.

## 214 4.9 Reduction of "Deny"

215 A **policy**,  $P$ , which evaluated to "Deny" in the **policy set**, MUST be reduced as follows in this section.

216 Form a reduction graph as described in section 4.7.

217 Start a graph search from the node corresponding to the **policy** to be reduced. Follow only DP edges. If it  
218 is possible to reach a node which corresponds to a **trusted policy**, the **policy P** is treated as “Deny” in  
219 combination of the **policy set**.

220 If it was not possible to reach a **trusted policy**, do a second graph search, following DP and DI edges. If  
221 it possible to reach a **trusted policy** in this manner, the **policy P** is treated as “Indeterminate” in  
222 combination of the **policy set**.

223 If it was not possible to reach a **trusted policy** with either search, the **policy P** is discarded and not  
224 combined in the **policy set**.

225 In all graph searches, the maximum delegation depth limit MUST be checked as described in section  
226 4.11.

227 In all graph searches obligations must be collected as described in section 4.12.

## 228 4.10 Reduction of “Indeterminate”

229 A **policy, P**, which evaluated to “Indeterminate” in the **policy set**, MUST be reduced as follows in this  
230 section.

231 Form a reduction graph as described in section 4.7.

232 Start a graph search from the node corresponding to the **policy** to be reduced. Follow PP and PI edges.  
233 If it is possible to reach a node which corresponds to a **trusted policy**, the **policy P** is treated as  
234 “Indeterminate” in combination of the **policy set**.

235 If it was not possible to reach a **trusted policy**, do a second graph search, following DP and DI edges. If  
236 it possible to reach a **trusted policy** in this manner, the **policy P** is treated as “Indeterminate” in  
237 combination of the **policy set**.

238 If it was not possible to reach a **trusted policy** with either search, the **policy P** is discarded and not  
239 combined in the **policy set**.

240 In all graph searches, the maximum delegation depth limit MUST be checked as described in section  
241 4.11.

242 In all graph searches obligations must be collected as described in section 4.12.

243 Note (non-normative): This process is designed in this way because it is important to  
244 reduce “Indeterminate” results before combining them. An unauthorized “Indeterminate”  
245 can be used as an attack by forcing the PEP into error handling, and possibly denying or  
246 allowing access depending on the bias of the PEP. Intuitively we test if the **policy** would  
247 be authorized if it would have been “Permit” or “Deny”. If neither a “Permit” nor a “Deny”  
248 would have been authorized, the **policy** is not authorized, so the “Indeterminate” is  
249 discarded.

## 250 4.11 Maximum delegation depth

251 A **policy** or **policy set** MAY contain an XML attribute called `MaxDelegationDepth`, which limits the  
252 depth of delegation which is authorized by the **policy**. During the searches in the **reduction** graph, a path  
253 MUST be aborted if the number of nodes on the path exceeds the integer value of this attribute. The node  
254 count on the path includes the initial node which is being reduced, but does not include the node  
255 corresponding to the **policy** with the `MaxDelegationDepth` attribute being checked.

## 256 4.12 Obligations

257 Obligations in the **access policies** that have been reduced and are being combined are treated exactly  
258 as in [XACML]. **Administrative policies** may contain obligations but the obligations apply to the access  
259 decision, not the administrative decisions. All obligations that are found in **policies** that are used to  
260 reduce an **access policy** are treated as if they would have appeared in the **access policy**.

261 Due to security concerns with obligations, a **PDP** MAY refuse to load a policy with an obligation it does  
262 not recognize. Also, see Section 8.5 for security considerations concerning obligations.

263

## 5 Example (non-normative)

264

The following example *policy set* is used for illustrating the processing model.

265

266

```
<PolicySet PolicySetId="PolicySet1"
  Version="1.0"
  PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-
overrides"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
  xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 xacml-core-v3-
schema-wd-17.xsd">
  <Target/>
  <Policy PolicyId="Policy1"
    Version="1.0"
    RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-
overrides">
    <Target>
      <AnyOf>
        <AllOf>
          <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
              DataType="http://www.w3.org/2001/XMLSchema#string"
              >employee</AttributeValue>
            <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-
category:delegated:urn:oasis:names:tc:xacml:1.0:subject-category:access-subject "
              AttributeId="group"
              MustBePresent="false"
              DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </Match>
        </AllOf>
      </AnyOf>
      <AnyOf>
        <AllOf>
          <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
              DataType="http://www.w3.org/2001/XMLSchema#string">printer</AttributeValue>
            <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-
category:delegated:urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
              AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
              MustBePresent="false"
              DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </Match>
        </AllOf>
      </AnyOf>
      <AnyOf>
        <AllOf>
          <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
              DataType="http://www.w3.org/2001/XMLSchema#string">print</AttributeValue>
            <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-
category:delegated:urn:oasis:names:tc:xacml:3.0:attribute-category:action"
              AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
              MustBePresent="false"
              DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </Match>
        </AllOf>
      </AnyOf>
      <AnyOf>
        <AllOf>
          <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
              DataType="http://www.w3.org/2001/XMLSchema#string">Carol</AttributeValue>
            <AttributeDesignator
              Category="urn:oasis:names:tc:xacml:3.0:attribute-category:delegate"
              AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
              MustBePresent="false"
            >
```

328

```

329         DataType="http://www.w3.org/2001/XMLSchema#string"/>
330     </Match>
331 </AllOf>
332 </AnyOf>
333 </Target>
334 <Rule RuleId="Rule1" Effect="Permit">
335     <Target/>
336 </Rule>
337 </Policy>
338
339 <Policy PolicyId="Policy2"
340     Version="1.0"
341     RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-
342 overrides">
343     <PolicyIssuer>
344         <Attribute
345             IncludeInResult="false"
346             AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id">
347             <AttributeValue
348                 DataType="http://www.w3.org/2001/XMLSchema#string">Carol</AttributeValue>
349             </AttributeValue>
350         </PolicyIssuer>
351         <Target>
352             <AnyOf>
353                 <AllOf>
354                     <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
355                         <AttributeValue
356                             DataType="http://www.w3.org/2001/XMLSchema#string"
357                             >employee</AttributeValue>
358                         <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-
359 category:delegated;urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"
360                             AttributeId="group"
361                             MustBePresent="false"
362                             DataType="http://www.w3.org/2001/XMLSchema#string"/>
363                     </Match>
364                 </AllOf>
365             </AnyOf>
366             <AnyOf>
367                 <AllOf>
368                     <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
369                         <AttributeValue
370                             DataType="http://www.w3.org/2001/XMLSchema#string">printer</AttributeValue>
371                         <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-
372 category:delegated;urn:oasis:names:tc:xacml:3.0:attribute-category:resource"
373                             AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
374                             MustBePresent="false"
375                             DataType="http://www.w3.org/2001/XMLSchema#string"/>
376                     </Match>
377                 </AllOf>
378             </AnyOf>
379             <AnyOf>
380                 <AllOf>
381                     <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
382                         <AttributeValue
383                             DataType="http://www.w3.org/2001/XMLSchema#string">print</AttributeValue>
384                         <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-
385 category:delegated;urn:oasis:names:tc:xacml:3.0:attribute-category:action"
386                             AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
387                             MustBePresent="false"
388                             DataType="http://www.w3.org/2001/XMLSchema#string"/>
389                     </Match>
390                 </AllOf>
391             </AnyOf>
392             <AnyOf>
393                 <AllOf>
394                     <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
395                         <AttributeValue
396                             DataType="http://www.w3.org/2001/XMLSchema#string">Bob</AttributeValue>
397                         <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-
398 category:delegate"
399                             AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
400                             MustBePresent="false"
401                             DataType="http://www.w3.org/2001/XMLSchema#string"/>

```

```

402     </Match>
403   </AllOf>
404 </AnyOf>
405 </Target>
406 <Rule RuleId="Rule2" Effect="Permit">
407   <Target/>
408 </Rule>
409 </Policy>
410
411 <Policy PolicyId="Policy3"
412   Version="1.0"
413   RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-
414 overrides">
415   <PolicyIssuer>
416     <Attribute
417       IncludeInResult="false"
418       AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id">
419       <AttributeValue
420         DataType="http://www.w3.org/2001/XMLSchema#string">Mallory</AttributeValue>
421       </AttributeValue>
422     </PolicyIssuer>
423   <Target>
424     <AnyOf>
425       <AllOf>
426         <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
427           <AttributeValue
428             DataType="http://www.w3.org/2001/XMLSchema#string">Alice</AttributeValue>
429           <AttributeDesignator Category="urn:oasis:names:tc:xacml:1.0:subject-
430 category:access-subject"
431             AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
432             MustBePresent="false"
433             DataType="http://www.w3.org/2001/XMLSchema#string"/>
434           </Match>
435         </AllOf>
436       </AnyOf>
437     <AnyOf>
438       <AllOf>
439         <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
440           <AttributeValue
441             DataType="http://www.w3.org/2001/XMLSchema#string">printer</AttributeValue>
442           <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-
443 category:resource"
444             AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
445             MustBePresent="false"
446             DataType="http://www.w3.org/2001/XMLSchema#string"/>
447           </Match>
448         </AllOf>
449       </AnyOf>
450     <AnyOf>
451       <AllOf>
452         <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
453           <AttributeValue
454             DataType="http://www.w3.org/2001/XMLSchema#string">print</AttributeValue>
455           <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-
456 category:action"
457             AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
458             MustBePresent="false"
459             DataType="http://www.w3.org/2001/XMLSchema#string"/>
460           </Match>
461         </AllOf>
462       </AnyOf>
463     </Target>
464   <Rule RuleId="Rule3" Effect="Permit">
465     <Target/>
466   </Rule>
467 </Policy>
468
469 <Policy PolicyId="Policy4"
470   Version="1.0"
471   RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-
472 overrides">
473   <PolicyIssuer>
474     <Attribute

```

```

475         IncludeInResult="false"
476         AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id">
477         <AttributeValue
478 DataHref="http://www.w3.org/2001/XMLSchema#string">Bob</AttributeValue>
479         </AttributeValue>
480     </PolicyIssuer>
481     <Target>
482         <AnyOf>
483             <AllOf>
484                 <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
485                     <AttributeValue
486                         DataHref="http://www.w3.org/2001/XMLSchema#string">Alice</AttributeValue>
487                     <AttributeDesignator Category="urn:oasis:names:tc:xacml:1.0:subject-
488 category:access-subject"
489                         AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
490                         MustBePresent="false"
491                         DataHref="http://www.w3.org/2001/XMLSchema#string"/>
492                 </Match>
493             </AllOf>
494         </AnyOf>
495     </AnyOf>
496     <AllOf>
497         <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
498             <AttributeValue
499                 DataHref="http://www.w3.org/2001/XMLSchema#string">printer</AttributeValue>
500             <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-
501 category:resource"
502                 AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
503                 MustBePresent="false"
504                 DataHref="http://www.w3.org/2001/XMLSchema#string"/>
505             </Match>
506         </AllOf>
507     </AnyOf>
508 </AnyOf>
509 </AllOf>
510 <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
511     <AttributeValue
512         DataHref="http://www.w3.org/2001/XMLSchema#string">print</AttributeValue>
513     <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:attribute-
514 category:action"
515         AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
516         MustBePresent="false"
517         DataHref="http://www.w3.org/2001/XMLSchema#string"/>
518     </Match>
519 </AllOf>
520 </AnyOf>
521 </Target>
522 <Rule RuleId="Rule4" Effect="Permit">
523     <Target/>
524 </Rule>
525 </Policy>
526 </PolicySet>

```

527 *Listing 1 The sample policy set.*

528

529 The **policy set** contains four **policies**. Policy 1 is a **trusted policy** since it has no **issuer**. The target with  
530 the standard attribute categories for the subject, resource and action constrain the **situation** that the  
531 **policy** applies to. The **policy** could have defined additional constraints on the **situation** by an  
532 environment target or by conditions or by rule targets. In this case the **policy** allows granting **policies**  
533 about any **situation** which is an employee who prints on the printer. Since there are `<Match>` elements  
534 with delegated categories in the **policy** target, Policy 1 is an **administrative policy**. In this case the  
535 **policy** allows for Carol to create any **policy** which allows a **situation** that is also allowed by Policy 1, that  
536 is, Carol can give access to the printer to any employee. Since there is no limit on the delegation depth,  
537 Carol can also create an **administrative policy** over these **situations**.

538 Policy 2 is issued by Carol as is indicated by the `<PolicyIssuer>` element. The allowed **situations** are  
539 again that an employee prints on the printer. Again, since there are `<Match>` elements with delegated

540 categories, Policy 2 is an **administrative policy**. In this case Bob is granted the right to issue **policies**  
541 granting access to **situations** that are allowed by Policy 2.

542 Policy 3 is issued by Mallory, as is indicated by the <PolicyIssuer> element. The <Match> elements  
543 are on non-delegated categories, so it is an **access policy**. It grants access to the printer for Alice. As we  
544 will see later on, this **policy** is unauthorized since Mallory has not been authorized to allow access for this  
545 **situation** (Alice accessing the printer).

546 Policy 4 is issued by Bob as is indicated by the <PolicyIssuer> element. There are no delegated  
547 categories, so it is an **access policy**. It grants access to the printer for Alice.

548 We start with the following example **access request**. The request indicates that Alice is trying to access  
549 the printer. In this case Alice is also associated with the employee group attribute.

```

550
551 <Request
552   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
553   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
554   xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 xacml-core-v3-
555   schema-wd-17.xsd"
556   CombinedDecision="false"
557   ReturnPolicyIdList="false">
558   <Attributes Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
559     <Attribute
560       IncludeInResult="false"
561       AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id">
562       <AttributeValue
563         DataType="http://www.w3.org/2001/XMLSchema#string">Alice</AttributeValue>
564       </Attribute>
565     <Attribute
566       IncludeInResult="false"
567       AttributeId="group">
568       <AttributeValue
569         DataType="http://www.w3.org/2001/XMLSchema#string">employee</AttributeValue>
570       </Attribute>
571     </Attributes>
572   <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
573     <Attribute
574       IncludeInResult="false"
575       AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id">
576       <AttributeValue
577         DataType="http://www.w3.org/2001/XMLSchema#string">printer</AttributeValue>
578       </Attribute>
579     </Attributes>
580   <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
581     <Attribute
582       IncludeInResult="false"
583       AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id">
584       <AttributeValue
585         DataType="http://www.w3.org/2001/XMLSchema#string">print</AttributeValue>
586       </Attribute>
587     </Attributes>
588   </Request>

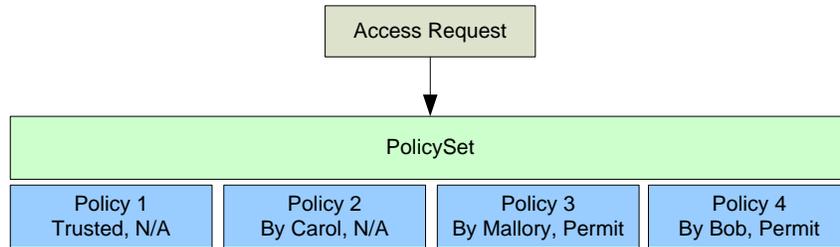
```

589 *Listing 2 The access request.*

590

591 The request is evaluated against the **policies** in the **policy set**. The request will not match the targets in  
592 Policy 1 or Policy 2 since there are no delegated categories in the request. Both Policy 3 and Policy 4 will  
593 evaluate to "Permit" since the targets match directly. This is illustrated in the following figure.

594



595

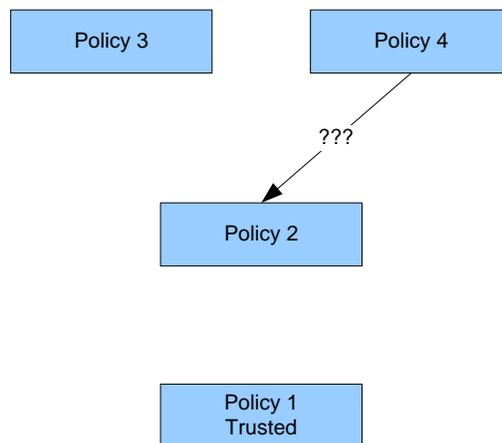
596

597 Policy 3 and Policy 4 need to be reduced since they are not trusted.

598 As specified in the processing model, **reduction** consists of two steps. First a **reduction** graph is built,  
 599 and then the PDP searches the graph for a path to the **trusted policies** for each **policy** with an **issuer**.  
 600 Note that this example follows the definition of the processing model and does not attempt to be efficient.  
 601 An efficient PDP can mix edge creation and path searching so that only those edges which are actually  
 602 needed are created. This example does not do so for simplicity and we create a full graph before we do a  
 603 search.

604 So, we begin by creating the **reduction** graph. Creating the **reduction** graph means finding any edges  
 605 between the **policies** in the **policy set**. We need to check each pair of **policies** for an edge (although in  
 606 practice a PDP may optimize the search to find a minimum set of edges as needed to determine the  
 607 result). First, consider the question whether there is any edge between Policy 4 and Policy 2:

608



609

610

611 As defined by the processing model, there is an edge if and only if the **administrative request** generated  
 612 from policy 4 evaluates to Permit (or Indeterminate) for policy 2. So to test for an edge, we create the  
 613 following **administrative request**, and evaluate it against Policy 2:

614

```

615 <Request
616   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
617   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
618   xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 xacml-core-v3-
619   schema-wd-17.xsd"
620   CombinedDecision="false"
621   ReturnPolicyIdList="false">
622   <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-
623   category:delegated:urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
624     <Attribute
625       IncludeInResult="false"
626       AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id">
627       <AttributeValue
628         DataType="http://www.w3.org/2001/XMLSchema#string">Alice</AttributeValue>
629     </Attribute>
630   </Attributes>
  
```

```

631         IncludeInResult="false"
632         AttributeId="group">
633         <AttributeValue
634 DataHref="http://www.w3.org/2001/XMLSchema#string">employee</AttributeValue>
635         </Attribute>
636     </Attributes>
637     <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:delegated:
638 urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
639         <Attribute
640             IncludeInResult="false"
641             AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id">
642             <AttributeValue
643 DataHref="http://www.w3.org/2001/XMLSchema#string">printer</AttributeValue>
644             </Attribute>
645         </Attributes>
646         <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:delegated:
647 urn:oasis:names:tc:xacml:3.0:attribute-category:action">
648             <Attribute
649                 IncludeInResult="false"
650                 AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id">
651                 <AttributeValue
652 DataHref="http://www.w3.org/2001/XMLSchema#string">print</AttributeValue>
653                 </Attribute>
654             </Attributes>
655             <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:delegate">
656                 <Attribute
657                     IncludeInResult="false"
658                     AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id">
659                     <AttributeValue
660 DataHref="http://www.w3.org/2001/XMLSchema#string">Bob</AttributeValue>
661                     </Attribute>
662                 <Attribute
663                     IncludeInResult="false"
664                     AttributeId="group">
665                     <AttributeValue
666 DataHref="http://www.w3.org/2001/XMLSchema#string">administrator</AttributeValue>
667                     </Attribute>
668                 </Attributes>
669             </Attributes>
670             <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:delegation-info">
671                 <Attribute
672                     IncludeInResult="false"
673                     AttributeId="urn:oasis:names:tc:xacml:3.0:delegation:decision">
674                     <AttributeValue
675 DataHref="http://www.w3.org/2001/XMLSchema#string">Permit</AttributeValue>
676                     </Attribute>
677                 </Attributes>
678     </Request>

```

679 *Listing 3 The administrative request for detecting edges from policy 4 to policy 2.*

680

681 The **administrative request** is created based on the request being evaluated against the whole **policy**  
682 **set** and the **issuer** of Policy 4, that is, Bob. The subject, resource and action from the **access request** in  
683 Listing 2 are transformed into delegated subject, resource and action in the **administrative request** in  
684 Listing 3 and the **issuer** of Policy 4 becomes the **delegate** of the **administrative request**. We perform  
685 the request with a permit decision initially.

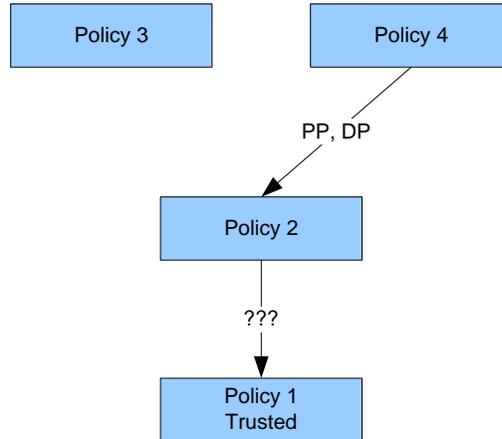
686 The interpretation of the **administrative request** is “Is Bob allowed to create a **policy** that concerns  
687 access to the printer for Alice?” In this case we also filled in the attribute representing membership in the  
688 administrators group for Bob in the request context. This represents the fact that the context handler can  
689 fill in attributes in the request context. (The details of how the context handler found the administrator  
690 attribute depend on the PDP implementation and the available attribute sources in the particular  
691 implementation.)

692 The request will evaluate to “Permit” on Policy 2. This means that there is a PP edge from Policy 4 to  
693 Policy 2, which represents that Policy 2 authorizes Policy 4 for a “Permit” decision on the particular  
694 **situation**. To test for a DP edge, another **administrative request** is created and evaluated. This request  
695 will have the same contents as the first one, except for a “Deny” decision in the delegation-info category.

696 (The request is not shown here. Also note that since policy 4 evaluated to “Permit”, the DP edge is not  
697 really needed, although it is specified in the definition of the graph, so this request could be skipped by an  
698 optimizing PDP.) This will also evaluate to “Permit”, so there is a DP edge as well. (It would have been  
699 possible for Policy 2 to include a condition so it would only allow a “Permit” decision, but this is not the  
700 case here.)

701 We have now established the edges going from Policy 4 to Policy 2. Next, we test for edges from Policy 2  
702 to Policy 1.

703



704

705

706 To test for PP and PI edges from Policy 2 to Policy 1, the following **administrative request** is generated:

707

```
708 <Request
709   xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
710   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
711   xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 xacml-core-v3-
712   schema-wd-17.xsd"
713   CombinedDecision="false"
714   ReturnPolicyIdList="false">
715   <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-
716   category:delegated:urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
717     <Attribute
718       IncludeInResult="false"
719       AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id">
720       <AttributeValue
721         DataType="http://www.w3.org/2001/XMLSchema#string">Alice</AttributeValue>
722       </AttributeValue>
723     </Attribute>
724     <Attribute
725       IncludeInResult="false"
726       AttributeId="group">
727       <AttributeValue
728         DataType="http://www.w3.org/2001/XMLSchema#string">employee</AttributeValue>
729       </AttributeValue>
730     </Attribute>
731   </Attributes>
732   <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:delegated:
733   urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
734     <Attribute
735       IncludeInResult="false"
736       AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id">
737       <AttributeValue
738         DataType="http://www.w3.org/2001/XMLSchema#string">printer</AttributeValue>
739       </AttributeValue>
740     </Attribute>
741   </Attributes>
742   <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:delegated:
743   urn:oasis:names:tc:xacml:3.0:attribute-category:action">
744     <Attribute
745       IncludeInResult="false"
746       AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id">
```

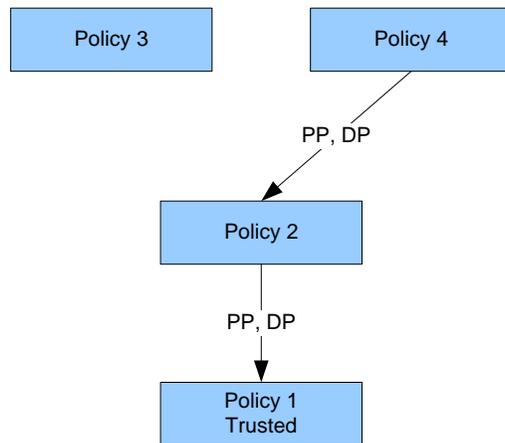
```

744     <AttributeValue
745     DataType="http://www.w3.org/2001/XMLSchema#string">print</AttributeValue>
746     </Attribute>
747   </Attributes>
748   <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:delegate">
749     <Attribute
750       IncludeInResult="false"
751       AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id">
752       <AttributeValue
753       DataType="http://www.w3.org/2001/XMLSchema#string">Carol</AttributeValue>
754       </Attribute>
755     </Attributes>
756   </Attributes>
757   <Attributes
758     Category="urn:oasis:names:tc:xacml:3.0:attribute-category:delegation-info">
759     <Attribute
760       IncludeInResult="false"
761       AttributeId="urn:oasis:names:tc:xacml:3.0:delegation:decision">
762       <AttributeValue
763       DataType="http://www.w3.org/2001/XMLSchema#string">Permit</AttributeValue>
764       </Attribute>
765     </Attributes>
766   </Request>

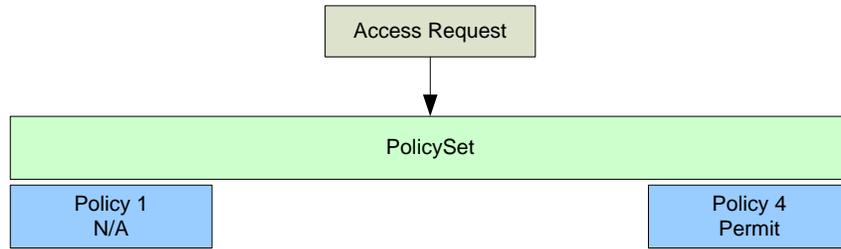
```

766 Listing 4 The administrative request for detecting edges from policy 2 to policy 1.

767  
768 Again, the subject, resource and action are copied from Listing 2 in to Listing 4 as delegated subject,  
769 resource and action and the **issuer** of policy 2, Carol, becomes the **delegate** of Listing 4. (In this case  
770 Carol is not a member of the administrator group so the context handler has not added such an attribute  
771 to Carol in this request.) This request and a corresponding request with a “Deny” decision evaluate to  
772 “Permit”, so we have found PP and DP edges. It remains to test the remaining combinations of nodes.  
773 These tests are not shown here to conserve space, but the end result will be a graph like this:  
774



775  
776  
777 This is the full **reduction** graph for the example.  
778 The second step of the PDP is now to find paths to the **trusted policies** from policies 3 and 4, which  
779 were the applicable **policies** to the original **access request**. In the graph we can see that there is a PP  
780 edged path to a **trusted policy** for Policy 4, so the Permit from Policy 4 is combined. There is no path for  
781 Policy 3, so policy 3 is disregarded. Policy 2 is not applicable and is not trusted, so it is also discarded.  
782 Policy 1 remains since it is trusted, although it is not applicable. We have the following:  
783



784

785

786

These **policies** are combined as usual, which in this case leads to a “Permit” for the **policy set** in whole.

787

## 6 Optimization (non-normative)

788

### 6.1 Optimization of Reduction

789

When **administrative policies** are simple and few in number, the previous process can be executed as written. However, when **policies** are numerous, preprocessing will help improve performance at access time. The following strategies may be employed.

790

791

792

- **Eliminate unauthorized polices**

793

Eliminating **administrative policies** for which there is no chain back to the **trusted policies** will greatly reduce the processing required at access time by eliminating backtracking. This works when **policies** are drawn exclusively from a repository. When **policies** may be presented dynamically at access time, it will be useful to limit what **policies** can be presented. For example, dynamic **policies** might be restricted to being only **access policies** or either access or leaf **administrative policies**. If root **policies** can be presented dynamically, then it will not be possible to perform this processing in advance.

794

795

796

797

798

799

800

- **Flatten delegation chains**

801

When a chain can be found from the **trusted policies** to a particular **access policy**, then a derived **trusted policy**, with the same allowed **situations** and effect value can be substituted for the original **access policy**.

802

803

804

- **Split policies**

805

It may be possible to split a **policy** into two (or more) simpler ones. For example, when a **policy** contains a disjunctive condition, it will be equivalent to two distinct **policies** each containing one of the alternatives, with the same effect value. The benefit of doing this is that it may then be possible to eliminate or flatten one of the derived **policies**.

806

807

808

809

- **Creating graph edges only as needed**

810

Typical **reduction** graphs are likely sparse, so rather than testing each pair of nodes, it may be more efficient to test for new edges as new nodes are reached with existing edges.

811

812

These optimizations may be done by **backward chaining**, **forward chaining** or both.

813

814

815

816

817

One of the main obstacles to performing these optimizations will be the lack of information about **situation** attributes in advance of access time so it will be possible to tell which **situation** constraint subsumes another. In particular implementations or applications the **policies** may have restricted forms, so the **situation** constraints are directly comparable or extra knowledge of attributes is available, such that comparisons between **situation** constraints can be made.

818

819

820

Since the **delegate** plays a particularly crucial role, and since the number of parties who are allowed to be **policy issuers** will typically be small compared to the total user population, it may be worthwhile to arrange that the authoritative source of these attributes be made available when doing optimizations.

821

### 6.2 Alternative forms of delegation

822

823

824

XACML **policies** are written in terms of attributes. This means that another way to achieve delegation, is to delegate attribute assignment, rather than XACML **policies**. Which is more efficient depends on the particular use case requirements.

825

826

827

For instance, if relatively few general rules can be used to express **policies**, and the requirement of delegation is to assign to whom these rules apply, delegation of attribute assignment may be more appropriate.

828

829

In contrast, for instance, if there are no general rules, and access permissions need to combine resources from many different authorities, the delegation model described in this profile may be ideal.

---

## 830 7 Actions Other Than Create

831 An **administrative policy** allows **policies** to be created by delegates. What about other operations on  
832 **policies**, such as Update and Delete?

833 Update (modify) can be treated as a Delete followed by a Create. In the case where **policies** are signed  
834 by the **policy issuer**, this is literally true

835 This profile does not specify a particular model for policy deletion (revocation of policies). An  
836 implementation MAY specify a model for policy deletion and may therefore disregard policies during  
837 processing. Revoked policies MAY also be removed from the policy repository, in which case they will not  
838 be seen by the PDP.

839 The following sections suggest some models for revocation which MAY be used. They are all optional  
840 and other models MAY be used as well.

### 841 7.1 Revocation by the issuer

842 One possible revocation model which may be implemented is that the **issuer** of a **policy** is the one who  
843 is authorized to remove it. How the **issuer** of the revocation is authenticated and how the effect of  
844 revocation is implemented is not specified by this profile.

### 845 7.2 Revocation by super administrators

846 One possible revocation model which may be implemented is that super administrators of the PDP (or  
847 **policy** repository) may remove any **policy** at their discretion.

### 848 7.3 Revocation as an action under access control

849 One possible revocation model is that access to the **policy** repository is controlled by XACML (or some  
850 other policy language) and removal of a **policy** is an action which can be performed. In this case the  
851 **policy** or the **policy** repository is modeled as a **resource** and the revocation as an action.

## 8 Security and Privacy Considerations (non-normative)

### 8.1 Dynamic Issuer Attributes

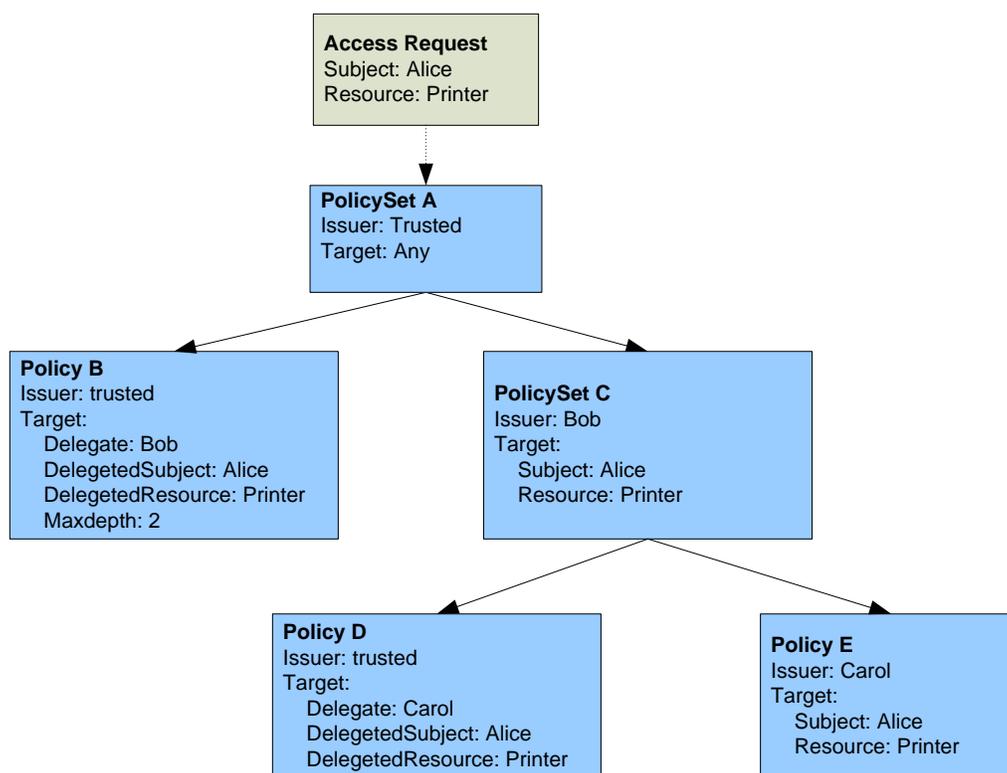
In case the attributes of an *issuer* may change with time, the choice of the point in time used for resolving them may affect the outcome of *administrative requests*. The PDP MUST treat this consistently and choose to operate in either historic or current *issuer* attribute mode. Policy writers need to be aware of the mode in which the PDP will operate.

Also in some environments it may be problematic to resolve old attributes and/or to reliably know at which time a *policy* was issued without special measures such as trusted time stamp authorities.

### 8.2 Enforcing Constraints on Delegation

This profile allows for defining a maximum depth for delegation. Implementers and users should be aware that this constraint cannot be enforced in the strict sense. It may be possible for someone with access rights to “delegate” that access right to anyone else “off-line” by just performing any operation himself on the behalf of the other person. However, in many applications these kinds of constraints can still be useful since they limit how the *policies* may evolve and indicate to users what policy is, and thus probably limiting casual policy violations.

Implementers should also be aware of that if there are nested *issuers* in a *policy set*, then the delegation that goes inside the outermost *issuer* is not visible to the outermost level of *reduction*. This means that constraints on delegation depth have no effect on the nested *issuers*. See the following figure for an example:



873  
874

875 During evaluation, **reduction** will be performed inside policy set C, where policy D will support policy E.  
876 This **reduction** is not visible outside policy set C. The maximum depth condition in policy B has no effect  
877 on the **reduction** which goes on inside policy set C. If you wish to use a maximum depth constraint, you  
878 must collect delegated **policies** at a single level of nesting in a **policy set**.

### 879 **8.3 Issuer and delegate attributes**

880 An implementation must take care to authenticate the contents of <PolicyIssuer> elements before the  
881 **policies** are included in the PDP. It is the responsibility of the entity issuing a **policy set** to verify that the  
882 attributes of all **issuers** of the immediately contained **policies** are correct. As a special case, it is the  
883 responsibility of the PDP owner to verify all **issuers** of the **policies** in the PDP at the PDP **policy set**  
884 level.

885 If the context handler provides additional attributes of delegates, naturally, the context handler must have  
886 verified their correctness.

887 A special case of **issuer** attribute verification is when the <PolicyIssuer> element is dynamically  
888 created when the **policy** is loaded from storage into the PDP. In this case the <PolicyIssuer> element  
889 could for instance be based on a digital signature on the **policy** in the storage.

### 890 **8.4 Denial of Service**

891 If an attacker can insert **policies** into the repository, even if the **issuers** of the **policies** would not be  
892 trusted and the **policy** could not be traced to a trusted source, it may be possible, depending on the  
893 implementation, for the attacker to draft **policies** such that there will be a lot of computation during  
894 request evaluation. This could degrade performance and result in denied or reduced service. An  
895 implementation must take this in consideration.

896 On case of such intensive computation is if the attacker is able to draft **policies** which contain complex  
897 conditional expressions.

898 Another identified attack is to create nested **policy sets** which contain **policies** which need to be  
899 reduced. Since creation of the **reduction** graph in worst case means that every **policy** will be evaluated  
900 twice, bu nesting **reduction** in **policy sets**, the number of times the deepest **policies** will be evaluated  
901 will increase exponentially with the depth of the **policy set** nesting. Possible protections against this  
902 attack include dynamic detection of it, not accepting **policies** with nested **policy sets** which need  
903 **reduction** and doing **reduction** graph generation by **forward chaining**, and not evaluate those **policies**  
904 which are not reached from the **trusted policies**.

### 905 **8.5 Obligations**

906 When **access policies** containing obligations are combined, an obligation from a **policy** will be included  
907 in the result, even if there is a **policy** evaluating to the same result but which does not contain the  
908 obligation. In a setting with decentralized administration where **policies** are issued by multiple **issuers**,  
909 this may in some cases be undesirable behavior. Depending on the nature of the obligation an obligation  
910 could be seen as an additional restriction to the access right. By adding an obligation to a **policy**, one  
911 **issuer** can in effect restrict the authority of another **issuer**. In particular, by including an obligation that is  
912 intentionally unrecognizable by the PEP, one **issuer** can completely deny the access that another **issuer**  
913 has granted.

914 When delegated XACML is used in an application, these issues must be considered. One possible  
915 solution is to allow only certain kinds of obligations. Another solution is to allow use of obligations only in  
916 the **trusted policies**.

917

918

919

---

920 **9 Conformance**

921 **9.1 Delegation by reduction**

922 An implementation conforms to this specification if it performs evaluation of XACML as specified in  
923 sections 4 and 7 of this document. The following URI identifies this functionality:

924 urn:oasis:names:tc:xacml:3.0:profile:administration:reduction

---

925 **A. Acknowledgements**

926 The following individuals have participated in the creation of this specification and are gratefully  
927 acknowledged:

- 928  
929 Anil Saldhana  
930 Anil Tappetla  
931 Anne Anderson  
932 Anthony Nadalin  
933 Bill Parducci  
934 Craig Forster  
935 David Chadwick  
936 David Staggs  
937 Dilli Arumugam  
938 Duane DeCouteau  
939 Erik Rissanen  
940 Gareth Richards  
941 Hal Lockhart  
942 Jan Herrmann  
943 John Tolbert  
944 Ludwig Seitz  
945 Michiharu Kudo  
946 Naomaru Itoi  
947 Paul Tyson  
948 Prateek Mishra  
949 Rich Levinson  
950 Ronald Jacobson  
951 Seth Proctor  
952 Sridhar Muppidi  
953 Tim Moses  
954 Vernon Murdoch  
955  
956

957

## B. Revision History

958 [optional; should not be included in OASIS Standards]

959

Revision	Date	Editor	Changes Made
WD 01	22 Mar 2005	Hal Lockhart	Initial working draft.
WD 02	8 Apr 2005	Tim Moses	Added PolicyIssuerMatch to <Target> element. Added delegation depth control.
WD 03	20 Apr 2005	Tim Moses	Added a pseudo-code description of the processing model Added schema for the request context.
WD 04	22 Apr 2005	Tim Moses	Added a plain-language description of the processing model. Modified <PolicyIssuerMatch> syntax and changed name to “delegates”. Made <PolicyIssuer> mandatory and included a URI for “root”.
WD 05			
WD 06			
WD 07	5 Jul 2005	Erik Rissanen	Added missing parts and corrected incorrect parts of the schema fragments. Clarified descriptive text. Added some new definitions in the terminology list. Fixed formatting.
WD 08	15 Aug 2005	Erik Rissanen	Improvements of the text, figures and formatting. Improved consistency and terminology. Fill in details, simplify and improve the processing model.
WD 09	13 Sep 2005	Erik Rissanen	Changed the definition of “situation”. Added max delegation depth to the processing model. Added obligations to the processing model. Added some security considerations. Changed IndirectDelegateDesignator to IndirectDelegatesCondition. Added the possibility for a target to match both access and administrative requests. Other improvements and corrections.
WD 10		Erik Rissanen	Removed the term <i>untrusted issuer</i> . It was confusing since it really meant “issuer not trusted yet”. Fixed some errors in the schema fragments

			<p>and the example.</p> <p>Removed the &lt;Policies&gt; element from the request. It will be placed in the SAML profile instead.</p> <p>Added historic/current attribute modes to the normative text.</p> <p>Made the effect part of the situation in order to support deny at the access level.</p> <p>Misc editing and fixing.</p>
WD 11	18 Jun 2006	Erik Rissanen	<p>Misc editing and corrections.</p> <p>Added description for the context &lt;Decision&gt; element.</p> <p>Added updated description of the access permitted function.</p> <p>Disallow even the trusted issuer to issue negative administrative decisions.</p>
WD 12	25 Jul 2006	Erik Rissanen	<p>Corrected typos.</p> <p>Added section with additions to the SAML profile of XACML.</p>
WD 13	4 Oct 2006	Erik Rissanen	<p>Updated to new OASIS document template.</p>
WD 14	5 Oct 2006	Erik Rissanen	<p>Fixed typos, formatting and clarified the text in multiple places.</p> <p>Removed statement in solution overview which stated that the policy which the PDP starts with by definition is issued by the trusted issuer. See issue #27 in the issues list.</p> <p>Major rewrite to make use of attribute categories.</p>
WD 15	4 Jan 2007	Erik Rissanen	<p>Clarified some of the text.</p>
WD 16		Erik Rissanen	<p>Removed indirect delegates.</p> <p>Updated XML based on new core schema.</p> <p>Removed section about SAML profile (moved into an updated SAML profile document).</p> <p>Removed sections about schema (moved to the core specification draft).</p> <p>Improved text and presentation.</p> <p>Updated processing model.</p>
WD 17		Erik Rissanen	<p>Changed to a reduction algorithm which handles indeterminate.</p> <p>Changed maximum depth to use a special XML attribute, rather being part of the request XACML attributes.</p> <p>Removed the non-normative overview of the processing model. It was not up to date and didn't really contribute anything beyond the examples.</p>

WD 18	24 Aug 2007	Erik Rissanen	Change disjunctive/conjunctive match to AnyOf/AllOf
WD 19	10 Oct 2007	Erik Rissanen	Fixed typos and improved descriptive text. Removed the trusted issuer. Rewrote the confusing section 4.
WD 20	28 Dec 2007	Erik Rissanen	Converted to current OASIS template.
WD 21	24 Feb 2008	Erik Rissanen	Added normative statement which for security reasons allows the PDP refuse policies which contain unknown obligations. Rewrote section on actions other than create and included some revocation models there. Updated the access-permitted function to the new generalized attribute categories.
WD 22	4 Nov 2008	Erik Rissanen	Moved the “access permitted” feature to the core specification.
WD 23	18 Mar 2009	Erik Rissanen	Fix error on treatment of delegation-info in section 4.5.
WD 24	4 Apr 2009	Erik Rissanen	Editorial cleanups Clarification of normative statements.
WD 25		Erik Rissanen	Fixed typos.
WD 26	17 Dec 2009	Erik Rissanen	Fixed formatting of OASIS references Updated acknowledgments
WD 27	12 Jan 2010	Erik Rissanen	Updated cross references Fixed the examples so the XML is valid against the XACML schema. Update acknowledgments
WD 28	8 Mar 2010	Erik Rissanen	Update cross references Fix OASIS style issues