



# eXtensible Access Control Markup Language (XACML) Version 2.0

Committee draft **04, 6 Dec** 2004

Document identifier: access\_control-xacml-2.0-core-spec-~~cd-04~~

Location: [http://docs.oasis-open.org/xacml/access\\_control-xacml-2.0-core-spec-~~cd-04~~.pdf](http://docs.oasis-open.org/xacml/access_control-xacml-2.0-core-spec-<del>cd-04</del>.pdf)

Editor:

Tim Moses, Entrust

Committee members:

Anne Anderson, Sun Microsystems  
Anthony Nadalin, IBM  
Bill Parducci, GlueCode Software  
Daniel Engovatov, BEA Systems  
Ed Coyne, Veterans Health Administration  
Frank Siebenlist, Argonne National Labs  
Hal Lockhart, BEA Systems  
Michael McIntosh, IBM  
Michiharu Kudo, IBM  
Polar Humenn, Self  
Ron Jacobson, Computer Associates  
Seth Proctor, Sun Microsystems  
Simon Godik, GlueCode Software  
Steve Anderson, OpenNetwork  
Tim Moses, Entrust

Abstract:

This specification defines version 2.0 of the extensible access-control markup language.

Status:

This version of the specification is an approved Committee Draft within the OASIS Access Control TC.

Access Control TC members should send comments on this specification to the [xacml@lists.oasis-open.org](mailto:xacml@lists.oasis-open.org) list. Others may use the following link and complete the comment form: [http://oasis-open.org/committees/comments/form.php?wg\\_abbrev=xacml](http://oasis-open.org/committees/comments/form.php?wg_abbrev=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 Access Control TC web page ([http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=xacml](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml)).

access\_control-xacml-2.0-core-spec-~~cd-04~~

Deleted: 3

Deleted: 10

Deleted: Nov

Deleted: cd-03

Field Code Changed

Deleted: cd-03

Deleted: cd-03

37 For any errata page for this specification, please refer to the Access Control TC web page  
38 ([http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=xacml](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml)).  
39 Copyright (C) OASIS Open 2004. All Rights Reserved.

## Table of contents

|       |   |    |
|-------|---|----|
| 1.    | Introduction (non-normative).....                       | 9  |
| 1.1.  | Glossary.....   | 9  |
| 1.1.1 | Preferred terms .....                                   | 9  |
| 1.1.2 | Related terms.....                                      | 10 |
| 1.2.  | Notation.....   | 11 |
| 1.3.  | Schema organization and namespaces .....                | 11 |
| 2.    | Background (non-normative).....                         | 12 |
| 2.1.  | Requirements.....                                       | 12 |
| 2.2.  | Rule and policy combining.....                          | 13 |
| 2.3.  | Combining algorithms.....                               | 13 |
| 2.4.  | Multiple subjects.....                                  | 14 |
| 2.5.  | Policies based on subject and resource attributes.....  | 14 |
| 2.6.  | Multi-valued attributes .....                           | 15 |
| 2.7.  | Policies based on resource contents .....               | 15 |
| 2.8.  | Operators.....  | 15 |
| 2.9.  | Policy distribution .....                               | 16 |
| 2.10. | Policy indexing.....                                    | 16 |
| 2.11. | Abstraction layer .....                                 | 16 |
| 2.12. | Actions performed in conjunction with enforcement ..... | 17 |
| 3.    | Models (non-normative).....                             | 17 |
| 3.1.  | Data-flow model .....                                   | 17 |
| 3.2.  | XACML context .....                                     | 19 |
| 3.3.  | Policy language model .....                             | 19 |
| 3.3.1 | Rule .....  | 20 |
| 3.3.2 | Policy .....  | 22 |
| 3.3.3 | Policy set.....   | 23 |
| 4.    | Examples (non-normative) .....                          | 23 |
| 4.1.  | Example one.....  | 24 |
| 4.1.1 | Example policy .....                                    | 24 |
| 4.1.2 | Example request context .....                           | 26 |
| 4.1.3 | Example response context.....                           | 27 |
| 4.2.  | Example two .....                                       | 27 |
| 4.2.1 | Example medical record instance .....                   | 28 |
| 4.2.2 | Example request context .....                           | 29 |
| 4.2.3 | Example plain-language rules.....                       | 31 |

Deleted: cd-03

|     |   |    |
|-----|---|----|
| 76  | 4.2.4 Example XACML rule instances.....                                       | 31 |
| 77  | 5. Policy syntax (normative, with the exception of the schema fragments)..... | 44 |
| 78  | 5.1. Element <PolicySet> .....  | 44 |
| 79  | 5.2. Element <Description> .....  | 46 |
| 80  | 5.3. Element <PolicySetDefaults> .....  | 46 |
| 81  | 5.4. Element <XPathVersion> .....   | 46 |
| 82  | 5.5. Element <Target> .....   | 46 |
| 83  | 5.6. Element <Subjects> .....   | 47 |
| 84  | 5.7. Element <Subject> .....  | 48 |
| 85  | 5.8. Element <SubjectMatch> .....   | 48 |
| 86  | 5.9. Element <Resources> .....  | 49 |
| 87  | 5.10. Element <Resource>.....   | 49 |
| 88  | 5.11. Element <ResourceMatch> .....   | 49 |
| 89  | 5.12. Element <Actions> .....   | 50 |
| 90  | 5.13. Element <Action> .....  | 50 |
| 91  | 5.14. Element <ActionMatch> .....   | 51 |
| 92  | 5.15. Element <Environments> .....  | 51 |
| 93  | 5.16. Element <Environment>.....  | 52 |
| 94  | 5.17. Element <EnvironmentMatch>.....   | 52 |
| 95  | 5.18. Element <PolicySetIdReference> .....                                    | 53 |
| 96  | 5.19. Element <PolicyIdReference> .....                                       | 53 |
| 97  | 5.20. Simple type VersionType.....  | 53 |
| 98  | 5.21. Simple type VersionMatchType .....                                      | 54 |
| 99  | 5.22. Element <Policy> .....  | 54 |
| 100 | 5.23. Element <PolicyDefaults> .....  | 56 |
| 101 | 5.24. Element <CombinerParameters> .....                                      | 56 |
| 102 | 5.25. Element <CombinerParameter> .....                                       | 57 |
| 103 | 5.26. Element <RuleCombinerParameters> .....                                  | 57 |
| 104 | 5.27. Element <PolicyCombinerParameters> .....                                | 58 |
| 105 | 5.28. Element <PolicySetCombinerParameters> .....                             | 58 |
| 106 | 5.29. Element <Rule> .....  | 59 |
| 107 | 5.30. Simple type EffectType.....   | 60 |
| 108 | 5.31. Element <VariableDefinition> .....                                      | 60 |
| 109 | 5.32. Element <VariableReference>.....  | 60 |
| 110 | 5.33. Element <Expression> .....  | 61 |
| 111 | 5.34. Element <Condition>.....  | 61 |
| 112 | 5.35. Element <Apply>.....  | 61 |

Deleted: cd-03

|     |   |    |
|-----|---|----|
| 113 | 5.36. Element <Function>.....   | 62 |
| 114 | 5.37. Complex type AttributeDesignatorType.....                               | 62 |
| 115 | 5.38. Element <SubjectAttributeDesignator>.....                               | 63 |
| 116 | 5.39. Element <ResourceAttributeDesignator>.....                              | 64 |
| 117 | 5.40. Element <ActionAttributeDesignator>.....                                | 65 |
| 118 | 5.41. Element <EnvironmentAttributeDesignator>.....                           | 65 |
| 119 | 5.42. Element <AttributeSelector>.....  | 65 |
| 120 | 5.43. Element <AttributeValue>.....   | 67 |
| 121 | 5.44. Element <Obligations>.....  | 67 |
| 122 | 5.45. Element <Obligation>.....   | 68 |
| 123 | 5.46. Element <AttributeAssignment>.....                                      | 68 |
| 124 | 6. Context syntax (normative with the exception of the schema fragments)..... | 69 |
| 125 | 6.1. Element <Request>.....   | 69 |
| 126 | 6.2. Element <Subject>.....   | 70 |
| 127 | 6.3. Element <Resource>.....  | 70 |
| 128 | 6.4. Element <ResourceContent>.....   | 71 |
| 129 | 6.5. Element <Action>.....  | 71 |
| 130 | 6.6. Element <Environment>.....   | 72 |
| 131 | 6.7. Element <Attribute>.....   | 72 |
| 132 | 6.8. Element <AttributeValue>.....  | 73 |
| 133 | 6.9. Element <Response>.....  | 73 |
| 134 | 6.10. Element <Result>.....   | 74 |
| 135 | 6.11. Element <Decision>.....   | 74 |
| 136 | 6.12. Element <Status>.....   | 75 |
| 137 | 6.13. Element <StatusCode>.....   | 75 |
| 138 | 6.14. Element <StatusMessage>.....  | 76 |
| 139 | 6.15. Element <StatusDetail>.....   | 76 |
| 140 | 6.16. Element <MissingAttributeDetail>.....                                   | 77 |
| 141 | 7. Functional requirements (normative).....                                   | 77 |
| 142 | 7.1. Policy enforcement point.....  | 77 |
| 143 | 7.1.1. Base PEP.....  | 78 |
| 144 | 7.1.2. Deny-biased PEP.....   | 78 |
| 145 | 7.1.3. Permit-biased PEP.....   | 78 |
| 146 | 7.2. Attribute evaluation.....  | 78 |
| 147 | 7.2.1. Structured attributes.....   | 78 |
| 148 | 7.2.2. Attribute bags.....  | 79 |
| 149 | 7.2.3. Multivalued attributes.....  | 79 |

Deleted: cd-03

|     |   |    |
|-----|---|----|
| 150 | 7.2.4. Attribute Matching.....                              | 79 |
| 151 | 7.2.5. Attribute Retrieval.....                             | 80 |
| 152 | 7.2.6. Environment Attributes.....                          | 80 |
| 153 | 7.3. Expression evaluation.....                             | 80 |
| 154 | 7.4. Arithmetic evaluation.....                             | 81 |
| 155 | 7.5. Match evaluation.....                                  | 81 |
| 156 | 7.6. Target evaluation.....                                 | 83 |
| 157 | 7.7. VariableReference Evaluation.....                      | 84 |
| 158 | 7.8. Condition evaluation.....                              | 84 |
| 159 | 7.9. Rule evaluation.....                                   | 84 |
| 160 | 7.10. Policy evaluation.....                                | 85 |
| 161 | 7.11. Policy Set evaluation.....                            | 86 |
| 162 | 7.12. Hierarchical resources.....                           | 87 |
| 163 | 7.13. Authorization decision.....                           | 87 |
| 164 | 7.14. Obligations.....                                      | 87 |
| 165 | 7.15. Exception handling.....                               | 87 |
| 166 | 7.15.1. Unsupported functionality.....                      | 88 |
| 167 | 7.15.2. Syntax and type errors.....                         | 88 |
| 168 | 7.15.3. Missing attributes.....                             | 88 |
| 169 | 8. XACML extensibility points (non-normative).....          | 88 |
| 170 | 8.1. Extensible XML attribute types.....                    | 89 |
| 171 | 8.2. Structured attributes.....                             | 89 |
| 172 | 9. Security and privacy considerations (non-normative)..... | 89 |
| 173 | 9.1. Threat model.....                                      | 90 |
| 174 | 9.1.1. Unauthorized disclosure.....                         | 90 |
| 175 | 9.1.2. Message replay.....                                  | 90 |
| 176 | 9.1.3. Message insertion.....                               | 90 |
| 177 | 9.1.4. Message deletion.....                                | 91 |
| 178 | 9.1.5. Message modification.....                            | 91 |
| 179 | 9.1.6. NotApplicable results.....                           | 91 |
| 180 | 9.1.7. Negative rules.....                                  | 91 |
| 181 | 9.2. Safeguards.....  | 92 |
| 182 | 9.2.1. Authentication.....                                  | 92 |
| 183 | 9.2.2. Policy administration.....                           | 92 |
| 184 | 9.2.3. Confidentiality.....                                 | 93 |
| 185 | 9.2.4. Policy integrity.....                                | 93 |
| 186 | 9.2.5. Policy identifiers.....                              | 94 |

Deleted: cd-03

|     |   |     |
|-----|---|-----|
| 187 | 9.2.6. Trust model.....                               | 94  |
| 188 | 9.2.7. Privacy.....                                   | 94  |
| 189 | 10. Conformance (normative).....                      | 95  |
| 190 | 10.1. Introduction.....                               | 95  |
| 191 | 10.2. Conformance tables.....                         | 95  |
| 192 | 10.2.1. Schema elements.....                          | 95  |
| 193 | 10.2.2. Identifier Prefixes.....                      | 96  |
| 194 | 10.2.3. Algorithms.....                               | 96  |
| 195 | 10.2.4. Status Codes.....                             | 97  |
| 196 | 10.2.5. Attributes.....                               | 97  |
| 197 | 10.2.6. Identifiers.....                              | 97  |
| 198 | 10.2.7. Data-types.....                               | 98  |
| 199 | 10.2.8. Functions.....                                | 98  |
| 200 | 11. References.....                                   | 102 |
| 201 | Appendix A. Data-types and functions (normative)..... | 105 |
| 202 | A.1. Introduction.....                                | 105 |
| 203 | A.2. Data-types.....                                  | 105 |
| 204 | A.3. Functions.....                                   | 107 |
| 205 | A.3.1 Equality predicates.....                        | 107 |
| 206 | A.3.2 Arithmetic functions.....                       | 109 |
| 207 | A.3.3 String conversion functions.....                | 110 |
| 208 | A.3.4 Numeric data-type conversion functions.....     | 110 |
| 209 | A.3.5 Logical functions.....                          | 110 |
| 210 | A.3.6 Numeric comparison functions.....               | 111 |
| 211 | A.3.7 Date and time arithmetic functions.....         | 112 |
| 212 | A.3.8 Non-numeric comparison functions.....           | 113 |
| 213 | A.3.9 String functions.....                           | 116 |
| 214 | A.3.10 Bag functions.....                             | 116 |
| 215 | A.3.11 Set functions.....                             | 117 |
| 216 | A.3.12 Higher-order bag functions.....                | 117 |
| 217 | A.3.13 Regular-expression-based functions.....        | 124 |
| 218 | A.3.14 Special match functions.....                   | 125 |
| 219 | A.3.15 XPath-based functions.....                     | 126 |
| 220 | A.3.16 Extension functions and primitive types.....   | 126 |
| 221 | Appendix B. XACML identifiers (normative).....        | 127 |
| 222 | B.1. XACML namespaces.....                            | 127 |
| 223 | B.2. Access subject categories.....                   | 127 |

Deleted: cd-03

|     |   |                     |
|-----|---|---------------------|
| 224 | <a href="#">B.3. Data-types .....</a>                             | <a href="#">127</a> |
| 225 | <a href="#">B.4. Subject attributes .....</a>                     | <a href="#">128</a> |
| 226 | <a href="#">B.6. Resource attributes .....</a>                    | <a href="#">129</a> |
| 227 | <a href="#">B.7. Action attributes .....</a>                      | <a href="#">129</a> |
| 228 | <a href="#">B.8. Environment attributes .....</a>                 | <a href="#">130</a> |
| 229 | <a href="#">B.9. Status codes .....</a>                           | <a href="#">130</a> |
| 230 | <a href="#">B.10. Combining algorithms .....</a>                  | <a href="#">130</a> |
| 231 | <a href="#">Appendix C. Combining algorithms (normative).....</a> | <a href="#">132</a> |
| 232 | <a href="#">C.1. Deny-overrides .....</a>                         | <a href="#">132</a> |
| 233 | <a href="#">C.2. Ordered-deny-overrides.....</a>                  | <a href="#">134</a> |
| 234 | <a href="#">C.3. Permit-overrides .....</a>                       | <a href="#">134</a> |
| 235 | <a href="#">C.4. Ordered-permit-overrides.....</a>                | <a href="#">136</a> |
| 236 | <a href="#">C.5. First-applicable.....</a>                        | <a href="#">136</a> |
| 237 | <a href="#">C.6. Only-one-applicable.....</a>                     | <a href="#">138</a> |
| 238 | <a href="#">Appendix D. Acknowledgments .....</a>                 | <a href="#">140</a> |
| 239 | <a href="#">Appendix E. Revision history .....</a>                | <a href="#">141</a> |
| 240 | <a href="#">Appendix F. Notices .....</a>                         | <a href="#">142</a> |
| 241 | ▼ .....   |                     |

Deleted: 1. Introduction (non-normative) 9¶

1.1. Glossary 9¶

1.1.1 Preferred terms 9¶

1.1.2 Related terms 10¶

1.2. Notation 11¶

1.3. Schema organization and namespaces 11¶

2. Background (non-normative) 12¶

2.1. Requirements 12¶

2.2. Rule and policy combining 13¶

2.3. Combining algorithms 13¶

2.4. Multiple subjects 14¶

2.5. Policies based on subject and resource attributes 14¶

2.6. Multi-valued attributes 15¶

2.7. Policies based on resource contents 15¶

2.8. Operators 15¶

2.9. Policy distribution 16¶

2.10. Policy indexing 16¶

2.11. Abstraction layer 16¶

2.12. Actions performed in conjunction with enforcement 17¶

3. Models (non-normative) 17¶

3.1. Data-flow model 17¶

3.2. XACML context 19¶

3.3. Policy language model 19¶

3.3.1 Rule 21¶

3.3.2 Policy 22¶

3.3.3 Policy set 23¶

4. Examples (non-normative) 23¶

4.1. Example one 24¶

4.1.1 Example policy 24¶

4.1.2 Example request context 26¶

4.1.3 Example response context 27¶

4.2. Example two 27¶

4.2.1 Example medical record instance 27¶

4.2.2 Example request context 29¶

4.2.3 Example plain-language rules 30¶

4.2.4 Example XACML rule instances 31¶

5. Policy syntax (normative, with the exception of the schema fragments) 43¶

5.1. Element <PolicySet> 43¶

5.2. Element <Description> 46¶

5.3. Element <PolicySetDefaults> 46¶

5.4. Element <XPathVersion> 46¶

5.5. Element <Target> 46¶

5.6. Element <Subjects> 47¶

5.7. Element <Subject> 47¶

5.8. Element <SubjectMatch> 48¶

5.9. Element <Resources> 48¶

5.10. Element <Resource> 49¶

5.11. Element <ResourceMatch> 49¶

5.12. Element <Actions> 50¶

5.13. Element <Action> 50¶

5.14. Element <ActionMatch> 50¶

5.15. Element <Environments> 51¶

5.16. Element <Environment> 51¶

5.17. Element <EnvironmentMatch> 52¶

5.18. Element ... [1]

Deleted: cd-03



## 1. Introduction (non-normative)

### 1.1. Glossary

#### 1.1.1 Preferred terms

**Access** - Performing an *action*

**Access control** - Controlling *access* in accordance with a *policy*

**Action** - An operation on a *resource*

**Applicable policy** - The set of *policies* and *policy sets* that governs *access* for a specific *decision request*

**Attribute** - Characteristic of a *subject*, *resource*, *action* or *environment* that may be referenced in a *predicate* or *target* (see also – *named attribute*)

**Authorization decision** - The result of evaluating *applicable policy*, returned by the *PDP* to the *PEP*. A function that evaluates to "Permit", "Deny", "Indeterminate" or "NotApplicable", and (optionally) a set of *obligations*

**Bag** – An unordered collection of values, in which there may be duplicate values

**Condition** - An expression of *predicates*. A function that evaluates to "True", "False" or "Indeterminate"

**Conjunctive sequence** - a sequence of *predicates* combined using the logical 'AND' operation

**Context** - The canonical representation of a *decision request* and an *authorization decision*

**Context handler** - The system entity that converts *decision requests* in the native request format to the XACML canonical form and converts *authorization decisions* in the XACML canonical form to the native response format

**Decision** – The result of evaluating a *rule*, *policy* or *policy set*

**Decision request** - The request by a *PEP* to a *PDP* to render an *authorization decision*

**Disjunctive sequence** - a sequence of *predicates* combined using the logical 'OR' operation

**Effect** - The intended consequence of a satisfied *rule* (either "Permit" or "Deny")

**Environment** - The set of *attributes* that are relevant to an *authorization decision* and are independent of a particular *subject*, *resource* or *action*

Deleted: cd-03

270 **Named attribute** – A specific instance of an **attribute**, determined by the **attribute** name and type,  
 271 the identity of the **attribute** holder (which may be of type: **subject**, **resource**, **action** or  
 272 **environment**) and (optionally) the identity of the issuing authority

273 **Obligation** - An operation specified in a **policy** or **policy set** that should be performed by the **PEP**  
 274 in conjunction with the enforcement of an **authorization decision**

275 **Policy** - A set of **rules**, an identifier for the **rule-combining algorithm** and (optionally) a set of  
 276 **obligations**. May be a component of a **policy set**

277 **Policy administration point (PAP)** - The system entity that creates a **policy** or **policy set**

278 **Policy-combining algorithm** - The procedure for combining the **decision** and **obligations** from  
 279 multiple **policies**

280 **Policy decision point (PDP)** - The system entity that evaluates **applicable policy** and renders an  
 281 **authorization decision**. This term is defined in a joint effort by the IETF Policy Framework  
 282 Working Group and the Distributed Management Task Force (DMTF)/Common Information Model  
 283 (CIM) in [RFC3198]. This term corresponds to "Access Decision Function" (ADF) in [ISO10181-3].

284 **Policy enforcement point (PEP)** - The system entity that performs **access control**, by making  
 285 **decision requests** and enforcing **authorization decisions**. This term is defined in a joint effort by  
 286 the IETF Policy Framework Working Group and the Distributed Management Task Force  
 287 (DMTF)/Common Information Model (CIM) in [RFC3198]. This term corresponds to "Access  
 288 Enforcement Function" (AEF) in [ISO10181-3].

289 **Policy information point (PIP)** - The system entity that acts as a source of **attribute** values

290 **Policy set** - A set of **policies**, other **policy sets**, a **policy-combining algorithm** and (optionally) a  
 291 set of **obligations**. May be a component of another **policy set**

292 **Predicate** - A statement about **attributes** whose truth can be evaluated

293 **Resource** - Data, service or system component

294 **Rule** - A **target**, an **effect** and a **condition**. A component of a **policy**

295 **Rule-combining algorithm** - The procedure for combining **decisions** from multiple **rules**

296 **Subject** - An actor whose **attributes** may be referenced by a **predicate**

297 **Target** - The set of **decision requests**, identified by definitions for **resource**, **subject** and **action**,  
 298 that a **rule**, **policy** or **policy set** is intended to evaluate

299 **Type Unification** - The method by which two type expressions are "unified". The type expressions  
 300 are matched along their structure. Where a type variable appears in one expression it is then  
 301 "unified" to represent the corresponding structure element of the other expression, be it another  
 302 variable or subexpression. All variable assignments must remain consistent in both structures.  
 303 Unification fails if the two expressions cannot be aligned, either by having dissimilar structure, or by  
 304 having instance conflicts, such as a variable needs to represent both "xs:string" and "xs:integer".  
 305 For a full explanation of **type unification**, please see [Hancock].

### 306 1.1.2 Related terms

307 In the field of access control and authorization there are several closely related terms in common  
 308 use. For purposes of precision and clarity, certain of these terms are not used in this specification.

access\_control-xacml-2.0-core-spec-[cd-04](#)

Deleted: cd-03

309 For instance, the term **attribute** is used in place of the terms: group and role.  
310 In place of the terms: privilege, permission, authorization, entitlement and right, we use the term  
311 **rule**.  
312 The term object is also in common use, but we use the term **resource** in this specification.  
313 Requestors and initiators are covered by the term **subject**.

## 314 1.2. Notation

315 This specification contains schema conforming to W3C XML Schema and normative text to  
316 describe the syntax and semantics of XML-encoded policy statements.

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

320 *"they MUST only be used where it is actually required for interoperation or to limit*  
321 *behavior which has potential for causing harm (e.g., limiting retransmissions)"*

322 These keywords are thus capitalized when used to unambiguously specify requirements over  
323 protocol and application features and behavior that affect the interoperability and security of  
324 implementations. When these words are not capitalized, they are meant in their natural-language  
325 sense.

326 Listings of XACML schema appear like this.

327  
328 [a01] Example code listings appear like this.

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

- 332 • The prefix `xacml:` stands for the XACML policy namespace.
- 333 • The prefix `xacml-context:` stands for the XACML context namespace.
- 334 • The prefix `ds:` stands for the W3C XML Signature namespace [DS].
- 335 • The prefix `xs:` stands for the W3C XML Schema namespace [XS].
- 336 • The prefix `xf:` stands for the XQuery 1.0 and XPath 2.0 Function and Operators  
337 specification namespace [XF].

338 This specification uses the following typographical conventions in text: `<XACMLElement>`,  
339 `<ns:ForeignElement>`, `Attribute`, **Datatype**, `OtherCode`. Terms in **italic bold-face** are  
340 intended to have the meaning defined in the Glossary.

## 341 1.3. Schema organization and namespaces

342 The XACML policy syntax is defined in a schema associated with the following XML namespace:

343 `urn:oasis:names:tc:xacml:2.0:policy`

344 The XACML context syntax is defined in a schema associated with the following XML namespace:

345 `urn:oasis:names:tc:xacml:2.0:context`

## 2. Background (non-normative)

The "economics of scale" have driven computing platform vendors to develop products with very generalized functionality, so that they can be used in the widest possible range of situations. "Out of the box", these products have the maximum possible privilege for accessing data and executing software, so that they can be used in as many application environments as possible, including those with the most permissive security policies. In the more common case of a relatively restrictive security policy, the platform's inherent privileges must be constrained, by configuration.

The security policy of a large enterprise has many elements and many points of enforcement. Elements of policy may be managed by the Information Systems department, by Human Resources, by the Legal department and by the Finance department. And the policy may be enforced by the extranet, mail, WAN and remote-access systems; platforms which inherently implement a permissive security policy. The current practice is to manage the configuration of each point of enforcement independently in order to implement the security policy as accurately as possible. Consequently, it is an expensive and unreliable proposition to modify the security policy. And, it is virtually impossible to obtain a consolidated view of the safeguards in effect throughout the enterprise to enforce the policy. At the same time, there is increasing pressure on corporate and government executives from consumers, shareholders and regulators to demonstrate "best practice" in the protection of the information assets of the enterprise and its customers.

For these reasons, there is a pressing need for a common language for expressing security policy. If implemented throughout an enterprise, a common policy language allows the enterprise to manage the enforcement of all the elements of its security policy in all the components of its information systems. Managing security policy may include some or all of the following steps: writing, reviewing, testing, approving, issuing, combining, analyzing, modifying, withdrawing, retrieving and enforcing policy.

XML is a natural choice as the basis for the common security-policy language, due to the ease with which its syntax and semantics can be extended to accommodate the unique requirements of this application, and the widespread support that it enjoys from all the main platform and tool vendors.

### 2.1. Requirements

The basic requirements of a policy language for expressing information system security policy are:

- To provide a method for combining individual **rules** and **policies** into a single **policy set** that applies to a particular **decision request**.
- To provide a method for flexible definition of the procedure by which **rules** and **policies** are combined.
- To provide a method for dealing with multiple **subjects** acting in different capacities.
- To provide a method for basing an **authorization decision** on **attributes** of the **subject** and **resource**.
- To provide a method for dealing with multi-valued **attributes**.
- To provide a method for basing an **authorization decision** on the contents of an information **resource**.
- To provide a set of logical and mathematical operators on **attributes** of the **subject**, **resource** and **environment**.

Deleted: cd-03

- 387 • To provide a method for handling a distributed set of **policy** components, while abstracting the  
388 method for locating, retrieving and authenticating the **policy** components.
  - 389 • To provide a method for rapidly identifying the **policy** that applies to a given action, based upon  
390 the values of **attributes** of the **subjects, resource** and **action**.
  - 391 • To provide an abstraction-layer that insulates the policy-writer from the details of the application  
392 environment.
  - 393 • To provide a method for specifying a set of actions that must be performed in conjunction with  
394 policy enforcement.
- 395 The motivation behind XACML is to express these well-established ideas in the field of access-  
396 control policy using an extension language of XML. The XACML solutions for each of these  
397 requirements are discussed in the following sections.

## 398 2.2. Rule and policy combining

399 The complete **policy** applicable to a particular **decision request** may be composed of a number of  
400 individual **rules** or **policies**. For instance, in a personal privacy application, the owner of the  
401 personal information may define certain aspects of disclosure **policy**, whereas the enterprise that is  
402 the custodian of the information may define certain other aspects. In order to render an  
403 **authorization decision**, it must be possible to combine the two separate **policies** to form the  
404 single **policy** applicable to the request.

405 XACML defines three top-level policy elements: `<Rule>`, `<Policy>` and `<PolicySet>`. The  
406 `<Rule>` element contains a Boolean expression that can be evaluated in isolation, but that is not  
407 intended to be accessed in isolation by a **PDP**. So, it is not intended to form the basis of an  
408 **authorization decision** by itself. It is intended to exist in isolation only within an XACML **PAP**,  
409 where it may form the basic unit of management, and be re-used in multiple **policies**.

410 The `<Policy>` element contains a set of `<Rule>` elements and a specified procedure for  
411 combining the results of their evaluation. It is the basic unit of **policy** used by the **PDP**, and so it is  
412 intended to form the basis of an **authorization decision**.

413 The `<PolicySet>` element contains a set of `<Policy>` or other `<PolicySet>` elements and a  
414 specified procedure for combining the results of their evaluation. It is the standard means for  
415 combining separate **policies** into a single combined **policy**.

416 Hinton et al [Hinton94] discuss the question of the compatibility of separate **policies** applicable to  
417 the same **decision request**.

## 418 2.3. Combining algorithms

419 XACML defines a number of combining algorithms that can be identified by a  
420 `RuleCombiningAlgId` or `PolicyCombiningAlgId` attribute of the `<Policy>` or `<PolicySet>`  
421 elements, respectively. The **rule-combining algorithm** defines a procedure for arriving at an  
422 **authorization decision** given the individual results of evaluation of a set of **rules**. Similarly, the  
423 **policy-combining algorithm** defines a procedure for arriving at an **authorization decision** given  
424 the individual results of evaluation of a set of **policies**. Standard combining algorithms are defined  
425 for:

- 426 • Deny-overrides (Ordered and Unordered),
- 427 • Permit-overrides (Ordered and Unordered),

access\_control-xacml-2.0-core-spec-[cd-04](#)

Deleted: cd-03

- First-applicable and
- Only-one-applicable.

In the case of the Deny-overrides algorithm, if a single <Rule> or <Policy> element is encountered that evaluates to "Deny", then, regardless of the evaluation result of the other <Rule> or <Policy> elements in the **applicable policy**, the combined result is "Deny".

Likewise, in the case of the Permit-overrides algorithm, if a single "Permit" result is encountered, then the combined result is "Permit".

In the case of the "First-applicable" combining algorithm, the combined result is the same as the result of evaluating the first <Rule>, <Policy> or <PolicySet> element in the list of **rules** whose **target** is applicable to the **decision request**.

The "Only-one-applicable" **policy-combining algorithm** only applies to **policies**. The result of this combining algorithm ensures that one and only one **policy** or **policy set** is applicable by virtue of their **targets**. If no **policy** or **policy set** applies, then the result is "NotApplicable", but if more than one **policy** or **policy set** is applicable, then the result is "Indeterminate". When exactly one **policy** or **policy set** is applicable, the result of the combining algorithm is the result of evaluating the single **applicable policy** or **policy set**.

**Policies** and **policy sets** may take parameters that modify the behaviour of the **combining algorithms**. However, none of the standard **combining algorithms** is affected by parameters.

Users of this specification may, if necessary, define their own combining algorithms.

## 2.4. Multiple subjects

Access-control policies often place requirements on the actions of more than one **subject**. For instance, the policy governing the execution of a high-value financial transaction may require the approval of more than one individual, acting in different capacities. Therefore, XACML recognizes that there may be more than one **subject** relevant to a **decision request**. An **attribute** called "subject-category" is used to differentiate between **subjects** acting in different capacities. Some standard values for this **attribute** are specified, and users may define additional ones.

## 2.5. Policies based on subject and resource attributes

Another common requirement is to base an **authorization decision** on some characteristic of the **subject** other than its identity. Perhaps, the most common application of this idea is the **subject's** role [RBAC]. XACML provides facilities to support this approach. **Attributes** of **subjects** contained in the request **context** may be identified by the <SubjectAttributeDesignator> element. This element contains a URN that identifies the **attribute**. Alternatively, the <AttributeSelector> element may contain an XPath expression over the request **context** to identify a particular **subject attribute** value by its location in the **context** (see Section 2.11 for an explanation of **context**).

XACML provides a standard way to reference the **attributes** defined in the LDAP series of specifications [LDAP-1, LDAP-2]. This is intended to encourage implementers to use standard **attribute** identifiers for some common **subject attributes**.

Another common requirement is to base an **authorization decision** on some characteristic of the **resource** other than its identity. XACML provides facilities to support this approach. **Attributes** of the **resource** may be identified by the <ResourceAttributeDesignator> element. This element contains a URN that identifies the **attribute**. Alternatively, the <AttributeSelector>

Deleted: cd-03

470 element may contain an XPath expression over the request **context** to identify a particular  
471 **resource attribute** value by its location in the **context**.

## 472 2.6. Multi-valued attributes

473 The most common techniques for communicating **attributes** (LDAP, XPath, SAML, etc.) support  
474 multiple values per **attribute**. Therefore, when an XACML **PDP** retrieves the value of a **named**  
475 **attribute**, the result may contain multiple values. A collection of such values is called a **bag**. A  
476 **bag** differs from a set in that it may contain duplicate values, whereas a set may not. Sometimes  
477 this situation represents an error. Sometimes the XACML **rule** is satisfied if any one of the  
478 **attribute** values meets the criteria expressed in the **rule**.

479 XACML provides a set of functions that allow a policy writer to be absolutely clear about how the  
480 **PDP** should handle the case of multiple **attribute** values. These are the “higher-order” functions  
481 (see Section A.3).

## 482 2.7. Policies based on resource contents

483 In many applications, it is required to base an **authorization decision** on data *contained in* the  
484 information **resource** to which **access** is requested. For instance, a common component of privacy  
485 **policy** is that a person should be allowed to read records for which he or she is the subject. The  
486 corresponding **policy** must contain a reference to the **subject** identified in the information **resource**  
487 itself.

488 XACML provides facilities for doing this when the information **resource** can be represented as an  
489 XML document. The <AttributeSelector> element may contain an XPath expression over the  
490 request **context** to identify data in the information **resource** to be used in the **policy** evaluation.

491 In cases where the information **resource** is not an XML document, specified **attributes** of the  
492 **resource** can be referenced, as described in Section 2.4.

## 493 2.8. Operators

494 Information security **policies** operate upon **attributes** of **subjects**, the **resource**, the **action** and  
495 the **environment** in order to arrive at an **authorization decision**. In the process of arriving at the  
496 **authorization decision**, **attributes** of many different types may have to be compared or computed.  
497 For instance, in a financial application, a person's available credit may have to be calculated by  
498 adding their credit limit to their account balance. The result may then have to be compared with the  
499 transaction value. This sort of situation gives rise to the need for arithmetic operations on  
500 **attributes** of the **subject** (account balance and credit limit) and the **resource** (transaction value).

501 Even more commonly, a **policy** may identify the set of roles that are permitted to perform a  
502 particular action. The corresponding operation involves checking whether there is a non-empty  
503 intersection between the set of roles occupied by the **subject** and the set of roles identified in the  
504 **policy**. Hence the need for set operations.

505 XACML includes a number of built-in functions and a method of adding non-standard functions.  
506 These functions may be nested to build arbitrarily complex expressions. This is achieved with the  
507 <Apply> element. The <Apply> element has an XML attribute called `FunctionId` that identifies  
508 the function to be applied to the contents of the element. Each standard function is defined for  
509 specific argument data-type combinations, and its return data-type is also specified. Therefore,  
510 data-type consistency of the **policy** can be checked at the time the **policy** is written or parsed.  
511 And, the types of the data values presented in the request **context** can be checked against the  
512 values expected by the **policy** to ensure a predictable outcome.

Deleted: cd-03



513 In addition to operators on numerical and set arguments, operators are defined for date, time and  
514 duration arguments.

515 Relationship operators (equality and comparison) are also defined for a number of data-types,  
516 including the RFC822 and X.500 name-forms, strings, URIs, etc..

517 Also noteworthy are the operators over Boolean data-types, which permit the logical combination of  
518 **predicates** in a **rule**. For example, a **rule** may contain the statement that **access** may be  
519 permitted during business hours AND from a terminal on business premises.

520 The XACML method of representing functions borrows from MathML [MathML] and from the  
521 XQuery 1.0 and XPath 2.0 Functions and Operators specification [XF].

## 522 2.9. Policy distribution

523 In a distributed system, individual **policy** statements may be written by several policy writers and  
524 enforced at several enforcement points. In addition to facilitating the collection and combination of  
525 independent **policy** components, this approach allows **policies** to be updated as required. XACML  
526 **policy** statements may be distributed in any one of a number of ways. But, XACML does not  
527 describe any normative way to do this. Regardless of the means of distribution, **PDPs** are  
528 expected to confirm, by examining the **policy's** <Target> element that the policy is applicable to  
529 the **decision request** that it is processing.

530 <Policy> elements may be attached to the information **resources** to which they apply, as  
531 described by Perritt [Perritt93]. Alternatively, <Policy> elements may be maintained in one or  
532 more locations from which they are retrieved for evaluation. In such cases, the **applicable policy**  
533 may be referenced by an identifier or locator closely associated with the information **resource**.

## 534 2.10. Policy indexing

535 For efficiency of evaluation and ease of management, the overall security policy in force across an  
536 enterprise may be expressed as multiple independent **policy** components. In this case, it is  
537 necessary to identify and retrieve the **applicable policy** statement and verify that it is the correct  
538 one for the requested action before evaluating it. This is the purpose of the <Target> element in  
539 XACML.

540 Two approaches are supported:

- 541 1. **Policy** statements may be stored in a database,. In this case, the **PDP** should form a database  
542 query to retrieve just those **policies** that are applicable to the set of **decision requests** to  
543 which it expects to respond. Additionally, the **PDP** should evaluate the <Target> element of  
544 the retrieved **policy** or **policy set** statements as defined by the XACML specification.
- 545 2. Alternatively, the **PDP** may be loaded with all available policies and evaluate their <Target>  
546 elements in the context of a particular **decision request**, in order to identify the **policies** and  
547 **policy sets** that are applicable to that request.

548 The use of constraints limiting the applicability of a **policy** were described by Sloman [Sloman94].

## 549 2.11. Abstraction layer

550 **PEPs** come in many forms. For instance, a **PEP** may be part of a remote-access gateway, part of  
551 a Web server or part of an email user-agent, etc.. It is unrealistic to expect that all **PEPs** in an  
552 enterprise do currently, or will in the future, issue **decision requests** to a **PDP** in a common format.  
553 Nevertheless, a particular **policy** may have to be enforced by multiple **PEPs**. It would be inefficient

Deleted: cd-03



554 to force a policy writer to write the same **policy** several different ways in order to accommodate the  
555 format requirements of each **PEP**. Similarly attributes may be contained in various envelope types  
556 (e.g. X.509 attribute certificates, SAML attribute assertions, etc.). Therefore, there is a need for a  
557 canonical form of the request and response handled by an XACML **PDP**. This canonical form is  
558 called the XACML **context**. Its syntax is defined in XML schema.

559 Naturally, XACML-conformant **PEPs** may issue requests and receive responses in the form of an  
560 XACML **context**. But, where this situation does not exist, an intermediate step is required to  
561 convert between the request/response format understood by the **PEP** and the XACML **context**  
562 format understood by the **PDP**.

563 The benefit of this approach is that **policies** may be written and analyzed independent of the  
564 specific environment in which they are to be enforced.

565 In the case where the native request/response format is specified in XML Schema (e.g. a SAML-  
566 conformant **PEP**), the transformation between the native format and the XACML **context** may be  
567 specified in the form of an Extensible Stylesheet Language Transformation [XSLT].

568 Similarly, in the case where the **resource** to which **access** is requested is an XML document, the  
569 **resource** itself may be included in, or referenced by, the request **context**. Then, through the use  
570 of XPath expressions [XPath] in the **policy**, values in the **resource** may be included in the **policy**  
571 evaluation.

## 572 2.12. Actions performed in conjunction with enforcement

573 In many applications, policies specify actions that **MUST** be performed, either instead of, or in  
574 addition to, actions that **MAY** be performed. This idea was described by Sloman [Sloman94].  
575 XACML provides facilities to specify actions that **MUST** be performed in conjunction with policy  
576 evaluation in the <Obligations> element. This idea was described as a provisional action by  
577 Kudo [Kudo00]. There are no standard definitions for these actions in version 2.0 of XACML.  
578 Therefore, bilateral agreement between a **PAP** and the **PEP** that will enforce its **policies** is required  
579 for correct interpretation. **PEPs** that conform with v2.0 of XACML are required to deny **access**  
580 unless they understand and can discharge all of the <Obligations> elements associated with the  
581 **applicable policy**. <Obligations> elements are returned to the **PEP** for enforcement.

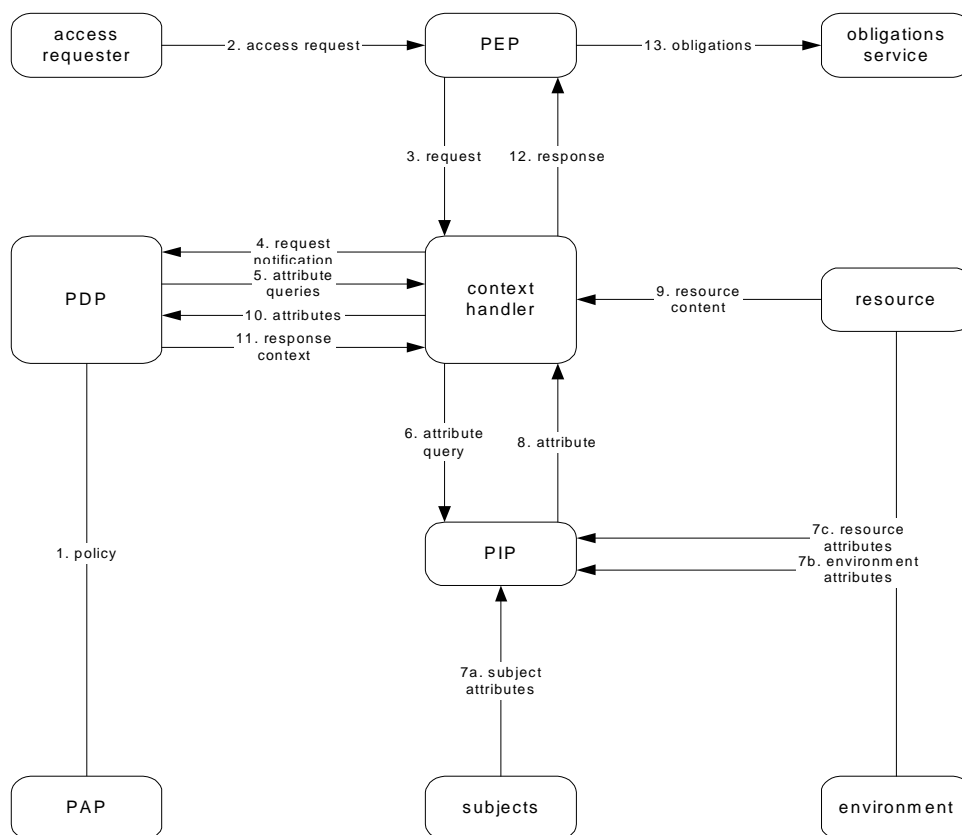
---

## 582 3. Models (non-normative)

583 The data-flow model and language model of XACML are described in the following sub-sections.

### 584 3.1. Data-flow model

585 The major actors in the XACML domain are shown in the data-flow diagram of Figure 1.



**Figure 1 - Data-flow diagram**

Note: some of the data-flows shown in the diagram may be facilitated by a repository. For instance, the communications between the **context handler** and the **PIP** or the communications between the **PDP** and the **PAP** may be facilitated by a repository. The XACML specification is not intended to place restrictions on the location of any such repository, or indeed to prescribe a particular communication protocol for any of the data-flows.

The model operates by the following steps.

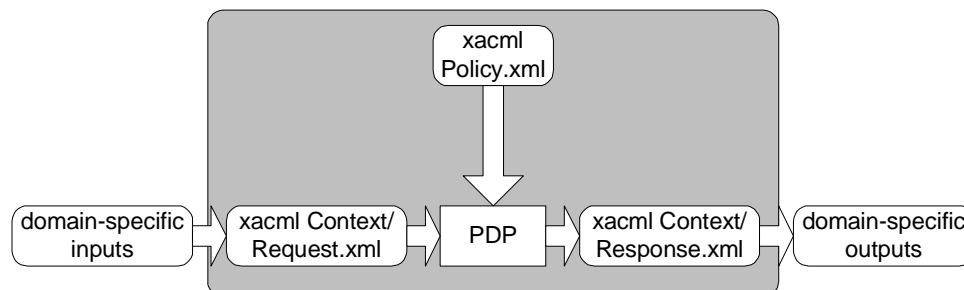
1. **PAPs** write **policies** and **policy sets** and make them available to the **PDP**. These **policies** or **policy sets** represent the complete policy for a specified **target**.
2. The access requester sends a request for access to the **PEP**.
3. The **PEP** sends the request for **access** to the **context handler** in its native request format, optionally including **attributes** of the **subjects**, **resource**, **action** and **environment**.
4. The **context handler** constructs an XACML request **context** and sends it to the **PDP**.
5. The **PDP** requests any additional **subject**, **resource**, **action** and **environment attributes** from the **context handler**.
6. The context handler requests the attributes from a **PIP**.

Deleted: cd-03

- 603 7. The **PIP** obtains the requested **attributes**.
- 604 8. The **PIP** returns the requested **attributes** to the **context handler**.
- 605 9. Optionally, the **context handler** includes the **resource** in the **context**.
- 606 10. The **context handler** sends the requested **attributes** and (optionally) the **resource** to the **PDP**.
- 607 The **PDP** evaluates the **policy**.
- 608 11. The **PDP** returns the response **context** (including the **authorization decision**) to the **context handler**.
- 609
- 610 12. The **context handler** translates the response **context** to the native response format of the
- 611 **PEP**. The **context handler** returns the response to the **PEP**.
- 612 13. The **PEP** fulfills the **obligations**.
- 613 14. (Not shown) If **access** is permitted, then the **PEP** permits **access** to the **resource**; otherwise, it
- 614 denies **access**.

### 615 3.2. XACML context

616 XACML is intended to be suitable for a variety of application environments. The core language is  
 617 insulated from the application environment by the XACML **context**, as shown in Figure 2, in which  
 618 the scope of the XACML specification is indicated by the shaded area. The XACML **context** is  
 619 defined in XML schema, describing a canonical representation for the inputs and outputs of the  
 620 **PDP**. **Attributes** referenced by an instance of XACML policy may be in the form of XPath  
 621 expressions over the **context**, or attribute designators that identify the **attribute** by **subject**,  
 622 **resource**, **action** or **environment** and its identifier, data-type and (optionally) its issuer.  
 623 Implementations must convert between the **attribute** representations in the application environment  
 624 (e.g., SAML, J2SE, CORBA, and so on) and the **attribute** representations in the XACML **context**.  
 625 How this is achieved is outside the scope of the XACML specification. In some cases, such as  
 626 SAML, this conversion may be accomplished in an automated way through the use of an XSLT  
 627 transformation.



628  
 629 **Figure 2 - XACML context**

630 Note: The **PDP** is not required to operate directly on the XACML representation of a policy. It may  
 631 operate directly on an alternative representation.

632 See Section 7.2.5 for a more detailed discussion of the request **context**.

### 633 3.3. Policy language model

634 The policy language model is shown in Figure 3. The main components of the model are:

access\_control-xacml-2.0-core-spec-[cd-04](#)

Deleted: cd-03

635 • **Rule**;  
636 • **Policy**; and  
637 • **Policy set**.  
638 These are described in the following sub-sections.

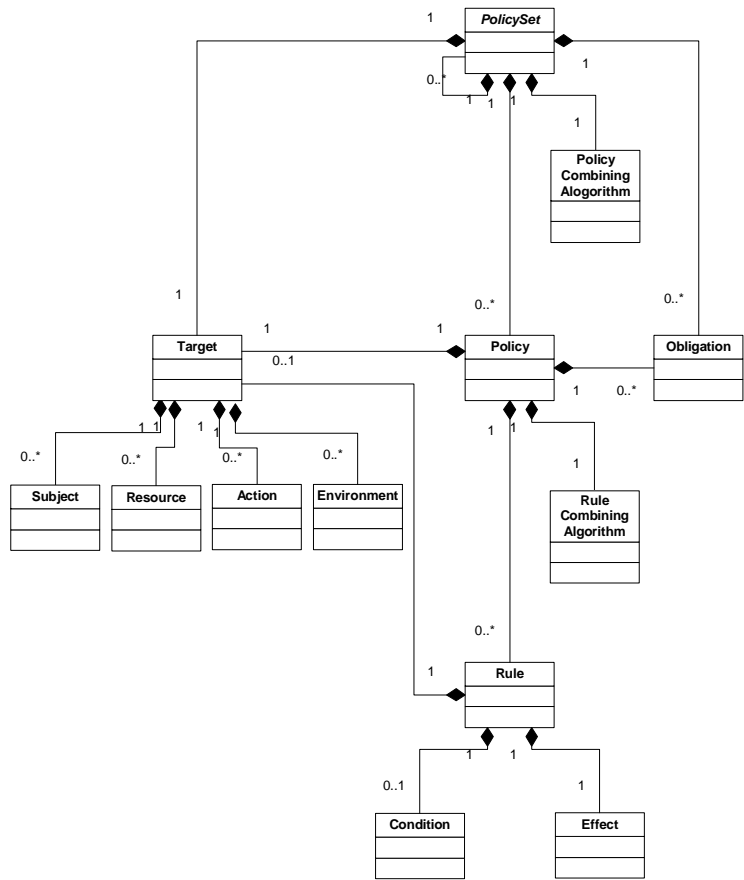


Figure 3 - Policy language model

### 3.3.1 Rule

642 A **rule** is the most elementary unit of **policy**. It may exist in isolation only *within* one of the major  
643 actors of the XACML domain. In order to exchange **rules** between major actors, they must be  
644 encapsulated in a **policy**. A **rule** can be evaluated on the basis of its contents. The main  
645 components of a **rule** are:

- 646 • a **target**;
- 647 • an **effect** and

648 • a **condition**.

649 These are discussed in the following sub-sections.

### 650 3.3.1.1. Rule target

651 The **target** defines the set of:

- 652 • **resources**;
- 653 • **subjects**;
- 654 • **actions** and
- 655 • **environment**

656 to which the **rule** is intended to apply. The `<Condition>` element may further refine the  
657 applicability established by the **target**. If the **rule** is intended to apply to all entities of a particular  
658 data-type, then the corresponding entity is omitted from the **target**. An XACML **PDP** verifies that  
659 the matches defined by the **target** are satisfied by the **subjects**, **resource**, **action** and  
660 **environment attributes** in the request **context**. **Target** definitions are discrete, in order that  
661 applicable **rules** may be efficiently identified by the **PDP**.

662 The `<Target>` element may be absent from a `<Rule>`. In this case, the **target** of the `<Rule>` is  
663 the same as that of the parent `<Policy>` element.

664 Certain **subject** name-forms, **resource** name-forms and certain types of **resource** are internally  
665 structured. For instance, the X.500 directory name-form and RFC 822 name-form are structured  
666 **subject** name-forms, whereas an account number commonly has no discernible structure. UNIX  
667 file-system path-names and URIs are examples of structured **resource** name-forms. And an XML  
668 document is an example of a structured **resource**.

669 Generally, the name of a node (other than a leaf node) in a structured name-form is also a legal  
670 instance of the name-form. So, for instance, the RFC822 name "med.example.com" is a legal  
671 RFC822 name identifying the set of mail addresses hosted by the med.example.com mail server.  
672 And the XPath/XPointer value `/xacml-context:Request/xacml-context:Resource/xacml-`  
673 `context:ResourceContent/md:record/md:patient/` is a legal XPath/XPointer value identifying a  
674 node-set in an XML document.

675 The question arises: how should a name that identifies a set of **subjects** or **resources** be  
676 interpreted by the **PDP**, whether it appears in a **policy** or a request **context**? Are they intended to  
677 represent just the node explicitly identified by the name, or are they intended to represent the entire  
678 sub-tree subordinate to that node?

679 In the case of **subjects**, there is no real entity that corresponds to such a node. So, names of this  
680 type always refer to the set of **subjects** subordinate in the name structure to the identified node.  
681 Consequently, non-leaf **subject** names should not be used in equality functions, only in match  
682 functions, such as "urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match" not  
683 "urn:oasis:names:tc:xacml:1.0:function:rfc822Name-equal" (see Appendix A).

### 684 3.3.1.2. Effect

685 The **effect** of the **rule** indicates the rule-writer's intended consequence of a "True" evaluation for  
686 the **rule**. Two values are allowed: "Permit" and "Deny".

Deleted: cd-03

### 3.3.1.3. Condition

**Condition** represents a Boolean expression that refines the applicability of the **rule** beyond the **predicates** implied by its **target**. Therefore, it may be absent.

### 3.3.2 Policy

From the data-flow model one can see that **rules** are not exchanged amongst system entities. Therefore, a **PAP** combines **rules** in a **policy**. A **policy** comprises four main components:

- a **target**;
- a **rule-combining algorithm**-identifier;
- a set of **rules**; and
- **obligations**.

**Rules** are described above. The remaining components are described in the following sub-sections.

#### 3.3.2.1. Policy target

An XACML <PolicySet>, <Policy> or <Rule> element contains a <Target> element that specifies the set of **subjects**, **resources**, **actions** and **environments** to which it applies. The <Target> of a <PolicySet> or <Policy> may be declared by the writer of the <PolicySet> or <Policy>, or it may be calculated from the <Target> elements of the <PolicySet>, <Policy> and <Rule> elements that it contains.

A system entity that calculates a <Target> in this way is not defined by XACML, but there are two logical methods that might be used. In one method, the <Target> element of the outer <PolicySet> or <Policy> (the "outer component") is calculated as the *union* of all the <Target> elements of the referenced <PolicySet>, <Policy> or <Rule> elements (the "inner components"). In another method, the <Target> element of the outer component is calculated as the *intersection* of all the <Target> elements of the inner components. The results of evaluation in each case will be very different: in the first case, the <Target> element of the outer component makes it applicable to any **decision request** that matches the <Target> element of at least one inner component; in the second case, the <Target> element of the outer component makes it applicable only to **decision requests** that match the <Target> elements of every inner component. Note that computing the intersection of a set of <Target> elements is likely only practical if the target data-model is relatively simple.

In cases where the <Target> of a <Policy> is *declared* by the **policy** writer, any component <Rule> elements in the <Policy> that have the same <Target> element as the <Policy> element may omit the <Target> element. Such <Rule> elements inherit the <Target> of the <Policy> in which they are contained.

#### 3.3.2.2. Rule-combining algorithm

The **rule-combining algorithm** specifies the procedure by which the results of evaluating the component **rules** are combined when evaluating the **policy**, i.e. the **Decision** value placed in the response **context** by the **PDP** is the value of the **policy**, as defined by the **rule-combining algorithm**. A **policy** may have combining parameters that affect the operation of the **rule-combining algorithm**.

Deleted: cd-03

727 See Appendix C for definitions of the normative *rule-combining algorithms*.

### 728 3.3.2.3. Obligations

729 *Obligations* may be added by the writer of the *policy*.

730 When a *PDP* evaluates a *policy* containing *obligations*, it returns certain of those *obligations* to  
731 the *PEP* in the response *context*. Section 7.14 explains which *obligations* are to be returned.

### 732 3.3.3 Policy set

733 A *policy set* comprises four main components:

- 734 • a *target*;
- 735 • a *policy-combining algorithm*-identifier
- 736 • a set of *policies*; and
- 737 • *obligations*.

738 The *target* and *policy* components are described above. The other components are described in  
739 the following sub-sections.

#### 740 3.3.3.1. Policy-combining algorithm

741 The *policy-combining algorithm* specifies the procedure by which the results of evaluating the  
742 component *policies* are combined when evaluating the *policy set*, i.e. the *Decision* value placed  
743 in the response *context* by the *PDP* is the result of evaluating the *policy set*, as defined by the  
744 *policy-combining algorithm*. A *policy set* may have combining parameters that affect the  
745 operation of the *policy-combining algorithm*.

746 See Appendix C for definitions of the normative *policy-combining algorithms*.

#### 747 3.3.3.2. Obligations

748 The writer of a *policy set* may add *obligations* to the *policy set*, in addition to those contained in  
749 the component *policies* and *policy sets*.

750 When a *PDP* evaluates a *policy set* containing *obligations*, it returns certain of those *obligations*  
751 to the *PEP* in its response *context*. Section 7.14 explains which *obligations* are to be returned.

---

## 752 4. Examples (non-normative)

753 This section contains two examples of the use of XACML for illustrative purposes. The first example  
754 is a relatively simple one to illustrate the use of *target*, *context*, matching functions and *subject*  
755 *attributes*. The second example additionally illustrates the use of the *rule-combining algorithm*,  
756 *conditions* and *obligations*.

757

## 4.1. Example one

758

### 4.1.1 Example policy

759

Assume that a corporation named Medi Corp (identified by its domain name: med.example.com) has an **access control policy** that states, in English:

760

761

Any user with an e-mail name in the "med.example.com" namespace is allowed to perform any **action** on any **resource**.

762

763

An XACML **policy** consists of header information, an optional text description of the policy, a **target**, one or more **rules** and an optional set of **obligations**.

764

765

```
[a02] <?xml version="1.0" encoding="UTF-8"?>
```

766

```
[a03] <Policy
```

767

```
[a04] xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:cd:04"
```

768

```
[a05] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

769

```
[a06] xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:cd:04
```

770

```
http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-cd-
```

771

```
04.xsd"
```

772

```
[a07] PolicyId="urn:oasis:names:tc:example:SimplePolicy1"
```

773

```
[a08] RuleCombiningAlgId="identifier:rule-combining-algorithm:deny-overrides">
```

774

```
[a09] <Description>
```

775

```
[a10] Medi Corp access control policy
```

776

```
[a11] </Description>
```

777

```
[a12] <Target/>
```

778

```
[a13] <Rule
```

779

```
[a14] RuleId="urn:oasis:names:tc:xacml:2.0:example:SimpleRule1"
```

780

```
[a15] Effect="Permit">
```

781

```
[a16] <Description>
```

782

```
[a17] Any subject with an e-mail name in the med.example.com domain
```

783

```
[a18] can perform any action on any resource.
```

784

```
[a19] </Description>
```

785

```
[a20] <Target>
```

786

```
[a21] <Subjects>
```

787

```
[a22] <Subject>
```

788

```
[a23] <SubjectMatch
```

789

```
[a24] MatchId="urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match">
```

790

```
[a25] <AttributeValue
```

791

```
[a26] DataType="http://www.w3.org/2001/XMLSchema#string">
```

792

```
[a27] med.example.com
```

793

```
[a28] </AttributeValue>
```

794

```
[a29] <SubjectAttributeDesignator
```

795

```
[a30] AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
```

796

```
[a31] DataType="urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"/>
```

797

```
[a32] </SubjectMatch>
```

798

```
[a33] </Subject>
```

799

```
[a34] </Subjects>
```

800

```
[a35] </Target>
```

801

```
[a36] </Rule>
```

802

```
[a37] </Policy>
```

803

[a02] is a standard XML document tag indicating which version of XML is being used and what the

804

character encoding is.

805

[a03] introduces the XACML Policy itself.

806

[a04] - [a05] are XML namespace declarations.

807

[a04] gives a URN for the XACML **policies** schema.

Deleted: cd:03

Deleted: cd:03

Deleted: cd:03

Deleted: cd:03

access\_control-xacml-2.0-core-spec-~~cd-04~~

24



808 [a07] assigns a name to this **policy** instance. The name of a **policy** has to be unique for a given  
809 **PDP** so that there is no ambiguity if one **policy** is referenced from another **policy**. The **version**  
810 attribute is omitted, so it takes its default value of "1.0".

811 [a08] specifies the algorithm that will be used to resolve the results of the various **rules** that may be  
812 in the **policy**. The **deny-overrides rule-combining algorithm** specified here says that, if any **rule**  
813 evaluates to "Deny", then the **policy** must return "Deny". If all **rules** evaluate to "Permit", then the  
814 **policy** must return "Permit". The **rule-combining algorithm**, which is fully described in Appendix  
815 C, also says what to do if an error were to occur when evaluating any **rule**, and what to do with  
816 **rules** that do not apply to a particular **decision request**.

817 [a09] - [a11] provide a text description of the policy. This description is optional.

818 [a12] describes the **decision requests** to which this **policy** applies. If the **subject, resource,**  
819 **action** and **environment** in a **decision request** do not match the values specified in the **policy**  
820 **target**, then the remainder of the **policy** does not need to be evaluated. This **target** section is  
821 useful for creating an index to a set of **policies**. In this simple example, the **target** section says the  
822 **policy** is applicable to any **decision request**.

823 [a13] introduces the one and only **rule** in this simple **policy**.

824 [a14] specifies the identifier for this **rule**. Just as for a **policy**, each **rule** must have a unique  
825 identifier (at least unique for any **PDP** that will be using the **policy**).

826 [a15] says what **effect** this **rule** has if the **rule** evaluates to "True". **Rules** can have an **effect** of  
827 either "Permit" or "Deny". In this case, if the **rule** is satisfied, it will evaluate to "Permit", meaning  
828 that, as far as this one **rule** is concerned, the requested **access** should be permitted. If a **rule**  
829 evaluates to "False", then it returns a result of "NotApplicable". If an error occurs when evaluating  
830 the **rule**, then the **rule** returns a result of "Indeterminate". As mentioned above, the **rule-**  
831 **combining algorithm** for the **policy** specifies how various **rule** values are combined into a single  
832 **policy** value.

833 [a16] - [a19] provide a text description of this **rule**. This description is optional.

834 [a20] introduces the **target** of the **rule**. As described above for the **target** of a policy, the **target** of  
835 a **rule** describes the **decision requests** to which this **rule** applies. If the **subject, resource,**  
836 **action** and **environment** in a **decision request** do not match the values specified in the **rule**  
837 **target**, then the remainder of the **rule** does not need to be evaluated, and a value of  
838 "NotApplicable" is returned to the **rule** evaluation.

839 The **rule target** is similar to the **target** of the **policy** itself, but with one important difference. [a23]-  
840 [a32] spells out a specific value that the **subject** in the **decision request** must match. The  
841 <SubjectMatch> element specifies a matching function in the **MatchId** attribute, a literal value of  
842 "med.example.com" and a pointer to a specific **subject attribute** in the request **context** by means  
843 of the <SubjectAttributeDesignator> element. The matching function will be used to  
844 compare the literal value with the value of the **subject attribute**. Only if the match returns "True"  
845 will this **rule** apply to a particular **decision request**. If the match returns "False", then this **rule** will  
846 return a value of "NotApplicable".

847 [a36] closes the **rule**. In this **rule**, all the **work** is done in the <Target> element. In more complex  
848 **rules**, the <Target> may have been followed by a <Condition> element (which could also be a  
849 set of **conditions** to be **ANDed** or **ORed** together).

850 [a37] closes the **policy**. As mentioned above, this **policy** has only one **rule**, but more complex  
851 **policies** may have any number of **rules**.

Deleted: cd-03

## 4.1.2 Example request context

Let's examine a hypothetical **decision request** that might be submitted to a **PDP** that executes the **policy** above. In English, the **access** request that generates the **decision request** may be stated as follows:

Bart Simpson, with e-mail name "bs@simpsons.com", wants to read his medical record at Medi Corp.

In XACML, the information in the **decision request** is formatted into a **request context** statement that looks as follows:

```
[a38] <?xml version="1.0" encoding="UTF-8"?>
[a39] <Request xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:cd:04"
[a40] xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:context:schema:cd:04
http://docs.oasis-open.org/xacml/access_control-xacml-2.0-context-schema-
04.xsd">
[a41]   <Subject>
[a42]     <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
[a43]       <AttributeValue>
[a44]         bs@simpsons.com
[a45]       </AttributeValue>
[a46]     </Attribute>
[a47]   </Subject>
[a48]   <Resource>
[a49]     <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-
id" DataType="http://www.w3.org/2001/XMLSchema#anyURI">
[a50]       <AttributeValue>
[a51]         file://example/med/record/patient/BartSimpson
[a52]       </AttributeValue>
[a53]     </Attribute>
[a54]   </Resource>
[a55]   <Action>
[a56]     <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
[a57]       <AttributeValue>
[a58]         read
[a59]       </AttributeValue>
[a60]     </Attribute>
[a61]   </Action>
[a62] </Environment/>
[a63] </Request>
```

Deleted: cd:03

Deleted: cd:03

Deleted: cd:03

[a38] - [a40] contain the header information for the **request context**, and are used the same way as the header for the **policy** explained above.

The **<Subject>** element contains one or more **attributes** of the entity making the **access** request. There can be multiple **subjects**, and each **subject** can have multiple **attributes**. In this case, in [a41] - [a47], there is only one **subject**, and the **subject** has only one **attribute**: the **subject's** identity, expressed as an e-mail name, is "bs@simpsons.com". In this example, the **subject-category** attribute is omitted. Therefore, it adopts its default value of "access-subject".

The **<Resource>** element contains one or more **attributes** of the **resource** to which the **subject** (or **subjects**) has requested **access**. There can be only one **<Resource>** per **decision request**<sup>1</sup>. Lines [a48] - [a54] contain the one **attribute** of the **resource** to which Bart Simpson has requested

<sup>1</sup> Some exceptions are described in the XACML Profile for Multiple Resources [MULT].

Deleted: cd:03

902 **access**: the **resource** identified by its file URI, which is  
 903 "file:///medico/record/patient/BartSimpson".

904 The <Action> element contains one or more **attributes** of the **action** that the **subject** (or  
 905 **subjects**) wishes to take on the **resource**. There can be only one **action** per **decision request**.  
 906 [a55] - [a61] describe the identity of the **action** Bart Simpson wishes to take, which is "read".

907 The <Environment> element, [a62], is empty.

908 [a63] closes the **request context**. A more complex **request context** may have contained some  
 909 **attributes** not associated with the **subject**, the **resource** or the **action**. These would have been  
 910 placed in an optional <Environment> element following the <Action> element.

911 The **PDP** processing this request **context** locates the **policy** in its policy repository. It compares  
 912 the **subject**, **resource**, **action** and **environment** in the request **context** with the **subjects**,  
 913 **resources**, **actions** and **environments** in the **policy target**. Since the **policy target** is empty, the  
 914 **policy** matches this **context**.

915 The **PDP** now compares the **subject**, **resource**, **action** and **environment** in the request **context**  
 916 with the **target** of the one **rule** in this **policy**. The requested **resource** matches the <Target>  
 917 element and the requested **action** matches the <Target> element, but the requesting subject-id  
 918 **attribute** does not match "med.example.com".

### 4.1.3 Example response context

920 As a result of evaluating the policy, there is no **rule** in this **policy** that returns a "Permit" result for  
 921 this request. The **rule-combining algorithm** for the **policy** specifies that, in this case, a result of  
 922 "NotApplicable" should be returned. The response **context** looks as follows:

```
923 [a64] <?xml version="1.0" encoding="UTF-8"?>
924 [a65] <Response xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:cd:04"
925      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
926      xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:context:schema:cd:04
927      http://docs.oasis-open.org/xacml/xacml-core-2.0-context-schema-cd-04.xsd">
928 [a66]   <Result>
929 [a67]     <Decision>NotApplicable</Decision>
930 [a68]   </Result>
931 [a69] </Response>
```

Deleted: cd:03

Deleted: cd:03

Deleted: cd-03

932 [a64] - [a65] contain the same sort of header information for the response as was described above  
 933 for a **policy**.

934 The <Result> element in lines [a66] - [a68] contains the result of evaluating the **decision request**  
 935 against the **policy**. In this case, the result is "NotApplicable". A **policy** can return "Permit", "Deny",  
 936 "NotApplicable" or "Indeterminate". Therefore, the **PEP** is required to deny **access**.

937 [a69] closes the response **context**.

## 4.2. Example two

939 This section contains an example XML document, an example request **context** and example  
 940 XACML **rules**. The XML document is a medical record. Four separate **rules** are defined. These  
 941 illustrate a **rule-combining algorithm**, **conditions** and **obligations**.

Deleted: cd-03

access\_control-xacml-2.0-core-spec-cd-04

## 4.2.1 Example medical record instance

The following is an instance of a medical record to which the example XACML *rules* can be applied. The <record> schema is defined in the registered namespace administered by Medi Corp.

```

946 [a70] <?xml version="1.0" encoding="UTF-8"?>
947 [a71] <record xmlns="urn:example:med:schemas:record"
948 [a72] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
949 [a73]   <patient>
950 [a74]     <patientName>
951 [a75]       <first>Bartholomew</first>
952 [a76]       <last>Simpson</last>
953 [a77]     </patientName>
954 [a78]     <patientContact>
955 [a79]       <street>27 Shelbyville Road</street>
956 [a80]       <city>Springfield</city>
957 [a81]       <state>MA</state>
958 [a82]       <zip>12345</zip>
959 [a83]       <phone>555.123.4567</phone>
960 [a84]       <fax/>
961 [a85]       <email/>
962 [a86]     </patientContact>
963 [a87]     <patientDoB>1992-03-21</patientDoB>
964 [a88]     <patientGender>male</patientGender>
965 [a89]     <patient-number>555555</patient-number>
966 [a90]   </patient>
967 [a91]   <parentGuardian>
968 [a92]     <parentGuardianId>HS001</parentGuardianId>
969 [a93]     <parentGuardianName>
970 [a94]       <first>Homer</first>
971 [a95]       <last>Simpson</last>
972 [a96]     </parentGuardianName>
973 [a97]     <parentGuardianContact>
974 [a98]       <street>27 Shelbyville Road</street>
975 [a99]       <city>Springfield</city>
976 [a100]      <state>MA</state>
977 [a101]      <zip>12345</zip>
978 [a102]      <phone>555.123.4567</phone>
979 [a103]      <fax/>
980 [a104]      <email>homers@aol.com</email>
981 [a105]    </parentGuardianContact>
982 [a106]   </parentGuardian>
983 [a107]   <primaryCarePhysician>
984 [a108]     <physicianName>
985 [a109]       <first>Julius</first>
986 [a110]       <last>Hibbert</last>
987 [a111]     </physicianName>
988 [a112]     <physicianContact>
989 [a113]       <street>1 First St</street>
990 [a114]       <city>Springfield</city>
991 [a115]       <state>MA</state>
992 [a116]       <zip>12345</zip>
993 [a117]       <phone>555.123.9012</phone>
994 [a118]       <fax>555.123.9013</fax>
995 [a119]       <email/>
996 [a120]     </physicianContact>
997 [a121]     <registrationID>ABC123</registrationID>
998 [a122]   </primaryCarePhysician>
999 [a123]   <insurer>
1000 [a124]     <name>Blue Cross</name>
1001 [a125]     <street>1234 Main St</street>
1002 [a126]     <city>Springfield</city>

```

Deleted: cd-03

```

1003 [a127]      <state>MA</state>
1004 [a128]      <zip>12345</zip>
1005 [a129]      <phone>555.123.5678</phone>
1006 [a130]      <fax>555.123.5679</fax>
1007 [a131]      <email/>
1008 [a132]      </insurer>
1009 [a133]      <medical>
1010 [a134]      <treatment>
1011 [a135]      <drug>
1012 [a136]      <name>methylphenidate hydrochloride</name>
1013 [a137]      <dailyDosage>30mgs</dailyDosage>
1014 [a138]      <startDate>1999-01-12</startDate>
1015 [a139]      </drug>
1016 [a140]      <comment>
1017 [a141]      patient exhibits side-effects of skin coloration and carpal
1018 degeneration
1019 [a142]      </comment>
1020 [a143]      </treatment>
1021 [a144]      <result>
1022 [a145]      <test>blood pressure</test>
1023 [a146]      <value>120/80</value>
1024 [a147]      <date>2001-06-09</date>
1025 [a148]      <performedBy>Nurse Betty</performedBy>
1026 [a149]      </result>
1027 [a150]      </medical>
1028 [a151]      </record>

```

## 4.2.2 Example request context

The following example illustrates a request **context** to which the example **rules** may be applicable. It represents a request by the physician Julius Hibbert to read the patient date of birth in the record of Bartholomew Simpson.

```

1033 [a152] <?xml version="1.0" encoding="UTF-8"?>
1034 [a153] <Request xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:cd:04"
1035 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
1036 urn:oasis:names:tc:xacml:2.0:context:schema:cd:04 http://docs.oasis-
1037 open.org/xacml/access_control-xacml-2.0-context-schema-cd-04.xsd">
1038 [a154] <Subject>
1039 [a155] <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject-category"
1040 DataType="http://www.w3.org/2001/XMLSchema#anyURI">
1041 [a156] <AttributeValue>urn:oasis:names:tc:xacml:1.0:subject-category:access-
1042 subject</AttributeValue>
1043 [a157] </Attribute>
1044 [a158] <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
1045 DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="med.example.com">
1046 [a159] <AttributeValue>CN=Julius Hibbert</AttributeValue>
1047 [a160] </Attribute>
1048 [a161] <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:name-
1049 format" DataType="http://www.w3.org/2001/XMLSchema#anyURI"
1050 Issuer="med.example.com">
1051 [a162] <AttributeValue>
1052 [a163] urn:oasis:names:tc:xacml:1.0:datatype:x500name
1053 [a164] </AttributeValue>
1054 [a165] </Attribute>
1055 [a166] <Attribute
1056 AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:role"
1057 DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="med.example.com">
1058 [a167] <AttributeValue>physician</AttributeValue>
1059 [a168] </Attribute>
1060 [a169] <Attribute
1061 AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:physician-id"
1062 DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="med.example.com">

```

Deleted: cd:03

Deleted: cd:03

Deleted: cd:03

Deleted: cd:03

access\_control-xacml-2.0-core-spec-cd-04

```

1063 [a170]    <AttributeValue>jhl234</AttributeValue>
1064 [a171]    </Attribute>
1065 [a172]    </Subject>
1066 [a173]    <Resource>
1067 [a174]    <ResourceContent>
1068 [a175]    <md:record xmlns:md="urn:example:med:schemas:record"
1069 xsi:schemaLocation="urn:example:med:schemas:record
1070 http://www.med.example.com/schemas/record.xsd">
1071 [a176]    <md:patient>
1072 [a177]    <md:patientDoB>1992-03-21</md:patientDoB>
1073 [a178]    <md:patient-number>555555</md:patient-number>
1074 [a179]    </md:patient>
1075 [a180]    </md:record>
1076 [a181]    </ResourceContent>
1077 [a182]    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-
1078 id" DataType="http://www.w3.org/2001/XMLSchema#string">
1079 [a183]    <AttributeValue>
1080 [a184]    //med.example.com/records/bart-simpson.xml#
1081 [a185]    xmlns(md=:Resource/ResourceContent/xpointer
1082 [a186]    (/md:record/md:patient/md:patientDoB)
1083 [a187]    </AttributeValue>
1084 [a188]    </Attribute>
1085 [a189]    </Resource>
1086 [a190]    <Action>
1087 [a191]    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1088 DataType="http://www.w3.org/2001/XMLSchema#string">
1089 [a192]    <AttributeValue>read</AttributeValue>
1090 [a193]    </Attribute>
1091 [a194]    </Action>
1092 [a195]    <Environment/>
1093 [a196]    </Request>

```

1094 [a152] - [a153] Standard namespace declarations.

1095 [a154] - [a172] **Subject** attributes are placed in the <Subject> element of the <Request>  
1096 element. Each **attribute** consists of the **attribute** meta-data and the **attribute** value. There is only  
1097 one subject involved in this **request**.

1098 [a155] - [a157] Each <Subject> element has a SubjectCategory attribute. The value of this  
1099 attribute describes the role that the related **subject** plays in making the **decision request**. The  
1100 value of "access-subject" denotes the identity for which the request was issued.

1101 [a158] - [a160] **Subject** subject-id **attribute**.

1102 [a161] - [a165] The format of the subject-id.

1103 [a166] - [a168] **Subject** role **attribute**.

1104 [a169] - [a171] **Subject** physician-id **attribute**.

1105 [a173] - [a189] **Resource attributes** are placed in the <Resource> element of the <Request>  
1106 element. Each **attribute** consists of **attribute** meta-data and an **attribute** value.

1107 [a174] - [a181] **Resource** content. The XML resource instance, access to all or part of which may  
1108 be requested, is placed here.

1109 [a182] - [a188] The identifier of the **Resource** instance for which access is requested, which is an  
1110 XPath expression into the <ResourceContent> element that selects the data to be accessed.

1111 [a190] - [a194] **Action attributes** are placed in the <Action> element of the <Request> element.

1112 [a192] **Action** identifier.

Deleted: cd-03

1113 [a195] The empty <Environment> element.

## 1114 4.2.3 Example plain-language rules

1115 The following plain-language rules are to be enforced:

1116 Rule 1: A person, identified by his or her patient number, may read any record for which he  
1117 or she is the designated patient.

1118 Rule 2: A person may read any record for which he or she is the designated parent or  
1119 guardian, and for which the patient is under 16 years of age.

1120 Rule 3: A physician may write to any medical element for which he or she is the designated  
1121 primary care physician, provided an email is sent to the patient.

1122 Rule 4: An administrator shall not be permitted to read or write to medical elements of a  
1123 patient record.

1124 These **rules** may be written by different **PAPs** operating independently, or by a single **PAP**.

## 1125 4.2.4 Example XACML rule instances

### 1126 4.2.4.1. Rule 1

1127 Rule 1 illustrates a simple **rule** with a single <Condition> element. It also illustrates the use of  
1128 the <VariableDefinition> element to define a function that may be used throughout the  
1129 **policy**. The following XACML <Rule> instance expresses Rule 1:

```
1130 [a197] <?xml version="1.0" encoding="UTF-8"?>
1131 [a198] <Policy
1132 [a199] xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:cd:04" xmlns:xacml-
1133 context="urn:oasis:names:tc:xacml:2.0:context:schema:cd:04"
1134 [a200] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
1135 urn:oasis:names:tc:xacml:2.0:policy:schema:cd:04 http://docs.oasis-
1136 open.org/xacml/access_control-xacml-2.0-context-schema-cd-04.xsd"
1137 [a201] xmlns:md="http://www.med.example.com/schemas/record.xsd"
1138 [a202] PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:1"
1139 [a203] RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1140 algorithm:deny-overrides">
1141 [a204] <PolicyDefaults>
1142 [a205] <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-
1143 19991116</XPathVersion>
1144 [a206] </PolicyDefaults>
1145 [a207] <Target/>
1146 [a208] <VariableDefinition VariableId="17590034">
1147 [a209] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1148 [a210] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-
1149 and-only">
1150 [a211] <SubjectAttributeDesignator
1151 AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:patient-number"
1152 [a212] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1153 [a213] </Apply>
1154 [a214] <Apply
1155 [a215] FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1156 [a216] <AttributeSelector
1157 [a217] RequestContextPath="//xacml-context:Resource/xacml-
1158 context:ResourceContent/md:record/md:patient/md:patient-number/text()"
1159 [a218] DataType="http://www.w3.org/2001/XMLSchema#string"/>
1160 [a219] </Apply>
```

Deleted: cd:03

Deleted: cd:03

Deleted: cd:03

Deleted: cd:03

Deleted: cd:03

access\_control-xacml-2.0-core-spec-cd-04



```

1161 [a220]    </Apply>
1162 [a221]    </VariableDefinition>
1163 [a222]    <Rule
1164 [a223]    RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:1"
1165 [a224]    Effect="Permit">
1166 [a225]    <Description>
1167 [a226]    A person may read any medical record in the
1168 [a227]    http://www.med.example.com/schemas/record.xsd namespace
1169 [a228]    for which he or she is the designated patient
1170 [a229]    </Description>
1171 [a230]    <Target>
1172 [a231]    <Resources>
1173 [a232]    <Resource>
1174 [a233]    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-
1175 equal">
1176 [a234]    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1177 [a235]    urn:example:med:schemas:record
1178 [a236]    </AttributeValue>
1179 [a237]    <ResourceAttributeDesignator AttributeId=
1180 [a238]    "urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1181 [a239]    DataType="http://www.w3.org/2001/XMLSchema#string"/>
1182 [a240]    </ResourceMatch>
1183 [a241]    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-
1184 node-match">
1185 [a242]    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1186 [a243]    /md:record
1187 [a244]    </AttributeValue>
1188 [a245]    <ResourceAttributeDesignator
1189 AttributeId="urn:oasis:names:tc:xacml:1.0:resource:xpath"
1190 [a246]    DataType="http://www.w3.org/2001/XMLSchema#string"/>
1191 [a247]    </ResourceMatch>
1192 [a248]    </Resource>
1193 [a249]    </Resources>
1194 [a250]    <Actions>
1195 [a251]    <Action>
1196 [a252]    <ActionMatch
1197 [a253]    MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1198 [a254]    <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1199 [a255]    read
1200 [a256]    </AttributeValue>
1201 [a257]    <ActionAttributeDesignator
1202 [a258]    AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1203 [a259]    DataType="http://www.w3.org/2001/XMLSchema#string"/>
1204 [a260]    </ActionMatch>
1205 [a261]    </Action>
1206 [a262]    </Actions>
1207 [a263]    </Target>
1208 [a264]    <Condition>
1209 [a265]    <VariableReference VariableId="17590034"/>
1210 [a266]    </Condition>
1211 [a267]    </Rule>
1212 [a268] </Policy>

```

1213 [a199] - [a201] XML namespace declarations.

1214 [a205] XPath expressions in the **policy** are to be interpreted according to the 1.0 version of the  
 1215 XPath specification.

1216 [a208] - [a221] A <VariableDefinition> element. It defines a function that evaluates the truth  
 1217 of the statement: the patient-number **subject attribute** is equal to the patient-number in the  
 1218 **resource**.

Deleted: cd-03



1219 [a209] The `FunctionId` attribute names the function to be used for comparison. In this case,  
 1220 comparison is done with the “urn:oasis:names:tc:xacml:1.0:function:string-equal” function; this  
 1221 function takes two arguments of type “http://www.w3.org/2001/XMLSchema#string”.

1222 [a210] The first argument of the variable definition is a function specified by the `FunctionId`  
 1223 attribute. Since `urn:oasis:names:tc:xacml:1.0:function:string-equal` takes  
 1224 arguments of type “http://www.w3.org/2001/XMLSchema#string” and  
 1225 `SubjectAttributeDesignator` selects a **bag** of type  
 1226 “http://www.w3.org/2001/XMLSchema#string”, “urn:oasis:names:tc:xacml:1.0:function:string-one-  
 1227 and-only” is used. This function guarantees that its argument evaluates to a **bag** containing exactly  
 1228 one value.

1229 [a211] The `SubjectAttributeDesignator` selects a **bag** of values for the `patient-number`  
 1230 **subject attribute** in the request **context**.

1231 [a215] The second argument of the variable definition is a function specified by the `FunctionId`  
 1232 attribute. Since “urn:oasis:names:tc:xacml:1.0:function:string-equal” takes arguments of type  
 1233 “http://www.w3.org/2001/XMLSchema#string” and the `AttributeSelector` selects a **bag** of type  
 1234 “http://www.w3.org/2001/XMLSchema#string”, “urn:oasis:names:tc:xacml:1.0:function:string-one-  
 1235 and-only” is used. This function guarantees that its argument evaluates to a **bag** containing exactly  
 1236 one value.

1237 [a216] The `<AttributeSelector>` element selects a **bag** of values from the request **context**  
 1238 using a free-form XPath expression. In this case, it selects the value of the `patient-number` in  
 1239 the **resource**. Note that the namespace prefixes in the XPath expression are resolved with the  
 1240 standard XML namespace declarations.

1241 [a223] **Rule** identifier.

1242 [a224] **Rule effect** declaration. When a **rule** evaluates to ‘True’ it emits the value of the `Effect`  
 1243 attribute. This value is then combined with the `Effect` values of other **rules** according to the **rule-**  
 1244 **combining algorithm**.

1245 [a225] - [a229] Free form description of the **rule**.

1246 [a230] - [a263] A **rule target** defines a set of **decision requests** that the **rule** is intended to  
 1247 evaluate. In this example, the `<Subjects>` and `<Environments>` elements are omitted.

1248 [a231] - [a249] The `<Resources>` element contains a **disjunctive sequence** of `<Resource>`  
 1249 elements. In this example, there is just one.

1250 [a232] - [a248] The `<Resource>` element encloses the **conjunctive sequence** of  
 1251 `ResourceMatch` elements. In this example, there are two.

1252 [a233] - [a240] The first `<ResourceMatch>` element compares its first and second child elements  
 1253 according to the matching function. A match is positive if the value of the first argument matches  
 1254 any of the values selected by the second argument. This match compares the target namespace of  
 1255 the requested document with the value of “urn:example:med:schemas:record”.

1256 [a233] The `MatchId` attribute names the matching function.

1257 [a235] Literal attribute value to match.

1258 [a237] - [a239] The `<ResourceAttributeDesignator>` element selects the target namespace  
 1259 from the resource contained in the request **context**. The **attribute** name is specified by the  
 1260 `AttributeId`.

1261 [a241] - [a247] The second `<ResourceMatch>` element. This match compares the results of two  
 1262 XPath expressions. The second XPath expression is the location path to the requested XML

Deleted: cd-03

1263 element and the first XPath expression is the literal value “/md:record”. The “xpath-node-match”  
 1264 function evaluates to “True” if the requested XML element is below the “/md:record” element.

1265 [a250] - [a262] The <Actions> element contains a **disjunctive sequence** of <Action> elements.  
 1266 In this case, there is just one <Action> element.

1267 [a251] - [a261] The <Action> element contains a **conjunctive sequence** of <ActionMatch>  
 1268 elements. In this case, there is just one <ActionMatch> element.

1269 [a252] - [a260] The <ActionMatch> element compares its first and second child elements  
 1270 according to the matching function. The match is positive if the value of the first argument matches  
 1271 any of the values selected by the second argument. In this case, the value of the action-id  
 1272 action attribute in the request **context** is compared with the literal value “read”.

1273 [a264] - [a266] The <Condition> element. A **condition** must evaluate to “True” for the **rule** to be  
 1274 applicable. This **condition** contains a reference to a variable definition defined elsewhere in the  
 1275 **policy**.

#### 1276 4.2.4.2. Rule 2

1277 Rule 2 illustrates the use of a mathematical function, i.e. the <Apply> element with functionId  
 1278 "urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration" to calculate the date of the  
 1279 patient's sixteenth birthday. It also illustrates the use of **predicate** expressions, with the  
 1280 functionId "urn:oasis:names:tc:xacml:1.0:function:and". This example has one function  
 1281 embedded in the <Condition> element and another one referenced in a  
 1282 <VariableDefinition> element.

```

1283 [a269] <?xml version="1.0" encoding="UTF-8"?>
1284 [a270] <Policy
1285 [a271] xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:cd:04" xmlns:xacml-
1286 context="urn:oasis:names:tc:xacml:2.0:context:schema:cd:04"
1287 [a272] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1288 xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:cd:04
1289 http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-cd-
1290 04.xsd"
1291 [a273] xmlns:xf="http://www.w3.org/TR/2002/WD-xquery-operators-20020816/#"
1292 [a274] xmlns:md="http://www.med.example.com/schemas/record.xsd"
1293 [a275] PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:2"
1294 RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-
1295 overrides">
1296 [a276] <PolicyDefaults>
1297 [a277] <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-
1298 19991116</XPathVersion>
1299 [a278] </PolicyDefaults>
1300 [a279] <Target/>
1301 [a280] <VariableDefinition VariableId="17590035">
1302 [a281] <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:date-less-or-
1303 equal">
1304 [a282] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-
1305 only">
1306 [a283] <EnvironmentAttributeDesignator
1307 [a284] AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-date"
1308 [a285] DataType="http://www.w3.org/2001/XMLSchema#date"/>
1309 [a286] </Apply>
1310 [a287] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-add-
1311 yearMonthDuration">
1312 [a288] <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:date-one-and-
1313 only">
1314 [a289] <AttributeSelector RequestContextPath=
1315 [a290] "/md:record/md:patient/md:patientDoB/text()"
  
```

Deleted: cd:03

Deleted: cd:03

Deleted: cd:03

Deleted: cd-03

Deleted: cd-03

```

1316 [a291]      DataType="http://www.w3.org/2001/XMLSchema#date" />
1317 [a292]      </Apply>
1318 [a293]      <AttributeValue
1319 [a294]      DataType="http://www.w3.org/TR/2002/WD-xquery-operators-
1320 20020816#yearMonthDuration">
1321 [a295]      <xf:dt-yearMonthDuration>
1322 [a296]      P16Y
1323 [a297]      </xf:dt-yearMonthDuration>
1324 [a298]      </AttributeValue>
1325 [a299]      </Apply>
1326 [a300]      </Apply>
1327 [a301]      </VariableDefinition>
1328 [a302]      <Rule
1329 [a303]      RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:2"
1330 [a304]      Effect="Permit">
1331 [a305]      <Description>
1332 [a306]      A person may read any medical record in the
1333 [a307]      http://www.med.example.com/records.xsd namespace
1334 [a308]      for which he or she is the designated parent or guardian,
1335 [a309]      and for which the patient is under 16 years of age
1336 [a310]      </Description>
1337 [a311]      <Target>
1338 [a312]      <Resources>
1339 [a313]      <Resource>
1340 [a314]      <ResourceMatch
1341 [a315]      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1342 [a316]      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1343 [a317]      http://www.med.example.com/schemas/record.xsd
1344 [a318]      </AttributeValue>
1345 [a319]      <ResourceAttributeDesignator AttributeId=
1346 "urn:oasis:names:tc:xacml:2.0:resource:target-namespace"
1347 [a320]      DataType="http://www.w3.org/2001/XMLSchema#string" />
1348 [a321]      </ResourceMatch>
1349 [a322]      <ResourceMatch
1350 [a323]      MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-node-match">
1351 [a324]      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1352 [a325]      /md:record
1353 [a326]      </AttributeValue>
1354 [a327]      <ResourceAttributeDesignator
1355 AttributeId="urn:oasis:names:tc:xacml:1.0:resource:xpath"
1356 [a328]      DataType="http://www.w3.org/2001/XMLSchema#string" />
1357 [a329]      </ResourceMatch>
1358 [a330]      </Resource>
1359 [a331]      </Resources>
1360 [a332]      <Actions>
1361 [a333]      <Action>
1362 [a334]      <ActionMatch
1363 [a335]      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1364 [a336]      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1365 [a337]      read
1366 [a338]      </AttributeValue>
1367 [a339]      <ActionAttributeDesignator
1368 AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1369 [a340]      DataType="http://www.w3.org/2001/XMLSchema#string" />
1370 [a341]      </ActionMatch>
1371 [a342]      </Action>
1372 [a343]      </Actions>
1373 [a344]      </Target>
1374 [a345]      <Condition>
1375 [a346]      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:and">
1376 [a347]      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1377 [a348]      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-
1378 and-only">

```

Deleted: cd-03

1379 [a349] <SubjectAttributeDesignator  
1380 AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:  
1381 [a350] parent-guardian-id"  
1382 [a351] DataType="http://www.w3.org/2001/XMLSchema#string"/>  
1383 [a352] </Apply>  
1384 [a353] <Apply  
1385 [a354] FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-  
1386 only">  
1387 [a355] <AttributeSelector  
1388 [a356] RequestContextPath="//xacml-context:Resource/xacml-  
1389 context:ResourceContent/md:record/md:parentGuardian/md:parentGuardianId/text()"  
1390 [a357] DataType="http://www.w3.org/2001/XMLSchema#string"/>  
1391 [a358] </Apply>  
1392 [a359] </Apply>  
1393 [a360] <VariableReference VariableId="17590035"/>  
1394 [a361] </Apply>  
1395 [a362] </Condition>  
1396 [a363] </Rule>  
1397 [a364] </Policy>

1398 [a280] - [a301] The <VariableDefinition> element contains part of the **condition** (i.e. is the  
1399 patient under 16 years of age?). The patient is under 16 years of age if the current date is less than  
1400 the date computed by adding 16 to the patient's date of birth.

1401 [a281] - [a300] "urn:oasis:names:tc:xacml:1.0:function:date-less-or-equal" is used to compute the  
1402 difference of two date arguments.

1403 [a282] - [a286] The first date argument uses "urn:oasis:names:tc:xacml:1.0:function:date-one-and-  
1404 only" to ensure that the **bag** of values selected by its argument contains exactly one value of type  
1405 "http://www.w3.org/2001/XMLSchema#date".

1406 [a284] The current date is evaluated by selecting the  
1407 "urn:oasis:names:tc:xacml:1.0:environment:current-date" **environment attribute**.

1408 [a287] - [a299] The second date argument uses "urn:oasis:names:tc:xacml:1.0:function:date-add-  
1409 yearMonthDuration" to compute the date of the patient's sixteenth birthday by adding 16 years to  
1410 the patient's date of birth. The first of its arguments is of type  
1411 "http://www.w3.org/2001/XMLSchema#date" and the second is of type  
1412 "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration".

1413 [a289] The <AttributeSelector> element selects the patient's date of birth by taking the XPath  
1414 expression over the **resource** content.

1415 [a293] - [a298] Year Month Duration of 16 years.

1416 [a311] - [a344] **Rule** declaration and **rule target**. See Rule 1 in Section 4.2.4.1 for the detailed  
1417 explanation of these elements.

1418 [a345] - [a362] The <Condition> element. The **condition** must evaluate to "True" for the **rule** to  
1419 be applicable. This **condition** evaluates the truth of the statement: the requestor is the designated  
1420 parent or guardian and the patient is under 16 years of age. It contains one embedded <Apply>  
1421 element and one referenced <VariableDefinition> element.

1422 [a346] The **condition** uses the "urn:oasis:names:tc:xacml:1.0:function:and" function. This is a  
1423 Boolean function that takes one or more Boolean arguments (2 in this case) and performs the  
1424 logical "AND" operation to compute the truth value of the expression.

1425 [a347] - [a359] The first part of the **condition** is evaluated (i.e. is the requestor the designated  
1426 parent or guardian?). The function is "urn:oasis:names:tc:xacml:1.0:function:string-equal" and it  
1427 takes two arguments of type "http://www.w3.org/2001/XMLSchema#string".

Deleted: cd-03

1428 [a348] designates the first argument. Since "urn:oasis:names:tc:xacml:1.0:function:string-equal"  
 1429 takes arguments of type "http://www.w3.org/2001/XMLSchema#string",  
 1430 "urn:oasis:names:tc:xacml:1.0:function:string-one-and-only" is used to ensure that the **subject**  
 1431 **attribute** "urn:oasis:names:tc:xacml:2.0:example:attribute:parent-guardian-id" in the request  
 1432 **context** contains exactly one value.

1433 [a353] designates the second argument. The value of the **subject attribute**  
 1434 "urn:oasis:names:tc:xacml:2.0:example:attribute:parent-guardian-id" is selected from the request  
 1435 **context** using the <SubjectAttributeDesignator> element.

1436 [a354] As above, the "urn:oasis:names:tc:xacml:1.0:function:string-one-and-only" is used to ensure  
 1437 that the **bag** of values selected by it's argument contains exactly one value of type  
 1438 "http://www.w3.org/2001/XMLSchema#string".

1439 [a355] The second argument selects the value of the <md:parentGuardianId> element from the  
 1440 **resource** content using the <AttributeSelector> element. This element contains a free-form  
 1441 XPath expression, pointing into the request **context**. Note that all namespace prefixes in the XPath  
 1442 expression are resolved with standard namespace declarations. The AttributeSelector  
 1443 evaluates to the **bag** of values of type "http://www.w3.org/2001/XMLSchema#string".

1444 [a360] references the <VariableDefinition> element, where the second part of the **condition**  
 1445 is defined.

#### 1446 4.2.4.3. Rule 3

1447 Rule 3 illustrates the use of an **obligation**. The XACML <Rule> element syntax does not include  
 1448 an element suitable for carrying an **obligation**, therefore Rule 3 has to be formatted as a  
 1449 <Policy> element.

```

1450 [a365] <?xml version="1.0" encoding="UTF-8"?>
1451 [a366] <Policy
1452 [a367]   xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:cd:04" xmlns:xacml-
1453 [a368]   context="urn:oasis:names:tc:xacml:2.0:context:schema:cd:04"
1454 [a369]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1455 [a369]   xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:cd:04
1456 [a369]   http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-cd-
1457 [a369]   04.xsd"
1458 [a370]   xmlns:md="http://www.med.example.com/schemas/record.xsd"
1459 [a371]   PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:3"
1460 [a372]   RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1461 [a372]   algorithm:deny-overrides">
1462 [a373]   <Description>
1463 [a374]     Policy for any medical record in the
1464 [a375]     http://www.med.example.com/schemas/record.xsd namespace
1465 [a376]   </Description>
1466 [a377]   <PolicyDefaults>
1467 [a378]     <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-
1468 [a378]     19991116</XPathVersion>
1469 [a379]   </PolicyDefaults>
1470 [a380]   <Target>
1471 [a381]     <Resources>
1472 [a382]       <Resource>
1473 [a383]         <ResourceMatch
1474 [a384]           MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1475 [a385]             <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1476 [a386]               urn:example:med:schemas:record
1477 [a387]             </AttributeValue>
1478 [a388]             <ResourceAttributeDesignator AttributeId=
1479 [a389]               "urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
1480 [a390]             DataType="http://www.w3.org/2001/XMLSchema#string" />

```

Deleted: cd:03

Deleted: cd:03

Deleted: cd:03

Deleted: cd-03

Deleted: cd-03

access\_control-xacml-2.0-core-spec-cd-04

37

```

1481 [a391]      </ResourceMatch>
1482 [a392]      </Resource>
1483 [a393]      </Resources>
1484 [a394]      </Target>
1485 [a395]      <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:3"
1486 [a396]      Effect="Permit">
1487 [a397]      <Description>
1488 [a398]      A physician may write any medical element in a record
1489 [a399]      for which he or she is the designated primary care
1490 [a400]      physician, provided an email is sent to the patient
1491 [a401]      </Description>
1492 [a402]      <Target>
1493 [a403]      <Subjects>
1494 [a404]      <Subject>
1495 [a405]      <SubjectMatch
1496 [a406]      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1497 [a407]      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1498 [a408]      physician
1499 [a409]      </AttributeValue>
1500 [a410]      <SubjectAttributeDesignator AttributeId=
1501 "urn:oasis:names:tc:xacml:2.0:example:attribute:role"
1502 [a411]      DataType="http://www.w3.org/2001/XMLSchema#string" />
1503 [a412]      </SubjectMatch>
1504 [a413]      </Subject>
1505 [a414]      </Subjects>
1506 [a415]      <Resources>
1507 [a416]      <Resource>
1508 [a417]      <ResourceMatch
1509 [a418]      MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-node-match">
1510 [a419]      <AttributeValue
1511 [a420]      DataType="http://www.w3.org/2001/XMLSchema#string">
1512 [a421]      /md:record/md:medical
1513 [a422]      </AttributeValue>
1514 [a423]      <ResourceAttributeDesignator
1515 AttributeId="urn:oasis:names:tc:xacml:1.0:resource:xpath"
1516 [a424]      DataType="http://www.w3.org/2001/XMLSchema#string" />
1517 [a425]      </ResourceMatch>
1518 [a426]      </Resource>
1519 [a427]      </Resources>
1520 [a428]      <Actions>
1521 [a429]      <Action>
1522 [a430]      <ActionMatch
1523 [a431]      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1524 [a432]      <AttributeValue
1525 [a433]      DataType="http://www.w3.org/2001/XMLSchema#string">
1526 [a434]      write
1527 [a435]      </AttributeValue>
1528 [a436]      <ActionAttributeDesignator
1529 AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1530 [a437]      DataType="http://www.w3.org/2001/XMLSchema#string" />
1531 [a438]      </ActionMatch>
1532 [a439]      </Action>
1533 [a440]      </Actions>
1534 [a441]      </Target>
1535 [a442]      <Condition>
1536 [a443]      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1537 [a444]      <Apply
1538 [a445]      FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1539 [a446]      <SubjectAttributeDesignator
1540 [a447]      AttributeId="urn:oasis:names:tc:xacml:2.0:example:
1541 attribute:physician-id"
1542 [a448]      DataType="http://www.w3.org/2001/XMLSchema#string" />
1543 [a449]      </Apply>

```

Deleted: cd-03

```

1544 [a450]      <Apply
1545 [a451]      FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
1546 [a452]      <AttributeSelector RequestContextPath=
1547 [a453]      " //xacml-context:Resource/xacml-
1548 context:ResourceContent/md:record/md:primaryCarePhysician/md:registrationID/text(
1549 ) "
1550 [a454]      DataType="http://www.w3.org/2001/XMLSchema#string"/>
1551 [a455]      </Apply>
1552 [a456]      </Apply>
1553 [a457]      </Condition>
1554 [a458]      </Rule>
1555 [a459]      <Obligations>
1556 [a460]      <Obligation
1557 ObligationId="urn:oasis:names:tc:xacml:example:obligation:email"
1558 [a461]      FulfillOn="Permit">
1559 [a462]      <AttributeAssignment
1560 AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:mailto"
1561 [a463]      DataType="http://www.w3.org/2001/XMLSchema#string">
1562 [a464]      <AttributeSelector RequestContextPath=
1563 [a465]      " //md:/record/md:patient/md:patientContact/md:email"
1564 [a466]      DataType="http://www.w3.org/2001/XMLSchema#string"/>
1565 [a467]      </AttributeAssignment>
1566 [a468]      <AttributeAssignment
1567 AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:text"
1568 [a469]      DataType="http://www.w3.org/2001/XMLSchema#string">
1569 [a470]      Your medical record has been accessed by:
1570 [a471]      </AttributeAssignment>
1571 [a472]      <AttributeAssignment
1572 AttributeId="urn:oasis:names:tc:xacml:2.0:example:attribute:text"
1573 [a473]      DataType="http://www.w3.org/2001/XMLSchema#string">
1574 [a474]      <SubjectAttributeDesignator
1575 AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
1576 [a475]      DataType="http://www.w3.org/2001/XMLSchema#string"/>
1577 [a476]      </AttributeAssignment>
1578 [a477]      </Obligation>
1579 [a478]      </Obligations>
1580 [a479] </Policy>

```

1581 [a366] - [a372] The <Policy> element includes standard namespace declarations as well as policy  
 1582 specific parameters, such as PolicyId and RuleCombiningAlgId.

1583 [a371] **Policy** identifier. This parameter allows the **policy** to be referenced by a **policy set**.

1584 [a372] The **rule combining algorithm** identifies the algorithm for combining the outcomes of **rule**  
 1585 evaluation.

1586 [a373] - [a376] Free-form description of the **policy**.

1587 [a379] - [a394] **Policy target**. The **policy target** defines a set of applicable decision requests. The  
 1588 structure of the <Target> element in the <Policy> is identical to the structure of the <Target>  
 1589 element in the <Rule>. In this case, the **policy target** is the set of all XML resources that conform  
 1590 to the namespace "urn:example:med:schemas:record".

1591 [a395] The only <Rule> element included in this <Policy>. Two parameters are specified in the  
 1592 **rule** header: RuleId and Effect.

1593 [a402] - [a441] The **rule target** further constrains the **policy target**.

1594 [a405] - [a412] The <SubjectMatch> element targets the **rule** at **subjects** whose  
 1595 "urn:oasis:names:tc:xacml:2.0:example:attribute:role" **subject attribute** is equal to "physician".

Deleted: cd-03



1596 [a417] - [a425] The <ResourceMatch> element targets the **rule** at **resources** that match the  
 1597 XPath expression “/md:record/md:medical”.

1598 [a430] - [a438] The <ActionMatch> element targets the **rule** at **actions** whose  
 1599 “urn:oasis:names:tc:xacml:1.0:action:action-id” **action attribute** is equal to “write”.

1600 [a442] - [a457] The <Condition> element. For the **rule** to be applicable to the **decision request**,  
 1601 the **condition** must evaluate to “True”. This **condition** compares the value of the  
 1602 “urn:oasis:names:tc:xacml:2.0:example:attribute:physician-id” **subject attribute** with the value of  
 1603 the <registrationId> element in the medical record that is being accessed.

1604 [a459] - [a478] The <Obligations> element. **Obligations** are a set of operations that must be  
 1605 performed by the **PEP** in conjunction with an **authorization decision**. An **obligation** may be  
 1606 associated with a “Permit” or “Deny” **authorization decision**. The element contains a single  
 1607 **obligation**.

1608 [a460] - [a477] The <Obligation> element consists of the ObligationId attribute, the  
 1609 **authorization decision** value for which it must be fulfilled, and a set of **attribute** assignments. The  
 1610 **PDP** does not resolve the attribute assignments. This is the job of the **PEP**.

1611 [a460] The ObligationId attribute identifies the **obligation**. In this case, the **PEP** is required to  
 1612 send email.

1613 [a461] The FulfillOn attribute defines the **authorization decision** value for which this  
 1614 **obligation** must be fulfilled. In this case, when access is permitted.

1615 [a462] - [a467] The first parameter indicates where the **PEP** will find the email address in the  
 1616 resource.

1617 [a468] - [a471] The second parameter contains literal text for the email body.

1618 [a472] - [a476] The third parameter indicates where the **PEP** will find further text for the email body  
 1619 in the resource.

#### 4.2.4.4. Rule 4

1621 Rule 4 illustrates the use of the “Deny” Effect value, and a <Rule> with no <Condition>  
 1622 element.

```

1623 [a480] <?xml version="1.0" encoding="UTF-8"?>
1624 [a481] <Policy
1625 | [a482] xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:cd:04"
1626 | [a483] xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1627 | xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:cd:04
1628 | http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-cd-
1629 | 04.xsd"
1630 [a484] xmlns:md="http://www.med.example.com/schemas/record.xsd"
1631 [a485] PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:4"
1632 [a486] RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
1633 algorithm:deny-overrides">
1634 [a487] <PolicyDefaults>
1635 | [a488] <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-
1636 | 19991116</XPathVersion>
1637 [a489] </PolicyDefaults>
1638 [a490] <Target/>
1639 [a491] <Rule
1640 | [a492] RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:4"
1641 | [a493] Effect="Deny">
1642 | [a494] <Description>
1643 | [a495] An Administrator shall not be permitted to read or write
1644 | [a496] medical elements of a patient record in the
  
```

Deleted: cd:03

Deleted: cd:03

Deleted: cd-03

Deleted: cd-03

access\_control-xacml-2.0-core-spec-cd-04



```

1645 [a497]      http://www.med.example.com/records.xsd namespace.
1646 [a498]      </Description>
1647 [a499]      <Target>
1648 [a500]      <Subjects>
1649 [a501]      <Subject>
1650 [a502]      <SubjectMatch
1651 [a503]      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1652 [a504]      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1653 [a505]      administrator
1654 [a506]      </AttributeValue>
1655 [a507]      <SubjectAttributeDesignator AttributeId=
1656 [a508]      "urn:oasis:names:tc:xacml:2.0:example:attribute:role"
1657 [a509]      DataType="http://www.w3.org/2001/XMLSchema#string"/>
1658 [a510]      </SubjectMatch>
1659 [a511]      </Subject>
1660 [a512]      </Subjects>
1661 [a513]      <Resources>
1662 [a514]      <Resource>
1663 [a515]      <ResourceMatch
1664 [a516]      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1665 [a517]      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1666 [a518]      urn:example:med:schemas:record
1667 [a519]      </AttributeValue>
1668 [a520]      <ResourceAttributeDesignator
1669 AttributeId="urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
1670 [a521]      DataType="http://www.w3.org/2001/XMLSchema#string"/>
1671 [a522]      </ResourceMatch>
1672 [a523]      <ResourceMatch
1673 [a524]      MatchId="urn:oasis:names:tc:xacml:1.0:function:xpath-node-match">
1674 [a525]      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1675 [a526]      /md:record/md:medical
1676 [a527]      </AttributeValue>
1677 [a528]      <ResourceAttributeDesignator
1678 AttributeId="urn:oasis:names:tc:xacml:1.0:resource:xpath"
1679 [a529]      DataType="http://www.w3.org/2001/XMLSchema#string"/>
1680 [a530]      </ResourceMatch>
1681 [a531]      </Resource>
1682 [a532]      </Resources>
1683 [a533]      <Actions>
1684 [a534]      <Action>
1685 [a535]      <ActionMatch
1686 [a536]      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1687 [a537]      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1688 [a538]      read
1689 [a539]      </AttributeValue>
1690 [a540]      <ActionAttributeDesignator
1691 AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1692 [a541]      DataType="http://www.w3.org/2001/XMLSchema#string"/>
1693 [a542]      </ActionMatch>
1694 [a543]      </Action>
1695 [a544]      <Action>
1696 [a545]      <ActionMatch
1697 [a546]      MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
1698 [a547]      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
1699 [a548]      write
1700 [a549]      </AttributeValue>
1701 [a550]      <ActionAttributeDesignator
1702 AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
1703 [a551]      DataType="http://www.w3.org/2001/XMLSchema#string"/>
1704 [a552]      </ActionMatch>
1705 [a553]      </Action>
1706 [a554]      </Actions>
1707 [a555]      </Target>

```

Deleted: cd-03

1708 [a556] </Rule>  
1709 [a557] </Policy>  
1710 [a492] - [a493] The <Rule> element declaration.  
1711 [a493] **Rule Effect.** Every **rule** that evaluates to "True" emits the **rule effect** as its value. This  
1712 **rule Effect** is "Deny" meaning that according to this **rule**, access must be denied when it  
1713 evaluates to "True".  
1714 [a494] - [a498] Free form description of the **rule**.  
1715 [a499] - [a555] **Rule target.** The **Rule target** defines the set of **decision requests** that are  
1716 applicable to the **rule**.  
1717 [a502] - [a510] The <SubjectMatch> element targets the **rule** at **subjects** whose  
1718 "urn:oasis:names:tc:xacml:2.0:example:attribute:role" **subject attribute** is equal to  
1719 "administrator".  
1720 [a513] - [a532] The <Resources> element contains one <Resource> element, which (in turn)  
1721 contains two <ResourceMatch> elements. The **target** matches if the **resource** identified by the  
1722 request **context** matches both **resource** match criteria.  
1723 [a515]-[a522] The first <ResourceMatch> element targets the **rule** at **resources**  
1724 whose "urn:oasis:names:tc:xacml:2.0:resource:target-namespace" **resource attribute**  
1725 is equal to "urn:example:med:schemas:record".  
1726 [a523] - [a530] The second <ResourceMatch> element targets the **rule** at XML elements that  
1727 match the XPath expression "/md:record/md:medical".  
1728 [a533] - [a554] The <Actions> element contains two <Action> elements, each of which contains  
1729 one <ActionMatch> element. The **target** matches if the **action** identified in the request **context**  
1730 matches either of the **action** match criteria.  
1731 [a535] - [a552] The <ActionMatch> elements target the **rule** at **actions** whose  
1732 "urn:oasis:names:tc:xacml:1.0:action:action-id" **action attribute** is equal to "read" or "write".  
1733 This **rule** does not have a <Condition> element.

1734 **4.2.4.5. Example PolicySet**

1735 This section uses the examples of the previous sections to illustrate the process of combining  
1736 **policies**. The policy governing read access to medical elements of a record is formed from each of  
1737 the four **rules** described in Section 4.2.3. In plain language, the combined rule is:

- 1738 • Either the requestor is the patient; or  
1739 • the requestor is the parent or guardian and the patient is under 16; or  
1740 • the requestor is the primary care physician and a notification is sent to the patient; and  
1741 • the requestor is not an administrator.

1742 The following **policy set** illustrates the combined **policies**. **Policy** 3 is included by reference and  
1743 **policy** 2 is explicitly included.

```
[a558] <?xml version="1.0" encoding="UTF-8"?>
[a559] <PolicySet
[a560]   xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:cd:04"
[a561]   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:cd:04
  http://docs.oasis-open.org/xacml/access_control-xacml-2.0-policy-schema-cd-
  04.xsd"
  access_control-xacml-2.0-core-spec:cd-04
```

Deleted: cd:03

Deleted: cd:03

Deleted: cd-03

Deleted: cd-03

```

[a562] PolicySetId=
[a563] "urn:oasis:names:tc:xacml:2.0:example:policysetid:1"
[a564] PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:
[a565] policy-combining-algorithm:deny-overrides">
[a566] <Description>
[a567]   Example policy set.
[a568] </Description>
[a569] <Target>
[a570]   <Resources>
[a571]     <Resource>
[a572]       <ResourceMatch
[a573]         MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
[a574]           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
[a575]             urn:example:med:schema:records
[a576]           </AttributeValue>
[a577]           <ResourceAttributeDesignator AttributeId=
[a578]             "urn:oasis:names:tc:xacml:1.0:resource:target-namespace"
[a579]             DataType="http://www.w3.org/2001/XMLSchema#string" />
[a580]         </ResourceMatch>
[a581]       </Resource>
[a582]     </Resources>
[a583]   </Target>
[a584] <PolicyIdReference>
[a585]   urn:oasis:names:tc:xacml:2.0:example:policyid:3
[a586] </PolicyIdReference>
[a587] <Policy
[a588]   PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:2"
[a589]   RuleCombiningAlgId=
[a590]   "urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
[a591]   <Target/>
[a592]   <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:1"
[a593]     Effect="Permit">
[a594]   </Rule>
[a595]   <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:2"
[a596]     Effect="Permit">
[a597]   </Rule>
[a598]   <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:4"
[a599]     Effect="Deny">
[a600]   </Rule>
[a601] </Policy>
[a602] </PolicySet>

```

1744

1745 | [\[a559\]](#) - [\[a565\]](#) The <PolicySet> element declaration. Standard XML namespace declarations  
1746 | are included.

Deleted: [a560]

Deleted: [a566]

1747 | [\[a562\]](#) The PolicySetId attribute is used for identifying this **policy set** for possible inclusion in  
1748 | another **policy set**.

Deleted: [a563]

1749 | [\[a564\]](#) The **policy combining algorithm** identifier. **Policies** and **policy sets** in this **policy set** are  
1750 | combined according to the specified **policy combining algorithm** when the **authorization**  
1751 | **decision** is computed.

Deleted: [a565]

1752 | [\[a566\]](#) - [\[a568\]](#) Free form description of the **policy set**.

Deleted: [a567]

Deleted: [a569]

1753 | [\[a569\]](#) - [\[a583\]](#) The **policy set** <Target> element defines the set of **decision requests** that are  
1754 | applicable to this <PolicySet> element.

Deleted: [a570]

Deleted: [a584]

1755 | [\[a584\]](#) PolicyIdReference includes a **policy** by id.

Deleted: [a585]

1756 | [\[a588\]](#) Policy 2 is explicitly included in this **policy set**. The **rules** in Policy 2 are omitted for  
1757 | clarity.

Deleted: [a589]

Deleted: cd-03

access\_control-xacml-2.0-core-spec-[cd-04](#)

43

## 5. Policy syntax (normative, with the exception of the schema fragments)

### 5.1. Element <PolicySet>

The <PolicySet> element is a top-level element in the XACML policy schema. <PolicySet> is an aggregation of other **policy sets** and **policies**. **Policy sets** MAY be included in an enclosing <PolicySet> element either directly using the <PolicySet> element or indirectly using the <PolicySetIdReference> element. **Policies** MAY be included in an enclosing <PolicySet> element either directly using the <Policy> element or indirectly using the <PolicyIdReference> element.

A <PolicySet> element MAY be evaluated, in which case the evaluation procedure defined in Section 7.11 SHALL be used.

If a <PolicySet> element contains references to other **policy sets** or **policies** in the form of URLs, then these references MAY be resolvable.

**Policy sets** and **policies** included in a <PolicySet> element MUST be combined using the algorithm identified by the PolicyCombiningAlgId attribute. <PolicySet> is treated exactly like a <Policy> in all **policy combining algorithms**.

The <Target> element defines the applicability of the <PolicySet> element to a set of **decision requests**. If the <Target> element within the <PolicySet> element matches the **request context**, then the <PolicySet> element MAY be used by the **PDP** in making its **authorization decision**. See Section 7.11.

The <Obligations> element contains a set of **obligations** that MUST be fulfilled by the **PEP** in conjunction with the **authorization decision**. If the **PEP** does not understand, or cannot fulfill, any of the **obligations**, then it MUST act as if the **PDP** had returned a "Deny" **authorization decision** value. See Section 7.14.

```
<xs:element name="PolicySet" type="xacml:PolicySetType"/>
<xs:complexType name="PolicySetType">
  <xs:sequence>
    <xs:element ref="xacml:Description" minOccurs="0"/>
    <xs:element ref="xacml:PolicySetDefaults" minOccurs="0"/>
    <xs:element ref="xacml:Target"/>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="xacml:PolicySet"/>
      <xs:element ref="xacml:Policy"/>
      <xs:element ref="xacml:PolicySetIdReference"/>
      <xs:element ref="xacml:PolicyIdReference"/>
      <xs:element ref="xacml:CombinerParameters"/>
      <xs:element ref="xacml:PolicyCombinerParameters"/>
      <xs:element ref="xacml:PolicySetCombinerParameters"/>
    </xs:choice>
    <xs:element ref="xacml:Obligations" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="PolicySetId" type="xs:anyURI" use="required"/>
  <xs:attribute name="Version" type="xacml:VersionType" default="1.0"/>
  <xs:attribute name="PolicyCombiningAlgId" type="xs:anyURI" use="required"/>
</xs:complexType>
```

The <PolicySet> element is of **PolicySetType** complex type.

The <PolicySet> element contains the following attributes and elements:

Deleted: cd-03

access\_control-xacml-2.0-core-spec-[cd-04](#)

1805 PolicySetId [Required]

1806       **Policy set** identifier. It is the responsibility of the **PAP** to ensure that no two **policies**  
1807 visible to the **PDP** have the same identifier. This MAY be achieved by following a  
1808 predefined URN or URI scheme. If the **policy set** identifier is in the form of a URL, then it  
1809 MAY be resolvable.

1810 Version [Default 1.0]

1811       The version number of the **PolicySet**.

1812 PolicyCombiningAlgId [Required]

1813       The identifier of the **policy-combining algorithm** by which the <PolicySet>, <CombinerParameters>, <PolicyCombinerParameters> and  
1814 <PolicySetCombinerParameters> components MUST be combined. Standard  
1815 **policy-combining algorithms** are listed in Appendix C. Standard **policy-combining**  
1816 **algorithm** identifiers are listed in Section B.10.

1818 <Description> [Optional]

1819       A free-form description of the **policy set**.

1820 <PolicySetDefaults> [Optional]

1821       A set of default values applicable to the **policy set**. The scope of the  
1822 <PolicySetDefaults> element SHALL be the enclosing **policy set**.

1823 <Target> [Required]

1824       The <Target> element defines the applicability of a **policy set** to a set of **decision**  
1825 **requests**.

1826       The <Target> element MAY be declared by the creator of the <PolicySet> or it MAY be  
1827 computed from the <Target> elements of the referenced <Policy> elements, either as  
1828 an intersection or as a union.

1829 <PolicySet> [Any Number]

1830       A **policy set** that is included in this **policy set**.

1831 <Policy> [Any Number]

1832       A **policy** that is included in this **policy set**.

1833 <PolicySetIdReference> [Any Number]

1834       A reference to a **policy set** that MUST be included in this **policy set**. If  
1835 <PolicySetIdReference> is a URL, then it MAY be resolvable.

1836 <PolicyIdReference> [Any Number]

1837       A reference to a **policy** that MUST be included in this **policy set**. If the  
1838 <PolicyIdReference> is a URL, then it MAY be resolvable.

1839 <Obligations> [Optional]

1840       Contains the set of <Obligation> elements. See Section 7.14 for a description of how  
1841 the set of **obligations** to be returned by the **PDP** shall be determined.

1842 <CombinerParameters> [Optional]

Deleted: cd-03

1843 Contains a sequence of `<CombinerParameter>` elements.

1844 `<PolicyCombinerParameters>` [Optional]

1845 Contains a sequence of `<CombinerParameter>` elements that are associated with a

1846 particular `<Policy>` or `<PolicyIdReference>` element within the `<PolicySet>`.

1847 `<PolicySetCombinerParameters>` [Optional]

1848 Contains a sequence of `<CombinerParameter>` elements that are associated with a

1849 particular `<PolicySet>` or `<PolicySetIdReference>` element within the

1850 `<PolicySet>`.

## 1851 5.2. Element `<Description>`

1852 The `<Description>` element contains a free-form description of the `<PolicySet>`, `<Policy>`

1853 or `<Rule>` element. The `<Description>` element is of **xs:string** simple type.

1854 

```
<xs:element name="Description" type="xs:string"/>
```

## 1855 5.3. Element `<PolicySetDefaults>`

1856 The `<PolicySetDefaults>` element SHALL specify default values that apply to the

1857 `<PolicySet>` element.

1858 

```
<xs:element name="PolicySetDefaults" type="xacml:DefaultsType"/>
```

1859 

```
<xs:complexType name="DefaultsType">
```

1860 

```
  <xs:sequence>
```

1861 

```
    <xs:choice>
```

1862 

```
      <xs:element ref="xacml:XPathVersion" minOccurs="0"/>
```

1863 

```
    </xs:choice>
```

1864 

```
  </xs:sequence>
```

1865 

```
</xs:complexType>
```

1866 `<PolicySetDefaults>` element is of **DefaultsType** complex type.

1867 The `<PolicySetDefaults>` element contains the following elements:

1868 `<XPathVersion>` [Optional]

1869 Default XPath version.

## 1870 5.4. Element `<XPathVersion>`

1871 The `<XPathVersion>` element SHALL specify the version of the XPath specification to be used by

1872 `<AttributeSelector>` elements and XPath-based functions in the **policy set** or **policy**.

1873 

```
<xs:element name="XPathVersion" type="xs:anyURI"/>
```

1874 The URI for the XPath 1.0 specification is "http://www.w3.org/TR/1999/Rec-xpath-19991116". The

1875 `<XPathVersion>` element is REQUIRED if the XACML enclosing **policy set** or **policy** contains

1876 `<AttributeSelector>` elements or XPath-based functions.

## 1877 5.5. Element `<Target>`

1878 The `<Target>` element identifies the set of **decision requests** that the parent element is intended

1879 to evaluate. The `<Target>` element SHALL appear as a child of a `<PolicySet>` and `<Policy>`

Deleted: cd-03

1880 element and MAY appear as a child of a <Rule> element. It contains definitions for **subjects**,  
1881 **resources**, **actions** and **environments**.

1882 The <Target> element SHALL contain a **conjunctive sequence** of <Subjects>, <Resources>  
1883 <Actions> and <Environments> elements. For the parent of the <Target> element to be  
1884 applicable to the **decision request**, there MUST be at least one positive match between each  
1885 section of the <Target> element and the corresponding section of the <xacml-  
1886 context:Request> element.

```
1887 <xs:element name="Target" type="xacml:TargetType" />
1888 <xs:complexType name="TargetType">
1889   <xs:sequence>
1890     <xs:element ref="xacml:Subjects" minOccurs="0" />
1891     <xs:element ref="xacml:Resources" minOccurs="0" />
1892     <xs:element ref="xacml:Actions" minOccurs="0" />
1893     <xs:element ref="xacml:Environments" minOccurs="0" />
1894   </xs:sequence>
1895 </xs:complexType>
```

1896 The <Target> element is of **TargetType** complex type.

1897 The <Target> element contains the following elements:

1898 <Subjects> [Optional]

1899 Matching specification for the **subject attributes** in the **context**. If this element is missing,  
1900 then the **target** SHALL match all **subjects**.

1901 <Resources> [Optional]

1902 Matching specification for the **resource attributes** in the **context**. If this element is  
1903 missing, then the **target** SHALL match all **resources**.

1904 <Actions> [Optional]

1905 Matching specification for the **action attributes** in the **context**. If this element is missing,  
1906 then the **target** SHALL match all **actions**.

1907 <Environments> [Optional]

1908 Matching specification for the **environment attributes** in the **context**. If this element is  
1909 missing, then the **target** SHALL match all **environments**.

## 1910 5.6. Element <Subjects>

1911 The <Subjects> element SHALL contain a **disjunctive sequence** of <Subject> elements.

```
1912 <xs:element name="Subjects" type="xacml:SubjectsType" />
1913 <xs:complexType name="SubjectsType">
1914   <xs:sequence>
1915     <xs:element ref="xacml:Subject" maxOccurs="unbounded" />
1916   </xs:sequence>
1917 </xs:complexType>
```

1918 The <Subjects> element is of **SubjectsType** complex type.

1919 The <Subjects> element contains the following elements:

1920 <Subject> [One to Many, Required]

1921 See Section 5.7.

Deleted: cd-03

## 5.7. Element <Subject>

The <Subject> element SHALL contain a **conjunctive sequence** of <SubjectMatch> elements.

```
<xs:element name="Subject" type="xacml:SubjectType" />
<xs:complexType name="SubjectType">
  <xs:sequence>
    <xs:element ref="xacml:SubjectMatch" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
```

The <Subject> element is of **SubjectType** complex type.

The <Subject> element contains the following elements:

<SubjectMatch> [One to Many]

A **conjunctive sequence** of individual matches of the **subject attributes** in the request **context** and the embedded **attribute** values. See Section 5.8.

## 5.8. Element <SubjectMatch>

The <SubjectMatch> element SHALL identify a set of **subject**-related entities by matching **attribute** values in a <xacml-context:Subject> element of the request **context** with the embedded **attribute** value.

```
<xs:element name="SubjectMatch" type="xacml:SubjectMatchType" />
<xs:complexType name="SubjectMatchType">
  <xs:sequence>
    <xs:element ref="xacml:AttributeValue" />
    <xs:choice>
      <xs:element ref="xacml:SubjectAttributeDesignator" />
      <xs:element ref="xacml:AttributeSelector" />
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required" />
</xs:complexType>
```

The <SubjectMatch> element is of **SubjectMatchType** complex type.

The <SubjectMatch> element contains the following attributes and elements:

MatchId [Required]

Specifies a matching function. The value of this attribute MUST be of type **xs:anyURI** with legal values documented in Section 7.5.

<xacml:AttributeValue> [Required]

Embedded attribute value.

<SubjectAttributeDesignator> [Required choice]

MAY be used to identify one or more **attribute** values in a <Subject> element of the request **context**.

<AttributeSelector> [Required choice]

MAY be used to identify one or more **attribute** values in the request **context**. The XPath expression SHOULD resolve to an **attribute** in a <Subject> element of the request **context**.

Deleted: cd-03



## 5.9. Element <Resources>

The <Resources> element SHALL contain a **disjunctive sequence** of <Resource> elements.

```
<xs:element name="Resources" type="xacml:ResourcesType" />
<xs:complexType name="ResourcesType">
  <xs:sequence>
    <xs:element ref="xacml:Resource" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
```

The <Resources> element is of **ResourcesType** complex type.

The <Resources> element contains the following elements:

<Resource> [One to Many, Required]

See Section 5.10.

## 5.10. Element <Resource>

The <Resource> element SHALL contain a **conjunctive sequence** of <ResourceMatch> elements.

```
<xs:element name="Resource" type="xacml:ResourceType" />
<xs:complexType name="ResourceType">
  <xs:sequence>
    <xs:element ref="xacml:ResourceMatch" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
```

The <Resource> element is of **ResourceType** complex type.

The <Resource> element contains the following elements:

<ResourceMatch> [One to Many]

A **conjunctive sequence** of individual matches of the **resource attributes** in the request **context** and the embedded **attribute** values. See Section 5.11.

## 5.11. Element <ResourceMatch>

The <ResourceMatch> element SHALL identify a set of **resource**-related entities by matching **attribute** values in the <xacml-context:Resource> element of the request **context** with the embedded **attribute** value.

```
<xs:element name="ResourceMatch" type="xacml:ResourceMatchType" />
<xs:complexType name="ResourceMatchType">
  <xs:sequence>
    <xs:element ref="xacml:AttributeValue" />
    <xs:choice>
      <xs:element ref="xacml:ResourceAttributeDesignator" />
      <xs:element ref="xacml:AttributeSelector" />
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required" />
</xs:complexType>
```

The <ResourceMatch> element is of **ResourceMatchType** complex type.

The <ResourceMatch> element contains the following attributes and elements:

Deleted: cd-03

access\_control-xacml-2.0-core-spec-[cd-04](#)

2008 MatchId [Required]

2009 Specifies a matching function. Values of this attribute MUST be of type **xs:anyURI**, with  
2010 legal values documented in Section 7.5.

2011 <xacml:AttributeValue> [Required]

2012 Embedded attribute value.

2013 <ResourceAttributeDesignator> [Required Choice]

2014 MAY be used to identify one or more **attribute** values in the <Resource> element of the  
2015 request **context**.

2016 <AttributeSelector> [Required Choice]

2017 MAY be used to identify one or more **attribute** values in the request **context**. The XPath  
2018 expression SHOULD resolve to an **attribute** in the <Resource> element of the request  
2019 **context**.

## 2020 5.12. Element <Actions>

2021 The <Actions> element SHALL contain a **disjunctive sequence** of <Action> elements.

```
2022 <xs:element name="Actions" type="xacml:ActionTypes" />
2023 <xs:complexType name="ActionTypes">
2024   <xs:sequence>
2025     <xs:element ref="xacml:Action" maxOccurs="unbounded" />
2026   </xs:sequence>
2027 </xs:complexType>
```

2028 The <Actions> element is of **ActionTypes** complex type.

2029 The <Actions> element contains the following elements:

2030 <Action> [One to Many, Required]

2031 See Section 5.13.

## 2032 5.13. Element <Action>

2033 The <Action> element SHALL contain a **conjunctive sequence** of <ActionMatch> elements.

```
2034 <xs:element name="Action" type="xacml:ActionType" />
2035 <xs:complexType name="ActionType">
2036   <xs:sequence>
2037     <xs:element ref="xacml:ActionMatch" maxOccurs="unbounded" />
2038   </xs:sequence>
2039 </xs:complexType>
```

2040 The <Action> element is of **ActionType** complex type.

2041 The <Action> element contains the following elements:

2042 <ActionMatch> [One to Many]

2043 A **conjunctive sequence** of individual matches of the **action attributes** in the request  
2044 **context** and the embedded **attribute** values. See Section 5.14.

Deleted: cd-03

## 5.14. Element <ActionMatch>

The <ActionMatch> element SHALL identify a set of **action**-related entities by matching **attribute** values in the <xacml-context:Action> element of the request **context** with the embedded **attribute** value.

```
<xs:element name="ActionMatch" type="xacml:ActionMatchType"/>
<xs:complexType name="ActionMatchType">
  <xs:sequence>
    <xs:element ref="xacml:AttributeValue"/>
    <xs:choice>
      <xs:element ref="xacml:ActionAttributeDesignator"/>
      <xs:element ref="xacml:AttributeSelector"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
</xs:complexType>
```

The <ActionMatch> element is of **ActionMatchType** complex type.

The <ActionMatch> element contains the following attributes and elements:

**MatchId** [Required]

Specifies a matching function. The value of this attribute MUST be of type **xs:anyURI**, with legal values documented in Section 7.5.

<xacml:AttributeValue> [Required]

Embedded attribute value.

<ActionAttributeDesignator> [Required Choice]

MAY be used to identify one or more **attribute** values in the <Action> element of the request **context**.

<AttributeSelector> [Required Choice]

MAY be used to identify one or more **attribute** values in the request **context**. The XPath expression SHOULD resolve to an **attribute** in the <Action> element of the **context**.

## 5.15. Element <Environments>

The <Environments> element SHALL contain a **disjunctive sequence** of <Environment> elements.

```
<xs:element name="Environments" type="xacml:EnvironmentsType"/>
<xs:complexType name="EnvironmentsType">
  <xs:sequence>
    <xs:element ref="xacml:Environment" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

The <Environments> element is of **EnvironmentsType** complex type.

The <Environments> element contains the following elements:

<Environment> [One to Many, Required]

See Section 5.16.

Deleted: cd-03

## 5.16. Element <Environment>

The <Environment> element SHALL contain a **conjunctive sequence** of <EnvironmentMatch> elements.

```
<xs:element name="Environment" type="xacml:EnvironmentType"/>
<xs:complexType name="EnvironmentType">
  <xs:sequence>
    <xs:element ref="xacml:EnvironmentMatch" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

The <Environment> element is of **EnvironmentType** complex type.

The <Environment> element contains the following elements:

<EnvironmentMatch> [One to Many]

A **conjunctive sequence** of individual matches of the **environment** attributes in the request **context** and the embedded **attribute** values.

## 5.17. Element <EnvironmentMatch>

The <EnvironmentMatch> element SHALL identify an environment by matching **attribute** values in the <xacml-context:Environment> element of the request **context** with the embedded **attribute** value.

```
<xs:element name="EnvironmentMatch" type="xacml:EnvironmentMatchType"/>
<xs:complexType name="EnvironmentMatchType">
  <xs:sequence>
    <xs:element ref="xacml:AttributeValue"/>
    <xs:choice>
      <xs:element ref="xacml:EnvironmentAttributeDesignator"/>
      <xs:element ref="xacml:AttributeSelector"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute name="MatchId" type="xs:anyURI" use="required"/>
</xs:complexType>
```

The <EnvironmentMatch> element is of **EnvironmentMatchType** complex type.

The <EnvironmentMatch> element contains the following attributes and elements:

MatchId [Required]

Specifies a matching function. The value of this attribute MUST be of type **xs:anyURI**, with legal values documented in Section A.3.

<xacml:AttributeValue> [Required]

Embedded attribute value.

<EnvironmentAttributeDesignator> [Required Choice]

MAY be used to identify one or more **attribute** values in the <Environment> element of the request **context**.

<AttributeSelector> [Required Choice]

MAY be used to identify one or more **attribute** values in the request **context**. The XPath expression SHOULD resolve to an **attribute** in the <Environment> element of the request **context**.

Deleted: cd-03

## 5.18. Element <PolicySetIdReference>

The <PolicySetIdReference> element SHALL be used to reference a <PolicySet> element by id. If <PolicySetIdReference> is a URL, then it MAY be resolvable to the <PolicySet> element. However, the mechanism for resolving a **policy set** reference to the corresponding **policy set** is outside the scope of this specification.

```
<xs:element name="PolicySetIdReference" type="xacml:IdReferenceType" />
xs:complexType name="IdReferenceType">
  <xs:simpleContent>
    <xs:extension base="xs:anyURI">
      <xs:attribute name="xacml:Version" type="xacml:VersionMatchType"
use="optional" />
      <xs:attribute name="xacml:EarliestVersion" type="xacml:VersionMatchType"
use="optional" />
      <xs:attribute name="xacml:LatestVersion" type="xacml:VersionMatchType"
use="optional" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

Element <PolicySetIdReference> is of **xacml:IdReferenceType** complex type.

**IdReferenceType** extends the **xs:anyURI** type with the following attributes:

Version [Optional]

Specifies a matching expression for the version of the **policy set** referenced.

EarliestVersion [Optional]

Specifies a matching expression for the earliest acceptable version of the **policy set** referenced.

LatestVersion [Optional]

Specifies a matching expression for the latest acceptable version of the **policy set** referenced.

The matching operation is defined in Section 5.21. Any combination of these attributes MAY be present in a <PolicySetIdReference>. The referenced **policy set** MUST match all expressions. If none of these attributes is present, then any version of the **policy set** is acceptable. In the case that more than one matching version can be obtained, then the most recent one SHOULD be used.

## 5.19. Element <PolicyIdReference>

The <xacml:PolicyIdReference> element SHALL be used to reference a <Policy> element by id. If <PolicyIdReference> is a URL, then it MAY be resolvable to the <Policy> element. However, the mechanism for resolving a **policy** reference to the corresponding **policy** is outside the scope of this specification.

```
<xs:element name="PolicyIdReference" type="xacml:IdReferenceType" />
```

Element <PolicyIdReference> is of **xacml:IdReferenceType** complex type (see Section 5.18).

## 5.20. Simple type VersionType

Elements of this type SHALL contain the version number of the **policy** or **policy set**.

```
<xs:simpleType name="VersionType">
```

Deleted: cd-03

access\_control-xacml-2.0-core-spec-[cd-04](#)

53

```

2172 <xs:restriction base="xs:string">
2173   <xs:pattern value="(\d+\.)*\d+"/>
2174 </xs:restriction>
2175 </xs:simpleType>

```

2176 The version number is expressed as a sequence of decimal numbers, each separated by a period  
 2177 (.). 'd+' represents a sequence of one or more decimal digits.

## 2178 5.21. Simple type VersionMatchType

2179 Elements of this type SHALL contain a restricted regular expression matching a version number  
 2180 (see Section 5.20). The expression SHALL match versions of a referenced **policy** or **policy set**  
 2181 that are acceptable for inclusion in the referencing **policy** or **policy set**.

```

2182 <xs:simpleType name="VersionMatchType">
2183   <xs:restriction base="xs:string">
2184     <xs:pattern value="((\d+|\*)\.)*(\d+|\*|\+)" />
2185   </xs:restriction>
2186 </xs:simpleType>

```

2187 A version match is '.'-separated, like a version string. A number represents a direct numeric match.  
 2188 A '\*' means that any single number is valid. A '+' means that any number, and any subsequent  
 2189 numbers, are valid. In this manner, the following four patterns would all match the version string  
 2190 '1.2.3': '1.2.3', '1.\*.3', '1.2.\*' and '1.+'

## 2191 5.22. Element <Policy>

2192 The <Policy> element is the smallest entity that SHALL be presented to the **PDP** for evaluation.

2193 A <Policy> element MAY be evaluated, in which case the evaluation procedure defined in  
 2194 Section 7.10 SHALL be used.

2195 The main components of this element are the <Target>, <Rule>, <CombinerParameters>,  
 2196 <RuleCombinerParameters> and <Obligations> elements and the RuleCombiningAlgId  
 2197 attribute.

2198 The <Target> element defines the applicability of the <Policy> element to a set of **decision**  
 2199 **requests**. If the <Target> element within the <Policy> element matches the **request context**,  
 2200 then the <Policy> element MAY be used by the **PDP** in making its **authorization decision**. See  
 2201 Section 7.10.

2202 The <Policy> element includes a sequence of choices between <VariableDefinition> and  
 2203 <Rule> elements.

2204 **Rules** included in the <Policy> element MUST be combined by the algorithm specified by the  
 2205 RuleCombiningAlgId attribute.

2206 The <Obligations> element contains a set of **obligations** that MUST be fulfilled by the **PEP** in  
 2207 conjunction with the **authorization decision**.

```

2208 <xs:element name="Policy" type="xacml:PolicyType" />
2209 <xs:complexType name="PolicyType">
2210   <xs:sequence>
2211     <xs:element ref="xacml:Description" minOccurs="0"/>
2212     <xs:element ref="xacml:PolicyDefaults" minOccurs="0"/>
2213     <xs:element ref="xacml:CombinerParameters" minOccurs="0"/>
2214     <xs:element ref="xacml:Target"/>
2215     <xs:choice maxOccurs="unbounded">
2216       <xs:element ref="xacml:CombinerParameters" minOccurs="0"/>
2217       <xs:element ref="xacml:RuleCombinerParameters" minOccurs="0"/>
2218       <xs:element ref="xacml:VariableDefinition"/>

```

Deleted: cd-03

```

2219     <xs:element ref="xacml:Rule"/>
2220   </xs:choice>
2221   <xs:element ref="xacml:Obligations" minOccurs="0"/>
2222 </xs:sequence>
2223 <xs:attribute name="PolicyId" type="xs:anyURI" use="required"/>
2224 <xs:attribute name="Version" type="xacml:VersionType" default="1.0"/>
2225 <xs:attribute name="RuleCombiningAlgId" type="xs:anyURI" use="required"/>
2226 </xs:complexType>
2227 The <Policy> element is of PolicyType complex type.
2228 The <Policy> element contains the following attributes and elements:
2229 PolicyId [Required]
2230     Policy identifier. It is the responsibility of the PAP to ensure that no two policies visible to
2231     the PDP have the same identifier. This MAY be achieved by following a predefined URN or
2232     URI scheme. If the policy identifier is in the form of a URL, then it MAY be resolvable.
2233 Version [Default 1.0]
2234     The version number of the Policy.
2235 RuleCombiningAlgId [Required]
2236     The identifier of the rule-combining algorithm by which the <Policy>,
2237     <CombinerParameters> and <RuleCombinerParameters> components MUST be
2238     combined. Standard rule-combining algorithms are listed in Appendix C. Standard rule-
2239     combining algorithm identifiers are listed in Section B.10.
2240 <Description> [Optional]
2241     A free-form description of the policy. See Section 5.2.
2242 <PolicyDefaults> [Optional]
2243     Defines a set of default values applicable to the policy. The scope of the
2244     <PolicyDefaults> element SHALL be the enclosing policy.
2245 <CombinerParameters> [Optional]
2246     A sequence of parameters to be used by the rule-combining algorithm.
2247 <RuleCombinerParameters> [Optional]
2248     A sequence of parameters to be used by the rule-combining algorithm.
2249 <Target> [Required]
2250     The <Target> element defines the applicability of a <Policy> to a set of decision requests.
2251     The <Target> element MAY be declared by the creator of the <Policy> element, or it
2252     MAY be computed from the <Target> elements of the referenced <Rule> elements either
2253     as an intersection or as a union.
2254 <VariableDefinition> [Any Number]
2255     Common function definitions that can be referenced from anywhere in a rule where an
2256     expression can be found.
2257 <Rule> [Any Number]

```

Deleted: cd-03

2258 A sequence of **rules** that MUST be combined according to the RuleCombiningAlgId  
2259 attribute. **Rules** whose <Target> elements match the **decision request** MUST be  
2260 considered. **Rules** whose <Target> elements do not match the **decision request** SHALL  
2261 be ignored.

2262 <Obligations> [Optional]

2263 A **conjunctive sequence** of **obligations** that MUST be fulfilled by the **PEP** in conjunction  
2264 with the **authorization decision**. See Section 7.14 for a description of how the set of  
2265 **obligations** to be returned by the **PDP** SHALL be determined.

### 2266 5.23. Element <PolicyDefaults>

2267 The <PolicyDefaults> element SHALL specify default values that apply to the <Policy>  
2268 element.

```
2269 <xs:element name="PolicyDefaults" type="xacml:DefaultsType"/>
2270 <xs:complexType name="DefaultsType">
2271   <xs:sequence>
2272     <xs:choice>
2273       <xs:element ref="xacml:XPathVersion" minOccurs="0"/>
2274     </xs:choice>
2275   </xs:sequence>
2276 </xs:complexType>
```

2277 <PolicyDefaults> element is of **DefaultsType** complex type.

2278 The <PolicyDefaults> element contains the following elements:

2279 <XPathVersion> [Optional]

2280 Default XPath version.

### 2281 5.24. Element <CombinerParameters>

2282 The <CombinerParameters> element conveys parameters for a **policy-** or **rule-combining**  
2283 **algorithm**.

2284 If multiple <CombinerParameters> elements occur within the same **policy** or **policy set**, they  
2285 SHALL be considered equal to one <CombinerParameters> element containing the  
2286 concatenation of all the sequences of <CombinerParameters> contained in all the aforementioned  
2287 <CombinerParameters> elements, such that the order of occurrence of the  
2288 <CominberParameters> elements is preserved in the concatenation of the  
2289 <CombinerParameter> elements.

2290 Note that none of the **combining algorithms** specified in XACML 2.0 is parameterized.

```
2291 <xs:element name="CombinerParameters" type="xacml:CombinerParametersType"/>
2292 <xs:complexType name="CombinerParametersType">
2293   <xs:sequence>
2294     <xs:element ref="xacml:CombinerParameter" minOccurs="0"
2295     maxOccurs="unbounded"/>
2296   </xs:sequence>
2297 </xs:complexType>
```

2298 The <CombinerParameters> element is of **CombinerParametersType** complex type.

2299 The <CombinerParameters> element contains the following elements:

2300 <CombinerParameter> [Any Number]

Deleted: cd-03



2301 A single parameter. See Section 5.25.  
2302 Support for the <CombinerParameters> element is optional.

2303 **5.25. Element <CombinerParameter>**

2304 The <CombinerParameter> element conveys a single parameter for a *policy*- or *rule-*  
2305 *combining algorithm*.

```
2306 <xs:element name="CombinerParameter" type="xacml:CombinerParameterType" />  
2307 <xs:complexType name="CombinerParameterType">  
2308   <xs:sequence>  
2309     <xs:element ref="xacml:AttributeValue" />  
2310   </xs:sequence>  
2311   <xs:attribute name="ParameterName" type="xs:string" use="required" />  
2312 </xs:complexType>
```

2313 The <CombinerParameter> element is of **CombinerParameterType** complex type.

2314 The <CombinerParameter> element contains the following attribute:

2315 ParameterName [Required]

2316 The identifier of the parameter.

2317 AttributeValue [Required]

2318 The value of the parameter.

2319 Support for the <CombinerParameter> element is optional.

2320 **5.26. Element <RuleCombinerParameters>**

2321 The <RuleCombinerParameters> element conveys *parameters* associated with a particular  
2322 *rule* within a *policy* for a *rule-combining algorithm*.

2323 Each <RuleCombinerParameters> element MUST be associated with a *rule* contained within  
2324 the same *policy*. If multiple <RuleCombinerParameters> elements reference the same *rule*,  
2325 they SHALL be considered equal to one <RuleCombinerParameters> element containing the  
2326 concatenation of all the sequences of <CombinerParameters> contained in all the aforementioned  
2327 <RuleCombinerParameters> elements, such that the order of occurrence of the  
2328 <RuleCominberParameters> elements is preserved in the concatenation of the  
2329 <CombinerParameter> elements.

2330 Note that none of the *rule-combining algorithms* specified in XACML 2.0 is parameterized.

```
2331 <xs:element name="RuleCombinerParameters"  
2332 type="xacml:RuleCombinerParametersType" />  
2333 <xs:complexType name="RuleCombinerParametersType">  
2334   <xs:complexContent>  
2335     <xs:extension base="xacml:CombinerParametersType">  
2336       <xs:attribute name="RuleIdRef" type="xs:string" use="required" />  
2337     </xs:extension>  
2338   </xs:complexContent>  
2339 </xs:complexType>
```

2340 The <RuleCombinerParameters> element contains the following elements:

2341 RuleIdRef [Required]

2342 The identifier of the <Rule> contained in the *policy*.

Deleted: cd-03

2343 Support for the <RuleCombinerParameters> element is optional, only if support for **combiner**  
2344 **parameters** is optional.

## 2345 5.27. Element <PolicyCombinerParameters>

2346 The <PolicyCombinerParameters> element conveys **parameters** associated with a particular  
2347 **policy** within a **policy set** for a **policy-combining algorithm**.

2348 Each <PolicyCombinerParameters> element MUST be associated with a **policy** contained  
2349 within the same **policy set**. If multiple <PolicyCombinerParameters> elements reference the  
2350 same **policy**, they SHALL be considered equal to one <PolicyCombinerParameters> element  
2351 containing the concatenation of all the sequences of <CombinerParameters> contained in all the  
2352 aforementioned <PolicyCombinerParameters> elements, such that the order of occurrence of  
2353 the <PolicyCombinerParameters> elements is preserved in the concatenation of the  
2354 <CombinerParameter> elements.

2355 Note that none of the **policy-combining algorithms** specified in XACML 2.0 is parameterized.

```
2356 <xs:element name="PolicyCombinerParameters"  
2357 type="xacml:PolicyCombinerParametersType" />  
2358 <xs:complexType name="PolicyCombinerParametersType">  
2359   <xs:complexContent>  
2360     <xs:extension base="xacml:CombinerParametersType">  
2361       <xs:attribute name="PolicyIdRef" type="xs:anyURI" use="required" />  
2362     </xs:extension>  
2363   </xs:complexContent>  
2364 </xs:complexType>
```

2365 The <PolicyCombinerParameters> element is of **PolicyCombinerParametersType** complex  
2366 type.

2367 The <PolicyCombinerParameters> element contains the following elements:

2368 PolicyIdRef [Required]

2369       The identifier of a <Policy> or the value of a <PolicyIdReference> contained in the  
2370 **policy set**.

2371 Support for the <PolicyCombinerParameters> element is optional, only if support for  
2372 **combiner parameters** is optional.

## 2373 5.28. Element <PolicySetCombinerParameters>

2374 The <PolicySetCombinerParameters> element conveys **parameters** associated with a  
2375 particular **policy set** within a **policy set** for a **policy-combining algorithm**.

2376 Each <PolicySetCombinerParameters> element MUST be associated with a **policy set**  
2377 contained within the same **policy set**. If multiple <PolicySetCombinerParameters> elements  
2378 reference the same **policy set**, they SHALL be considered equal to one  
2379 <PolicySetCombinerParameters> element containing the concatenation of all the sequences  
2380 of <CombinerParameters> contained in all the aforementioned  
2381 <PolicySetCombinerParameters> elements, such that the order of occurrence of the  
2382 <PolicySetCombinerParameters> elements is preserved in the concatenation of the  
2383 <CombinerParameter> elements.

2384 Note that none of the **policy-combining algorithms** specified in XACML 2.0 is parameterized.

```
2385 <xs:element name="PolicySetCombinerParameters"  
2386 type="xacml:PolicySetCombinerParametersType" />
```

Deleted: cd-03

access\_control-xacml-2.0-core-spec-[cd-04](#)

58

```

2387 <xs:complexType name="PolicySetCombinerParametersType">
2388   <xs:complexContent>
2389     <xs:extension base="xacml:CombinerParametersType">
2390       <xs:attribute name="PolicySetIdRef" type="xs:anyURI" use="required"/>
2391     </xs:extension>
2392   </xs:complexContent>
2393 </xs:complexType>

```

2394 The <PolicySetCombinerParameters> element is of **PolicySetCombinerParametersType**

2395 complex type.

2396 The <PolicySetCombinerParameters> element contains the following elements:

2397 PolicySetIdRef [Required]

2398       The identifier of a <PolicySet> or the value of a <PolicySetIdReference> contained

2399       in the **policy set**.

2400 Support for the <PolicySetCombinerParameters> element is optional, only if support for

2401 **combiner parameters** is optional.

## 2402 5.29. Element <Rule>

2403 The <Rule> element SHALL define the individual **rules** in the **policy**. The main components of

2404 this element are the <Target> and <Condition> elements and the Effect attribute.

2405 A <Rule> element MAY be evaluated, in which case the evaluation procedure defined in Section

2406 7.9 SHALL be used.

```

2407 <xs:element name="Rule" type="xacml:RuleType"/>
2408 <xs:complexType name="RuleType">
2409   <xs:sequence>
2410     <xs:element ref="xacml:Description" minOccurs="0"/>
2411     <xs:element ref="xacml:Target" minOccurs="0"/>
2412     <xs:element ref="xacml:Condition" minOccurs="0"/>
2413   </xs:sequence>
2414   <xs:attribute name="RuleId" type="xs:string" use="required"/>
2415   <xs:attribute name="Effect" type="xacml:EffectType" use="required"/>
2416 </xs:complexType>

```

2417 The <Rule> element is of **RuleType** complex type.

2418 The <Rule> element contains the following attributes and elements:

2419 RuleId [Required]

2420       A string identifying this **rule**.

2421 Effect [Required]

2422       **Rule effect.** The value of this attribute is either "Permit" or "Deny".

2423 <Description> [Optional]

2424       A free-form description of the **rule**.

2425 <Target> [Optional]

2426       Identifies the set of **decision requests** that the <Rule> element is intended to evaluate. If

2427       this element is omitted, then the **target** for the <Rule> SHALL be defined by the

2428       <Target> element of the enclosing <Policy> element. See Section 7.6 for details.

Deleted: cd-03

2429 <Condition> [Optional]

2430 A **predicate** that MUST be satisfied for the **rule** to be assigned its **Effect** value.

### 2431 5.30. Simple type **EffectType**

2432 The **EffectType** simple type defines the values allowed for the **Effect** attribute of the <Rule>  
2433 element and for the **FulfillOn** attribute of the <Obligation> element.

```
2434 <xs:simpleType name="EffectType">  
2435   <xs:restriction base="xs:string">  
2436     <xs:enumeration value="Permit"/>  
2437     <xs:enumeration value="Deny"/>  
2438   </xs:restriction>  
2439 </xs:simpleType>
```

### 2440 5.31. Element <VariableDefinition>

2441 The <VariableDefinition> element SHALL be used to define a value that can be referenced  
2442 by a <VariableReference> element. The name supplied for its **VariableId** attribute SHALL  
2443 NOT occur in the **VariableId** attribute of any other <VariableDefinition> element within the  
2444 encompassing **policy**. The <VariableDefinition> element MAY contain undefined  
2445 <VariableReference> element, but if it does, a corresponding <VariableDefinition> element  
2446 MUST be defined later in the encompassing **policy**. <VariableDefinition> elements MAY be  
2447 grouped together or MAY be placed close to the reference in the encompassing **policy**. There  
2448 MAY be zero or more references to each <VariableDefinition> element.

```
2449 <xs:element name="VariableDefinition" type="xacml:VariableDefinitionType"/>  
2450 <xs:complexType name="VariableDefinitionType">  
2451   <xs:sequence>  
2452     <xs:element ref="xacml:Expression"/>  
2453   </xs:sequence>  
2454   <xs:attribute name="VariableId" type="xs:string" use="required"/>  
2455 </xs:complexType>
```

2456 The <VariableDefinition> element is of **VariableDefinitionType** complex type. The  
2457 <VariableDefinition> element has the following elements and attributes:

2458 <Expression> [Required]

2459 Any element of **ExpressionType** complex type.

2460 **VariableId** [Required]

2461 The name of the variable definition.

### 2462 5.32. Element <VariableReference>

2463 The <VariableReference> element is used to reference a value defined within the same  
2464 encompassing <Policy> element. The <VariableReference> element SHALL refer to the  
2465 <VariableDefinition> element by string equality on the value of their respective **VariableId**  
2466 attributes. There SHALL exist one and only one <VariableDefinition> within the same  
2467 encompassing <Policy> element to which the <VariableReference> refers. There MAY be  
2468 zero or more <VariableReference> elements that refer to the same <VariableDefinition>  
2469 element.

```
2470 <xs:element name="VariableReference" type="xacml:VariableReferenceType"  
2471 substitutionGroup="xacml:Expression"/>  
2472 <xs:complexType name="VariableReferenceType">
```

Deleted: cd-03

access\_control-xacml-2.0-core-spec-[cd-04](#)

60

```

2473 <xs:complexContent>
2474   <xs:extension base="xacml:ExpressionType">
2475     <xs:attribute name="VariableId" type="xs:string" use="required" />
2476   </xs:extension>
2477 </xs:complexContent>
2478 </xs:complexType>

```

2479 The <VariableReference> element is of the **VariableReferenceType** complex type, which is of  
 2480 the **ExpressionType** complex type and is a member of the <Expression> element substitution  
 2481 group. The <VariableReference> element MAY appear any place where an <Expression>  
 2482 element occurs in the schema.

2483 The <VariableReference> element has the following attributes:

2484 VariableId [Required]

2485       The name used to refer to the value defined in a <VariableDefinition> element.

### 2486 5.33. Element <Expression>

2487 The <Expression> element is not used directly in a **policy**. The <Expression> element  
 2488 signifies that an element that extends the **ExpressionType** and is a member of the  
 2489 <Expression> element substitution group SHALL appear in its place.

```

2490 <xs:element name="Expression" type="xacml:ExpressionType" abstract="true" />
2491 <xs:complexType name="ExpressionType" abstract="true" />

```

2492 The following elements are in the <Expression> element substitution group:

2493 <Apply>, <AttributeSelector>, <AttributeValue>, <Function>,  
 2494 <VariableReference>, <ActionAttributeDesignator>,  
 2495 <ResourceAttributeDesignator>, <SubjectAttributeDesignator> and  
 2496 <EnvironmentAttributeDesignator>.

### 2497 5.34. Element <Condition>

2498 The <Condition> element is a Boolean function over **subject**, **resource**, **action** and  
 2499 **environment attributes** or functions of **attributes**.

```

2500 <xs:element name="Condition" type="xacml:ConditionType" />
2501 <xs:complexType name="ConditionType">
2502   <xs:sequence>
2503     <xs:element ref="xacml:Expression" />
2504   </xs:sequence>
2505 </xs:complexType>

```

2506 The <Condition> contains one <Expression> element, with the restriction that the  
 2507 <Expression> return data-type MUST be "http://www.w3.org/2001/XMLSchema#boolean".  
 2508 Evaluation of the <Condition> element is described in Section 7.8.

### 2509 5.35. Element <Apply>

2510 The <Apply> element denotes application of a function to its arguments, thus encoding a function  
 2511 call. The <Apply> element can be applied to any combination of the members of the  
 2512 <Expression> element substitution group. See Section 5.33.

```

2513 <xs:element name="Apply" type="xacml:ApplyType"
2514   substitutionGroup="xacml:Expression" />
2515 <xs:complexType name="ApplyType">
2516   <xs:complexContent>

```

Deleted: cd-03

```

2517     <xs:extension base="xacml:ExpressionType">
2518       <xs:sequence>
2519         <xs:element ref="xacml:Expression" minOccurs="0"
2520 maxOccurs="unbounded" />
2521       </xs:sequence>
2522       <xs:attribute name="FunctionId" type="xs:anyURI" use="required" />
2523     </xs:extension>
2524   </xs:complexContent>
2525 </xs:complexType>

```

The <Apply> element is of **ApplyType** complex type.

The <Apply> element contains the following attributes and elements:

FunctionId [Required]

The identifier of the function to be applied to the arguments. XACML-defined functions are described in Appendix A.

<Expression> [Optional]

Arguments to the function, which may include other functions.

### 5.36. Element <Function>

The <Function> element SHALL be used to name a function as an argument to the function defined by the parent <Apply> element. In the case where the parent <Apply> element is a higher-order **bag** function, the named function is applied to every element of the **bag** or **bags** identified in the other arguments of the parent element. The higher-order **bag** functions are described in Section A3A.3.12.

```

2539 <xs:element name="Function" type="xacml:FunctionType"
2540 substitutionGroup="xacml:Expression" />
2541 <xs:complexType name="FunctionType">
2542   <xs:complexContent>
2543     <xs:extension base="xacml:ExpressionType">
2544       <xs:attribute name="FunctionId" type="xs:anyURI" use="required" />
2545     </xs:extension>
2546   </xs:complexContent>
2547 </xs:complexType>

```

The Function element is of **FunctionType** complex type.

The Function element contains the following attributes:

FunctionId [Required]

The identifier of the function.

### 5.37. Complex type AttributeDesignatorType

The **AttributeDesignatorType** complex type is the type for elements that identify **attributes** by name. It contains the information required to match **attributes** in the request **context**. See Section 7.2.4.

It also contains information to control behaviour in the event that no matching **attribute** is present in the **context**.

Elements of this type SHALL NOT alter the match semantics of **named attributes**, but MAY narrow the search space.

```

2560 <xs:complexType name="AttributeDesignatorType">

```

access\_control-xacml-2.0-core-spec-[cd-04](#)

Deleted: cd-03

```

2561 <xs:complexContent>
2562   <xs:extension base="xacml:ExpressionType">
2563     <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
2564     <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
2565     <xs:attribute name="Issuer" type="xs:string" use="optional"/>
2566     <xs:attribute name="MustBePresent" type="xs:boolean" use="optional"
2567 default="false"/>
2568   </xs:extension>
2569 </xs:complexContent>
2570 </xs:complexType>

```

A **named attribute** SHALL match an **attribute** if the values of their respective **AttributeId**, **DataType** and **Issuer** attributes match. The **attribute** designator's **AttributeId** MUST match, by URI equality, the **AttributeId** of the **attribute**. The **attribute** designator's **DataType** MUST match, by URI equality, the **DataType** of the same **attribute**.

If the **Issuer** attribute is present in the **attribute** designator, then it MUST match, using the "urn:oasis:names:tc:xacml:1.0:function:string-equal" function, the **Issuer** of the same **attribute**. If the **Issuer** is not present in the **attribute** designator, then the matching of the **attribute** to the **named attribute** SHALL be governed by **AttributeId** and **DataType** attributes alone.

The <AttributeDesignatorType> contains the following attributes:

**AttributeId** [Required]

This attribute SHALL specify the **AttributeId** with which to match the **attribute**.

**DataType** [Required]

The bag returned by the <AttributeDesignator> element SHALL contain values of this data-type.

**Issuer** [Optional]

This attribute, if supplied, SHALL specify the **Issuer** with which to match the **attribute**.

**MustBePresent** [Optional]

This attribute governs whether the element returns "Indeterminate" or an empty **bag** in the event the **named attribute** is absent from the request **context**. See Section 7.2.5. Also see Sections 7.15.2 and 7.15.3.

### 5.38. Element <SubjectAttributeDesignator>

The <SubjectAttributeDesignator> element retrieves a **bag** of values for a **named** categorized **subject attribute** from the request **context**. A **subject attribute** is an **attribute** contained within a <Subject> element of the request **context**. A categorized **subject** is a **subject** that is identified by a particular **subject-category** attribute. A **named categorized subject attribute** is a **named subject attribute** for a particular **categorized subject**.

The <SubjectAttributeDesignator> element SHALL return a **bag** containing all the **subject attribute** values that are matched by the **named categorized subject attribute**. In the event that no matching attribute is present in the context, the **MustBePresent** attribute governs whether this element returns an empty **bag** or "Indeterminate". See Section 7.2.5.

The **SubjectAttributeDesignatorType** extends the match semantics of the **AttributeDesignatorType** (See Section 5.37) such that it narrows the **attribute** search space to the specific **categorized subject** such that the value of this element's **SubjectCategory** attribute

Deleted: cd-03



2604 matches, by URI equality, the value of the request **context's** <Subject> element's  
2605 SubjectCategory attribute.

2606 If the request context contains multiple **subjects** with the same SubjectCategory XML attribute,  
2607 then they SHALL be treated as if they were one *categorized subject*.

2608 The <SubjectAttributeDesignator> MAY appear in the <SubjectMatch> element and  
2609 MAY be passed to the <Apply> element as an argument.

```
2610 <xs:element name="SubjectAttributeDesignator"  
2611 type="xacml:SubjectAttributeDesignatorType"  
2612 substitutionGroup="xacml:Expression"/>  
2613 <xs:complexType name="SubjectAttributeDesignatorType">  
2614   <xs:complexContent>  
2615     <xs:extension base="xacml:AttributeDesignatorType">  
2616       <xs:attribute name="SubjectCategory" type="xs:anyURI" use="optional"  
2617       default="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"/>  
2618     </xs:extension>  
2619   </xs:complexContent>  
2620 </xs:complexType>
```

2621 The <SubjectAttributeDesignator> element is of type **SubjectAttributeDesignatorType**.  
2622 The **SubjectAttributeDesignatorType** complex type extends the **AttributeDesignatorType**  
2623 complex type with a SubjectCategory attribute.

2624 SubjectCategory [Optional]

2625 This attribute SHALL specify the *categorized subject* from which to match *named subject*  
2626 **attributes**. If SubjectCategory is not present, then its default value of  
2627 "urn:oasis:names:tc:xacml:1.0:subject-category:access-subject" SHALL be used. Standard  
2628 values of the SubjectCategory are listed in Section B.2.

## 2629 5.39. Element <ResourceAttributeDesignator>

2630 The <ResourceAttributeDesignator> element retrieves a **bag** of values for a *named*  
2631 **resource attribute** from the request **context**. A **resource attribute** is an **attribute** contained  
2632 within the <Resource> element of the request **context**. A *named resource attribute* is a *named*  
2633 **attribute** that matches a **resource attribute**. A *named resource attribute* SHALL be considered  
2634 **present** if there is at least one **resource attribute** that matches the criteria set out below. A  
2635 **resource attribute** value is an **attribute** value that is contained within a **resource attribute**.

2636 The <ResourceAttributeDesignator> element SHALL return a **bag** containing all the  
2637 **resource attribute** values that are matched by the *named resource attribute*. In the event that no  
2638 matching attribute is present in the context, the MustBePresent attribute governs whether this  
2639 element returns an empty **bag** or "Indeterminate". See Section 7.2.5.

2640 A *named resource attribute* SHALL match a **resource attribute** as per the match semantics  
2641 specified in the **AttributeDesignatorType** complex type. See Section 5.37.

2642 The <ResourceAttributeDesignator> MAY appear in the <ResourceMatch> element and  
2643 MAY be passed to the <Apply> element as an argument.

```
2644 <xs:element name="ResourceAttributeDesignator"  
2645 type="xacml:AttributeDesignatorType" substitutionGroup="xacml:Expression"/>
```

2646 The <ResourceAttributeDesignator> element is of the **AttributeDesignatorType** complex  
2647 type.

Deleted: cd-03



## 5.40. Element <ActionAttributeDesignator>

The <ActionAttributeDesignator> element retrieves a **bag** of values for a *named action attribute* from the request **context**. An *action attribute* is an *attribute* contained within the <Action> element of the request **context**. A *named action attribute* has specific criteria (described below) with which to match an *action attribute*. A *named action attribute* SHALL be considered *present*, if there is at least one *action attribute* that matches the criteria. An *action attribute value* is an *attribute value* that is contained within an *action attribute*.

The <ActionAttributeDesignator> element SHALL return a **bag** of all the *action attribute* values that are matched by the *named action attribute*. In the event that no matching attribute is present in the context, the `MustBePresent` attribute governs whether this element returns an empty **bag** or "Indeterminate". See Section 7.2.5.

A *named action attribute* SHALL match an *action attribute* as per the match semantics specified in the **AttributeDesignatorType** complex type. See Section 5.37.

The <ActionAttributeDesignator> MAY appear in the <ActionMatch> element and MAY be passed to the <Apply> element as an argument.

```
<xs:element name="ActionAttributeDesignator" type="xacml:AttributeDesignatorType"
substitutionGroup="xacml:Expression"/>
```

The <ActionAttributeDesignator> element is of the **AttributeDesignatorType** complex type.

## 5.41. Element <EnvironmentAttributeDesignator>

The <EnvironmentAttributeDesignator> element retrieves a **bag** of values for a *named environment attribute* from the request **context**. An *environment attribute* is an *attribute* contained within the <Environment> element of request **context**. A *named environment attribute* has specific criteria (described below) with which to match an *environment attribute*. A *named environment attribute* SHALL be considered *present*, if there is at least one *environment attribute* that matches the criteria. An *environment attribute value* is an *attribute value* that is contained within an *environment attribute*.

The <EnvironmentAttributeDesignator> element SHALL evaluate to a **bag** of all the *environment attribute* values that are matched by the *named environment attribute*. In the event that no matching attribute is present in the context, the `MustBePresent` attribute governs whether this element returns an empty **bag** or "Indeterminate". See Section 7.2.5.

A *named environment attribute* SHALL match an *environment attribute* as per the match semantics specified in the **AttributeDesignatorType** complex type. See Section 5.37.

The <EnvironmentAttributeDesignator> MAY be passed to the <Apply> element as an argument.

```
<xs:element name="EnvironmentAttributeDesignator"
type="xacml:AttributeDesignatorType" substitutionGroup="xacml:Expression"/>
```

The <EnvironmentAttributeDesignator> element is of the **AttributeDesignatorType** complex type.

## 5.42. Element <AttributeSelector>

The <AttributeSelector> element identifies attributes by their location in the request **context**. Support for the <AttributeSelector> element is OPTIONAL.

Deleted: cd-03

2690 The <AttributeSelector> element's RequestContextPath XML attribute SHALL contain a  
2691 legal XPath expression whose context node is the <xacml-context:Request> element. The  
2692 AttributeSelector element SHALL evaluate to a **bag** of values whose data-type is specified by  
2693 the element's DataType attribute. If the DataType specified in the AttributeSelector is a  
2694 primitive data type defined in [XF] or [XS], then the value returned by the XPath expression SHALL  
2695 be converted to the DataType specified in the <AttributeSelector> using the constructor  
2696 function below [XF Section 4] that corresponds to the DataType. If an error results from using the  
2697 constructor function, then the value of the <AttributeSelector> SHALL be "Indeterminate".

2698  
2699 xs:string()  
2700 xs:boolean()  
2701 xs:integer()  
2702 xs:double()  
2703 xs:dateTime()  
2704 xs:date()  
2705 xs:time()  
2706 xs:hexBinary()  
2707 xs:base64Binary()  
2708 xs:anyURI()  
2709 xf:yearMonthDuration()  
2710 xf:dayTimeDuration()  
2711

2712 If the DataType specified in the AttributeSelector is not one of the preceding primitive  
2713 DataTypes, then the AttributeSelector SHALL return a **bag** of instances of the specified  
2714 DataType. If an error occurs when converting the values returned by the XPath expression to the  
2715 specified DataType, then the result of the AttributeSelector SHALL be "Indeterminate".  
2716

2717 Each node selected by the specified XPath expression MUST be either a text node, an attribute  
2718 node, a processing instruction node or a comment node. The string representation of the value of  
2719 each node MUST be converted to an **attribute** value of the specified data-type, and the result of  
2720 the AttributeSelector is the **bag** of the **attribute** values generated from all the selected  
2721 nodes.  
2722

2723 If the node selected by the specified XPath expression is not one of those listed above (i.e. a text  
2724 node, an attribute node, a processing instruction node or a comment node), then the result of the  
2725 enclosing **policy** SHALL be "Indeterminate" with a StatusCode value of  
2726 "urn:oasis:names:tc:xacml:1.0:status:syntax-error".  
2727

```
2728 <xs:element name="AttributeSelector" type="xacml:AttributeSelectorType"
2729 substitutionGroup="xacml:Expression"/>
2730 <xs:complexType name="AttributeSelectorType">
2731   <xs:complexContent>
2732     <xs:extension base="xacml:ExpressionType">
2733       <xs:attribute name="RequestContextPath" type="xs:string" use="required"/>
2734       <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
2735       <xs:attribute name="MustBePresent" type="xs:boolean" use="optional"
2736 default="false"/>
2737     </xs:extension>
2738   </xs:complexContent>
2739 </xs:complexType>
```

2740 The <AttributeSelector> element is of **AttributeSelectorType** complex type.

2741 The <AttributeSelector> element has the following attributes:

2742 RequestContextPath [Required]

Deleted: cd-03

2743 An XPath expression whose context node is the `<xacml-context:Request>` element.  
2744 There SHALL be no restriction on the XPath syntax. See also Section 5.4.

2745 **DataType** [Required]

2746 The **bag** returned by the `<AttributeSelector>` element SHALL contain values of this  
2747 data-type.

2748 **MustBePresent** [Optional]

2749 This attribute governs whether the element returns "Indeterminate" or an empty **bag** in the  
2750 event the XPath expression selects no node. See Section 7.2.5. Also see Sections 7.15.2  
2751 and 7.15.3.

## 2752 5.43. Element `<AttributeValue>`

2753 The `<xacml:AttributeValue>` element SHALL contain a literal **attribute** value.

```
2754 <xs:element name="AttributeValue" type="xacml:AttributeValueType"
2755 substitutionGroup="xacml:Expression"/>
2756 <xs:complexType name="AttributeValueType" mixed="true">
2757   <xs:complexContent>
2758     <xs:extension base="xacml:ExpressionType">
2759       <xs:sequence>
2760         <xs:any namespace="##any" processContents="lax" minOccurs="0"
2761 maxOccurs="unbounded"/>
2762       </xs:sequence>
2763       <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
2764       <xs:anyAttribute namespace="##any" processContents="lax"/>
2765     </xs:extension>
2766   </xs:complexContent>
2767 </xs:complexType>
```

2768 The `<xacml:AttributeValue>` element is of **AttributeValueType** complex type.

2769 The `<xacml:AttributeValue>` element has the following attributes:

2770 **DataType** [Required]

2771 The data-type of the **attribute** value.

## 2772 5.44. Element `<Obligations>`

2773 The `<Obligations>` element SHALL contain a set of `<Obligation>` elements.

2774 Support for the `<Obligations>` element is OPTIONAL.

```
2775 <xs:element name="Obligations" type="xacml:ObligationsType"/>
2776 <xs:complexType name="ObligationsType">
2777   <xs:sequence>
2778     <xs:element ref="xacml:Obligation" maxOccurs="unbounded"/>
2779   </xs:sequence>
2780 </xs:complexType>
```

2781 The `<Obligations>` element is of **ObligationsType** complexType.

2782 The `<Obligations>` element contains the following element:

2783 `<Obligation>` [One to Many]

2784 A sequence of **obligations**. See Section 5.45.

Deleted: cd-03

## 5.45. Element <Obligation>

The <Obligation> element SHALL contain an identifier for the **obligation** and a set of **attributes** that form arguments of the action defined by the **obligation**. The FulfillOn attribute SHALL indicate the **effect** for which this **obligation** must be fulfilled by the **PEP**.

```
<xs:element name="Obligation" type="xacml:ObligationType"/>
<xs:complexType name="ObligationType">
  <xs:sequence>
    <xs:element ref="xacml:AttributeAssignment" minOccurs="0"
maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="ObligationId" type="xs:anyURI" use="required"/>
  <xs:attribute name="FulfillOn" type="xacml:EffectType" use="required"/>
</xs:complexType>
```

The <Obligation> element is of **ObligationType** complexType. See Section 7.14 for a description of how the set of **obligations** to be returned by the **PDP** is determined.

The <Obligation> element contains the following elements and attributes:

ObligationId [Required]

**Obligation** identifier. The value of the **obligation** identifier SHALL be interpreted by the **PEP**.

FulfillOn [Required]

The **effect** for which this **obligation** must be fulfilled by the **PEP**.

<AttributeAssignment> [Optional]

**Obligation** arguments assignment. The values of the **obligation** arguments SHALL be interpreted by the **PEP**.

## 5.46. Element <AttributeAssignment>

The <AttributeAssignment> element is used for including arguments in **obligations**. It SHALL contain an AttributeId and the corresponding **attribute** value, by extending the **AttributeValueType** type definition. The <AttributeAssignment> element MAY be used in any way that is consistent with the schema syntax, which is a sequence of <xs:any> elements. The value specified SHALL be understood by the **PEP**, but it is not further specified by XACML. See Section 7.14. Section 4.2.4.3 provides a number of examples of arguments included in **obligations**.

```
<xs:element name="AttributeAssignment" type="xacml:AttributeAssignmentType"/>
<xs:complexType name="AttributeAssignmentType" mixed="true">
  <xs:complexContent>
    <xs:extension base="xacml:AttributeValueType">
      <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

The <AttributeAssignment> element is of **AttributeAssignmentType** complex type.

The <AttributeAssignment> element contains the following attributes:

AttributeId [Required]

The **attribute** Identifier.

Deleted: cd-03

## 6. Context syntax (normative with the exception of the schema fragments)

### 6.1. Element <Request>

The <Request> element is a top-level element in the XACML **context** schema. The <Request> element is an abstraction layer used by the policy language. For simplicity of expression, this document describes **policy** evaluation in terms of operations on the **context**. However a conforming **PDP** is not required to actually instantiate the **context** in the form of an XML document. But, any system conforming to the XACML specification MUST produce exactly the same **authorization decisions** as if all the inputs had been transformed into the form of an <xacml-context:Request> element.

The <Request> element contains <Subject>, <Resource>, <Action> and <Environment> elements. There may be multiple <Subject> elements and, under some conditions, multiple <Resource> elements<sup>2</sup>. Each child element contains a sequence of <xacml-context:Attribute> elements associated with the **subject**, **resource**, **action** and **environment** respectively. These <Attribute> elements MAY form a part of **policy** evaluation.

```
<xs:element name="Request" type="xacml-context:RequestType"/>
<xs:complexType name="RequestType">
  <xs:sequence>
    <xs:element ref="xacml-context:Subject" maxOccurs="unbounded"/>
    <xs:element ref="xacml-context:Resource" maxOccurs="unbounded"/>
    <xs:element ref="xacml-context:Action"/>
    <xs:element ref="xacml-context:Environment"/>
  </xs:sequence>
</xs:complexType>
```

The <Request> element is of **RequestType** complex type.

The <Request> element contains the following elements:

<Subject> [One to Many]

Specifies information about a **subject** of the request **context** by listing a sequence of <Attribute> elements associated with the **subject**. One or more <Subject> elements are allowed. A **subject** is an entity associated with the **access** request. For example, one **subject** might represent the human user that initiated the application from which the request was issued; another **subject** might represent the application's executable code responsible for creating the request; another **subject** might represent the machine on which the application was executing; and another **subject** might represent the entity that is to be the recipient of the **resource**. Attributes of each of these entities MUST be enclosed in separate <Subject> elements.

<Resource> [One to Many]

Specifies information about the **resource** or **resources** for which **access** is being requested by listing a sequence of <Attribute> elements associated with the **resource**. It MAY include a <ResourceContent> element.

<sup>2</sup> The conditions under which multiple <Resource> elements are allowed are described in the XACML Profile for Multiple Resources [MULT].

2869 <Action> [Required]  
2870 Specifies the requested **action** to be performed on the **resource** by listing a set of  
2871 <Attribute> elements associated with the **action**.  
2872 <Environment> [Required]  
2873 Contains a set of <Attribute> elements for the **environment**.

## 2874 6.2. Element <Subject>

2875 The <Subject> element specifies a **subject** by listing a sequence of <Attribute> elements  
2876 associated with the **subject**.

```
2877 <xs:element name="Subject" type="xacml-context:SubjectType" />  
2878 <xs:complexType name="SubjectType">  
2879   <xs:sequence>  
2880     <xs:element ref="xacml-context:Attribute" minOccurs="0"  
2881     maxOccurs="unbounded" />  
2882   </xs:sequence>  
2883   <xs:attribute name="SubjectCategory" type="xs:anyURI"  
2884   default="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject" />  
2885 </xs:complexType>
```

2886 The <Subject> element is of **SubjectType** complex type.

2887 The <Subject> element contains the following elements and attributes:

2888 SubjectCategory [Optional]

2889 This attribute indicates the role that the parent <Subject> played in the formation of the  
2890 **access** request. If this attribute is not present in a given <Subject> element, then the  
2891 default value of "urn:oasis:names:tc:xacml:1.0:subject-category:access-subject" SHALL be  
2892 used, indicating that the parent <Subject> element represents the entity ultimately  
2893 responsible for initiating the **access** request.

2894 If more than one <Subject> element contains a "urn:oasis:names:tc:xacml:2.0:subject-  
2895 category" attribute with the same value, then the PDP SHALL treat the contents of those  
2896 elements as if they were contained in the same <Subject> element.

2897 <Attribute> [Any Number]

2898 A sequence of **attributes** that apply to the subject.

2899 Typically, a <Subject> element will contain an <Attribute> with an AttributeId of  
2900 "urn:oasis:names:tc:xacml:1.0:subject:subject-id", containing the identity of the **subject**.

2901 A <Subject> element MAY contain additional <Attribute> elements.

## 2902 6.3. Element <Resource>

2903 The <Resource> element specifies information about the **resource** to which **access** is requested,  
2904 by listing a sequence of <Attribute> elements associated with the **resource**. It MAY include the  
2905 **resource** content.

```
2906 <xs:element name="Resource" type="xacml-context:ResourceType" />  
2907 <xs:complexType name="ResourceType">  
2908   <xs:sequence>  
2909     <xs:element ref="xacml-context:ResourceContent" minOccurs="0" />
```

Deleted: cd-03

```

2910 <xs:element ref="xacml-context:Attribute" minOccurs="0"
2911 maxOccurs="unbounded"/>
2912 </xs:sequence>
2913 </xs:complexType>
2914 The <Resource> element is of ResourceType complex type.
2915 The <Resource> element contains the following elements:
2916 <ResourceContent> [Optional]
2917     The resource content.
2918 <Attribute> [Any Number]
2919     A sequence of resource attributes.
2920     The <Resource> element MAY contain one or more <Attribute> elements with an
2921     AttributeId of "urn:oasis:names:tc:xacml:2.0:resource:resource-id". Each such
2922     <Attribute> SHALL be an absolute and fully-resolved representation of the identity of
2923     the single resource to which access is being requested. If there is more than one such
2924     absolute and fully-resolved representation, and if any <Attribute> with this
2925     AttributeId is specified, then an <Attribute> for each such distinct representation of
2926     the resource identity SHALL be specified. All such <Attribute> elements SHALL refer
2927     to the same single resource instance. A Profile for a particular resource MAY specify a
2928     single normative representation for instances of the resource; in this case, any
2929     <Attribute> with this AttributeId SHALL use only this one representation.
2930     A <Resource> element MAY contain additional <Attribute> elements.

```

#### 2931 6.4. Element <ResourceContent>

2932 The <ResourceContent> element is a notional placeholder for the content of the **resource**. If an  
 2933 XACML **policy** references the contents of the **resource** by means of an <AttributeSelector>  
 2934 element, then the <ResourceContent> element MUST be included in the  
 2935 RequestContextPath string.

```

2936 <xs:complexType name="ResourceContentType" mixed="true">
2937   <xs:sequence>
2938     <xs:any namespace="##any" processContents="lax" minOccurs="0"
2939     maxOccurs="unbounded"/>
2940   </xs:sequence>
2941   <xs:anyAttribute namespace="##any" processContents="lax"/>
2942 </xs:complexType>

```

2943 The <ResourceContent> element is of **ResourceContentType** complex type.

2944 The <ResourceContent> element allows arbitrary elements and attributes.

#### 2945 6.5. Element <Action>

2946 The <Action> element specifies the requested **action** on the **resource**, by listing a set of  
 2947 <Attribute> elements associated with the **action**.

```

2948 <xs:element name="Action" type="xacml-context:ActionType"/>
2949 <xs:complexType name="ActionType">
2950   <xs:sequence>
2951     <xs:element ref="xacml-context:Attribute" minOccurs="0"
2952     maxOccurs="unbounded"/>
2953   </xs:sequence>

```

Deleted: cd-03



2954 </xs:complexType>  
2955 The <Action> element is of **ActionType** complex type.  
2956 The <Action> element contains the following elements:  
2957 <Attribute> [Any Number]  
2958 List of **attributes** of the **action** to be performed on the **resource**.

## 2959 6.6. Element <Environment>

2960 The <Environment> element contains a set of **attributes** of the **environment**.  
2961 <xs:element name="Environment" type="xacml-context:EnvironmentType"/>  
2962 <xs:complexType name="EnvironmentType">  
2963 <xs:sequence>  
2964 <xs:element ref="xacml-context:Attribute" minOccurs="0"  
2965 maxOccurs="unbounded"/>  
2966 </xs:sequence>  
2967 </xs:complexType>  
2968 The <Environment> element is of **EnvironmentType** complex type.  
2969 The <Environment> element contains the following elements:  
2970 <Attribute> [Any Number]  
2971 A list of **environment attributes**. Environment **attributes** are **attributes** that are not  
2972 associated with either the **resource**, the **action** or any of the **subjects** of the **access**  
2973 request.

## 2974 6.7. Element <Attribute>

2975 The <Attribute> element is the central abstraction of the request **context**. It contains **attribute**  
2976 meta-data and one or more **attribute** values. The **attribute** meta-data comprises the **attribute**  
2977 identifier and the **attribute** issuer. <AttributeDesignator> and <AttributeSelector>  
2978 elements in the **policy** MAY refer to **attributes** by means of this meta-data.  
2979 <xs:element name="Attribute" type="xacml-context:AttributeType"/>  
2980 <xs:complexType name="AttributeType">  
2981 <xs:sequence>  
2982 <xs:element ref="xacml-context:AttributeValue" maxOccurs="unbounded"/>  
2983 </xs:sequence>  
2984 <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>  
2985 <xs:attribute name="DataType" type="xs:anyURI" use="required"/>  
2986 <xs:attribute name="Issuer" type="xs:string" use="optional"/>  
2987 </xs:complexType>  
2988 The <Attribute> element is of **AttributeType** complex type.  
2989 The <Attribute> element contains the following attributes and elements:  
2990 AttributeId [Required]  
2991 The **Attribute** identifier. A number of identifiers are reserved by XACML to denote  
2992 commonly used **attributes**. See Appendix B.

2993 DataType [Required]  
2994 The data-type of the contents of the <xacml-context:AttributeValue> element.  
2995 This SHALL be either a primitive type defined by the XACML 2.0 specification or a type

Deleted: cd-03



2996 (primitive or structured) defined in a namespace declared in the <xacml-context>  
 2997 element.

2998 **Issuer** [Optional]

2999 The **Attribute** issuer. For example, this attribute value MAY be an x500Name that binds to  
 3000 a public key, or it may be some other identifier exchanged out-of-band by issuing and  
 3001 relying parties.

3002 <xacml-context:AttributeValue> [One to Many]

3003 One or more **attribute** values. Each **attribute** value MAY have contents that are empty,  
 3004 occur once or occur multiple times.

## 3005 6.8. Element <AttributeValue>

3006 The <xacml-context:AttributeValue> element contains the value of an **attribute**.

```
3007 <xs:element name="AttributeValue" type="xacml-context:AttributeValueType" />
3008 <xs:complexType name="AttributeValueType" mixed="true">
3009   <xs:sequence>
3010     <xs:any namespace="##any" processContents="lax" minOccurs="0"
3011     maxOccurs="unbounded" />
3012   </xs:sequence>
3013   <xs:anyAttribute namespace="##any" processContents="lax" />
3014 </xs:complexType>
```

3015 The <xacml-context:AttributeValue> element is of **AttributeValueType** complex type.

3016 The data-type of the <xacml-context:AttributeValue> SHALL be specified by using the  
 3017 **DataType** attribute of the parent <Attribute> element.

## 3018 6.9. Element <Response>

3019 The <Response> element is a top-level element in the XACML **context** schema. The  
 3020 <Response> element is an abstraction layer used by the **policy** language. Any proprietary  
 3021 system using the XACML specification MUST transform an XACML **context** <Response> element  
 3022 into the form of its **authorization decision**.

3023 The <Response> element encapsulates the **authorization decision** produced by the **PDP**. It includes  
 3024 a sequence of one or more results, with one <Result> element per requested **resource**. Multiple  
 3025 results MAY be returned by some implementations, in particular those that support the XACML  
 3026 Profile for Requests for Multiple Resources [MULT]. Support for multiple results is OPTIONAL.

```
3027 <xs:element name="Response" type="xacml-context:ResponseType" />
3028 <xs:complexType name="ResponseType">
3029   <xs:sequence>
3030     <xs:element ref="xacml-context:Result" maxOccurs="unbounded" />
3031   </xs:sequence>
3032 </xs:complexType>
```

3033 The <Response> element is of **ResponseType** complex type.

3034 The <Response> element contains the following elements:

3035 <Result> [One to Many]

3036 An authorization decision result. See Section 6.10.

Deleted: cd-03

## 6.10. Element <Result>

The <Result> element represents an **authorization decision** result for the **resource** specified by the **ResourceId** **attribute**. It MAY include a set of **obligations** that MUST be fulfilled by the **PEP**. If the **PEP** does not understand or cannot fulfill an **obligation**, then it MUST act as if the **PDP** had denied **access** to the requested **resource**.

```
<xs:complexType name="ResultType">
  <xs:sequence>
    <xs:element ref="xacml-context:Decision"/>
    <xs:element ref="xacml-context:Status" minOccurs="0"/>
    <xs:element ref="xacml:Obligations" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="ResourceId" type="xs:string" use="optional"/>
</xs:complexType>
```

The <Result> element is of **ResultType** complex type.

The <Result> element contains the following attributes and elements:

**ResourceId** [Optional]

The identifier of the requested **resource**. If this attribute is omitted, then the **resource** identity is that specified by the "urn:oasis:names:tc:xacml:1.0:resource:resource-id" **resource attribute** in the corresponding <Request> element.

**<Decision>** [Required]

The **authorization decision**: "Permit", "Deny", "Indeterminate" or "NotApplicable".

**<Status>** [Optional]

Indicates whether errors occurred during evaluation of the **decision request**, and optionally, information about those errors. If the <Response> element contains <Result> elements whose <Status> elements are all identical, and the <Response> element is contained in a protocol wrapper that can convey status information, then the common status information MAY be placed in the protocol wrapper and this <Status> element MAY be omitted from all <Result> elements.

**<Obligations>** [Optional]

A list of **obligations** that MUST be fulfilled by the **PEP**. If the **PEP** does not understand or cannot fulfill an **obligation**, then it MUST act as if the **PDP** had denied **access** to the requested **resource**. See Section 7.14 for a description of how the set of **obligations** to be returned by the PDP is determined.

## 6.11. Element <Decision>

The <Decision> element contains the result of **policy** evaluation.

```
<xs:element name="Decision" type="xacml-context:DecisionType"/>
<xs:simpleType name="DecisionType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="Permit"/>
    <xs:enumeration value="Deny"/>
    <xs:enumeration value="Indeterminate"/>
    <xs:enumeration value="NotApplicable"/>
  </xs:restriction>
</xs:simpleType>
```

Deleted: cd-03

access\_control-xacml-2.0-core-spec-[cd-04](#)

3082 The <Decision> element is of **DecisionType** simple type.

3083 The values of the <Decision> element have the following meanings:

3084 "Permit": the requested **access** is permitted.

3085 "Deny": the requested **access** is denied.

3086 "Indeterminate": the PDP is unable to evaluate the requested **access**. Reasons for such inability include: missing **attributes**, network errors while retrieving **policies**, division by zero during **policy** evaluation, syntax errors in the **decision request** or in the **policy**, etc..

3089 "NotApplicable": the **PDP** does not have any **policy** that applies to this **decision request**.

## 3090 6.12. Element <Status>

3091 The <Status> element represents the status of the **authorization decision** result.

```
3092 <xs:element name="Status" type="xacml-context:StatusType" />
3093 <xs:complexType name="StatusType">
3094   <xs:sequence>
3095     <xs:element ref="xacml-context:StatusCode" />
3096     <xs:element ref="xacml-context:StatusMessage" minOccurs="0" />
3097     <xs:element ref="xacml-context:StatusDetail" minOccurs="0" />
3098   </xs:sequence>
3099 </xs:complexType>
```

3100 The <Status> element is of **StatusType** complex type.

3101 The <Status> element contains the following elements:

3102 <StatusCode> [Required]

3103 Status code.

3104 <StatusMessage> [Optional]

3105 A status message describing the status code.

3106 <StatusDetail> [Optional]

3107 Additional status information.

## 3108 6.13. Element <StatusCode>

3109 The <StatusCode> element contains a major status code value and an optional sequence of minor status codes.

```
3111 <xs:element name="StatusCode" type="xacml-context:StatusCodeType" />
3112 <xs:complexType name="StatusCodeType">
3113   <xs:sequence>
3114     <xs:element ref="xacml-context:StatusCode" minOccurs="0" />
3115   </xs:sequence>
3116   <xs:attribute name="Value" type="xs:anyURI" use="required" />
3117 </xs:complexType>
```

3118 The <StatusCode> element is of **StatusCodeType** complex type.

3119 The <StatusCode> element contains the following attributes and elements:

3120 Value [Required]

Deleted: cd-03

3121 See Section B.9 for a list of values.  
3122 <StatusCode> [Any Number]  
3123 Minor status code. This status code qualifies its parent status code.

## 3124 6.14. Element <StatusMessage>

3125 The <StatusMessage> element is a free-form description of the status code.

```
3126 <xs:element name="StatusMessage" type="xs:string"/>
```

3127 The <StatusMessage> element is of **xs:string** type.

## 3128 6.15. Element <StatusDetail>

3129 The <StatusDetail> element qualifies the <Status> element with additional information.

```
3130 <xs:element name="StatusDetail" type="xacml-context:StatusDetailType"/>  
3131 <xs:complexType name="StatusDetailType">  
3132 <xs:sequence>  
3133 <xs:any namespace="##any" processContents="lax" minOccurs="0"  
3134 maxOccurs="unbounded"/>  
3135 </xs:sequence>  
3136 </xs:complexType>
```

3137 The <StatusDetail> element is of **StatusDetailType** complex type.

3138 The <StatusDetail> element allows arbitrary XML content.

3139 Inclusion of a <StatusDetail> element is optional. However, if a **PDP** returns one of the  
3140 following XACML-defined <StatusCode> values and includes a <StatusDetail> element, then  
3141 the following rules apply.

3142 urn:oasis:names:tc:xacml:1.0:status:ok

3143 A **PDP** MUST NOT return a <StatusDetail> element in conjunction with the “ok” status value.

3144 urn:oasis:names:tc:xacml:1.0:status:missing-attribute

3145 A **PDP** MAY choose not to return any <StatusDetail> information or MAY choose to return a  
3146 <StatusDetail> element containing one or more <xacml-context:  
3147 MissingAttributeDetail> elements.

3148 urn:oasis:names:tc:xacml:1.0:status:syntax-error

3149 A **PDP** MUST NOT return a <StatusDetail> element in conjunction with the “syntax-error” status  
3150 value. A syntax error may represent either a problem with the **policy** being used or with the  
3151 request **context**. The **PDP** MAY return a <StatusMessage> describing the problem.

3152 urn:oasis:names:tc:xacml:1.0:status:processing-error

3153 A **PDP** MUST NOT return <StatusDetail> element in conjunction with the “processing-error”  
3154 status value. This status code indicates an internal problem in the **PDP**. For security reasons, the  
3155 **PDP** MAY choose to return no further information to the **PEP**. In the case of a divide-by-zero error  
3156 or other computational error, the **PDP** MAY return a <StatusMessage> describing the nature of  
3157 the error.

Deleted: cd-03

## 6.16. Element <MissingAttributeDetail>

The <MissingAttributeDetail> element conveys information about **attributes** required for **policy** evaluation that were missing from the request **context**.

```
<xs:element name="MissingAttributeDetail" type="xacml-  
context:MissingAttributeDetailType"/>  
<xs:complexType name="MissingAttributeDetailType">  
  <xs:sequence>  
    <xs:element ref="xacml-context:AttributeValue" minOccurs="0"  
      maxOccurs="unbounded"/>  
  </xs:sequence>  
  <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>  
  <xs:attribute name="DataType" type="xs:anyURI" use="required"/>  
  <xs:attribute name="Issuer" type="xs:string" use="optional"/>  
</xs:complexType>
```

The <MissingAttributeDetail> element is of **MissingAttributeDetailType** complex type.

The <MissingAttributeDetail> element contains the following attributes and elements:

AttributeValue [Optional]

The required value of the missing **attribute**.

<AttributeId> [Required]

The identifier of the missing **attribute**.

<DataType> [Required]

The data-type of the missing **attribute**.

Issuer [Optional]

This attribute, if supplied, SHALL specify the required **Issuer** of the missing **attribute**.

If the PDP includes <xacml-context:AttributeValue> elements in the <MissingAttributeDetail> element, then this indicates the acceptable values for that attribute. If no <xacml-context:AttributeValue> elements are included, then this indicates the names of attributes that the PDP failed to resolve during its evaluation. The list of attributes may be partial or complete. There is no guarantee by the PDP that supplying the missing values or attributes will be sufficient to satisfy the policy.

## 7. Functional requirements (normative)

This section specifies certain functional requirements that are not directly associated with the production or consumption of a particular XACML element.

### 7.1. Policy enforcement point

This section describes the requirements for the **PEP**.

An application functions in the role of the **PEP** if it guards access to a set of **resources** and asks the **PDP** for an **authorization decision**. The **PEP** MUST abide by the **authorization decision** as described in one of the following sub-sections

3196 **7.1.1. Base PEP**

3197 If the **decision** is "Permit", then the **PEP** SHALL permit **access**. If **obligations** accompany the  
3198 **decision**, then the **PEP** SHALL permit **access** only if it understands and it can and will discharge  
3199 those **obligations**.

3200 If the **decision** is "Deny", then the **PEP** SHALL deny **access**. If **obligations** accompany the  
3201 **decision**, then the **PEP** shall deny **access** only if it understands, and it can and will discharge  
3202 those **obligations**.

3203 If the **decision** is "Not Applicable", then the **PEP's** behavior is undefined.

3204 If the **decision** is "Indeterminate", then the **PEP's** behavior is undefined.

3205 **7.1.2. Deny-biased PEP**

3206 If the **decision** is "Permit", then the **PEP** SHALL permit **access**. If **obligations** accompany the  
3207 **decision**, then the **PEP** SHALL permit **access** only if it understands and it can and will discharge  
3208 those **obligations**.

3209 All other **decisions** SHALL result in the denial of **access**.

3210 Note: other actions, e.g. consultation of additional **PDPs**, reformulation/resubmission of the  
3211 **decision request**, etc., are not prohibited.

3212 **7.1.3. Permit-biased PEP**

3213 If the **decision** is "Deny", then the **PEP** SHALL deny **access**. If **obligations** accompany the  
3214 **decision**, then the **PEP** shall deny **access** only if it understands, and it can and will discharge  
3215 those **obligations**.

3216 All other **decisions** SHALL result in the permission of **access**.

3217 Note: other actions, e.g. consultation of additional **PDPs**, reformulation/resubmission of the  
3218 **decision request**, etc., are not prohibited.

3219 **7.2. Attribute evaluation**

3220 **Attributes** are represented in the request **context** by the **context handler**, regardless of whether  
3221 or not they appeared in the original **decision request**, and are referred to in the **policy** by **subject**,  
3222 **resource**, **action** and **environment attribute** designators and **attribute** selectors. A **named**  
3223 **attribute** is the term used for the criteria that the specific **subject**, **resource**, **action** and  
3224 **environment attribute** designators and selectors use to refer to particular **attributes** in the  
3225 **subject**, **resource**, **action** and **environment** elements of the request **context**, respectively.

3226 **7.2.1. Structured attributes**

3227 <xacml:AttributeValue> and <xacml-context:AttributeValue> elements MAY contain  
3228 an instance of a structured XML data-type, for example <ds:KeyInfo>. XACML 2.0 supports  
3229 several ways for comparing the contents of such elements.

- 3230 1. In some cases, such elements MAY be compared using one of the XACML string functions,  
3231 such as "~~string-regexp-match~~", described below. This requires that the element be given  
3232 the data-type "<http://www.w3.org/2001/XMLSchema#string>". For example, a structured  
3233 data-type that is actually a ds:KeyInfo/KeyName would appear in the Context as:

Deleted: regexp-string

Deleted: cd-03

access\_control-xacml-2.0-core-spec-~~cd-04~~

```
3234 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
3235   <ds:KeyName>jhibbert-key</ds:KeyName>
3236 </AttributeValue>
```

3237 In general, this method will not be adequate unless the structured data-type is quite simple.

3238 2. An `<AttributeSelector>` element MAY be used to select the contents of a leaf sub-  
3239 element of the structured data-type by means of an XPath expression. That value MAY  
3240 then be compared using one of the supported XACML functions appropriate for its primitive  
3241 data-type. This method requires support by the **PDP** for the optional XPath expressions  
3242 feature.

3243 3. An `<AttributeSelector>` element MAY be used to select any node in the structured  
3244 data-type by means of an XPath expression. This node MAY then be compared using one  
3245 of the XPath-based functions described in Section A.3. This method requires support by  
3246 the **PDP** for the optional XPath expressions and XPath functions features.

3247 See also Section 8.2.

## 3248 7.2.2. Attribute bags

3249 XACML defines implicit collections of its data-types. XACML refers to a collection of values that are  
3250 of a single data-type as a **bag**. **Bags** of data-types are needed because selections of nodes from  
3251 an XML **resource** or XACML request **context** may return more than one value.

3252 The `<AttributeSelector>` element uses an XPath expression to specify the selection of data  
3253 from an XML **resource**. The result of an XPath expression is termed a *node-set*, which contains all  
3254 the leaf nodes from the XML **resource** that match the predicate in the XPath expression. Based on  
3255 the various indexing functions provided in the XPath specification, it SHALL be implied that a  
3256 resultant node-set is the collection of the matching nodes. XACML also defines the  
3257 `<AttributeDesignator>` element to have the same matching methodology for **attributes** in the  
3258 XACML request **context**.

3259 The values in a **bag** are not ordered, and some of the values may be duplicates. There SHALL be  
3260 no notion of a **bag** containing **bags**, or a **bag** containing values of differing types. I.e. a **bag** in  
3261 XACML SHALL contain only values that are of the same data-type.

## 3262 7.2.3. Multivalued attributes

3263 If a single `<Attribute>` element in a request **context** contains multiple `<xacml-`  
3264 `context:AttributeValue>` child elements, then the **bag** of values resulting from evaluation of  
3265 the `<Attribute>` element MUST be identical to the **bag** of values that results from evaluating a  
3266 **context** in which each `<xacml-context:AttributeValue>` element appears in a separate  
3267 `<Attribute>` element, each carrying identical meta-data.

## 3268 7.2.4. Attribute Matching

3269 A **named attribute** includes specific criteria with which to match **attributes** in the **context**. An  
3270 **attribute** specifies an `AttributeId` and `DataType`, and a **named attribute** also specifies the  
3271 Issuer. A **named attribute** SHALL match an **attribute** if the values of their respective  
3272 `AttributeId`, `DataType` and optional Issuer attributes match within their particular element -  
3273 **subject**, **resource**, **action** or **environment** - of the **context**. The `AttributeId` of the **named**  
3274 **attribute** MUST match, by URI equality, the `AttributeId` of the corresponding **context attribute**.  
3275 The `DataType` of the **named attribute** MUST match, by URI equality, the `DataType` of the  
3276 corresponding **context attribute**. If Issuer is supplied in the **named attribute**, then it MUST

Deleted: cd-03



3277 match, using the `urn:oasis:names:tc:xacml:1.0:function:string-equal` function, the  
3278 Issuer of the corresponding **context attribute**. If Issuer is not supplied in the **named attribute**,  
3279 then the matching of the **context attribute** to the **named attribute** SHALL be governed by  
3280 AttributeId and DataType alone, regardless of the presence, absence, or actual value of  
3281 Issuer in the corresponding **context attribute**. In the case of an **attribute** selector, the matching  
3282 of the **attribute** to the **named attribute** SHALL be governed by the XPath expression and  
3283 DataType.

### 3284 7.2.5. Attribute Retrieval

3285 The **PDP** SHALL request the values of **attributes** in the request **context** from the **context handler**.  
3286 The **PDP** SHALL reference the **attributes** as if they were in a physical request **context** document,  
3287 but the **context handler** is responsible for obtaining and supplying the requested values by  
3288 whatever means it deems appropriate. The **context handler** SHALL return the values of  
3289 **attributes** that match the **attribute** designator or **attribute** selector and form them into a **bag** of  
3290 values with the specified data-type. If no **attributes** from the request **context** match, then the  
3291 **attribute** SHALL be considered missing. If the **attribute** is missing, then **MustBePresent**  
3292 governs whether the **attribute** designator or **attribute** selector returns an empty **bag** or an  
3293 "Indeterminate" result. If **MustBePresent** is "False" (default value), then a missing **attribute**  
3294 SHALL result in an empty **bag**. If **MustBePresent** is "True", then a missing **attribute** SHALL  
3295 result in "Indeterminate". This "Indeterminate" result SHALL be handled in accordance with the  
3296 specification of the encompassing expressions, **rules**, **policies** and **policy sets**. If the result is  
3297 "Indeterminate", then the AttributeId, DataType and Issuer of the **attribute** MAY be listed in  
3298 the **authorization decision** as described in Section 7.13. However, a **PDP** MAY choose not to  
3299 return such information for security reasons.

### 3300 7.2.6. Environment Attributes

3301 Standard **environment attributes** are listed in Section B.8. If a value for one of these **attributes** is  
3302 supplied in the **decision request**, then the **context handler** SHALL use that value. Otherwise, the  
3303 **context handler** SHALL supply a value. In the case of date and time **attributes**, the supplied  
3304 value SHALL have the semantics of the "date and time that apply to the **decision request**".

## 3305 7.3. Expression evaluation

3306 XACML specifies expressions in terms of the elements listed below, of which the `<Apply>` and  
3307 `<Condition>` elements recursively compose greater expressions. Valid expressions SHALL be  
3308 type correct, which means that the types of each of the elements contained within `<Apply>` and  
3309 `<Condition>` elements SHALL agree with the respective argument types of the function that is  
3310 named by the `FunctionId` attribute. The resultant type of the `<Apply>` or `<Condition>`  
3311 element SHALL be the resultant type of the function, which MAY be narrowed to a primitive data-  
3312 type, or a **bag** of a primitive data-type, by type-unification. XACML defines an evaluation result of  
3313 "Indeterminate", which is said to be the result of an invalid expression, or an operational error  
3314 occurring during the evaluation of the expression.

3315 XACML defines these elements to be in the substitution group of the `<Expression>` element:

- 3316 • `<xacml:AttributeValue>`
- 3317 • `<xacml:SubjectAttributeDesignator>`
- 3318 • `<xacml:ResourceAttributeDesignator>`
- 3319 • `<xacml:ActionAttributeDesignator>`

Deleted: cd-03



- 3320 • <xacml:EnvironmentAttributeDesignator>
- 3321 • <xacml:AttributeSelector>
- 3322 • <xacml:Apply>
- 3323 • <xacml:Condition>
- 3324 • <xacml:Function>
- 3325 • <xacml:VariableReference>

#### 3326 7.4. Arithmetic evaluation

3327 IEEE 754 [IEEE 754] specifies how to evaluate arithmetic functions in a context, which specifies  
3328 defaults for precision, rounding, etc. XACML SHALL use this specification for the evaluation of all  
3329 integer and double functions relying on the *Extended Default Context*, enhanced with double  
3330 precision:

- 3331 flags - all set to 0
- 3332 trap-enablers - all set to 0 (IEEE 854 §7) with the exception of the "division-by-zero" trap
- 3333 enabler, which SHALL be set to 1
- 3334 precision - is set to the designated double precision
- 3335 rounding - is set to round-half-even (IEEE 854 §4.1)

#### 3336 7.5. Match evaluation

3337 **Attribute** matching elements appear in the <Target> element of **rules**, **policies** and **policy sets**.  
3338 They are the following:

- 3339 <SubjectMatch>
- 3340 <ResourceMatch>
- 3341 <ActionMatch>
- 3342 <EnvironmentMatch>

3343 These elements represent Boolean expressions over **attributes** of the **subject**, **resource**, **action**  
3344 and **environment**, respectively. A matching element contains a MatchId attribute that specifies  
3345 the function to be used in performing the match evaluation, an <xacml:AttributeValue> and an  
3346 <AttributeDesignator> or <AttributeSelector> element that specifies the **attribute** in the  
3347 **context** that is to be matched against the specified value.

3348 The MatchId attribute SHALL specify a function that compares two arguments, returning a result  
3349 type of "http://www.w3.org/2001/XMLSchema#boolean". The **attribute** value specified in the  
3350 matching element SHALL be supplied to the MatchId function as its first argument. An element of  
3351 the **bag** returned by the <AttributeDesignator> or <AttributeSelector> element SHALL  
3352 be supplied to the MatchId function as its second argument, as explained below. The DataType  
3353 of the <xacml:AttributeValue> SHALL match the data-type of the first argument expected by  
3354 the MatchId function. The DataType of the <AttributeDesignator> or  
3355 <AttributeSelector> element SHALL match the data-type of the second argument expected  
3356 by the MatchId function.

Deleted: cd-03

3357 The XACML standard functions that meet the requirements for use as a `MatchId` attribute value  
3358 are:

3359 urn:oasis:names:tc:xacml:2.0:function:-type-equal  
3360 urn:oasis:names:tc:xacml:2.0:function:-type-greater-than  
3361 urn:oasis:names:tc:xacml:2.0:function:-type-greater-than-or-equal  
3362 urn:oasis:names:tc:xacml:2.0:function:-type-less-than  
3363 urn:oasis:names:tc:xacml:2.0:function:-type-less-than-or-equal  
3364 urn:oasis:names:tc:xacml:2.0:function:-type-match

3365 In addition, functions that are strictly within an extension to XACML MAY appear as a value for the  
3366 `MatchId` attribute, and those functions MAY use data-types that are also extensions, so long as  
3367 the extension function returns a Boolean result and takes two single base types as its inputs. The  
3368 function used as the value for the `MatchId` attribute SHOULD be easily indexable. Use of non-  
3369 indexable or complex functions may prevent efficient evaluation of **decision requests**.

3370 The evaluation semantics for a matching element is as follows. If an operational error were to  
3371 occur while evaluating the `<AttributeDesignator>` or `<AttributeSelector>` element, then  
3372 the result of the entire expression SHALL be "Indeterminate". If the `<AttributeDesignator>` or  
3373 `<AttributeSelector>` element were to evaluate to an empty **bag**, then the result of the  
3374 expression SHALL be "False". Otherwise, the `MatchId` function SHALL be applied between the  
3375 `<xacml:AttributeValue>` and each element of the **bag** returned from the  
3376 `<AttributeDesignator>` or `<AttributeSelector>` element. If at least one of those function  
3377 applications were to evaluate to "True", then the result of the entire expression SHALL be "True".  
3378 Otherwise, if at least one of the function applications results in "Indeterminate", then the result  
3379 SHALL be "Indeterminate". Finally, if all function applications evaluate to "False", then the result of  
3380 the entire expression SHALL be "False".

3381 It is also possible to express the semantics of a **target** matching element in a **condition**. For  
3382 instance, the **target** match expression that compares a "subject-name" starting with the name  
3383 "John" can be expressed as follows:

```
3384 <SubjectMatch  
3385 MatchId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match">  
3386 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">  
3387 John.*  
3388 </AttributeValue>  
3389 <SubjectAttributeDesignator  
3390 AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"  
3391 DataType="http://www.w3.org/2001/XMLSchema#string"/>  
3392 </SubjectMatch>
```

Deleted: regexp-string

3393 Alternatively, the same match semantics can be expressed as an `<Apply>` element in a **condition**  
3394 by using the "urn:oasis:names:tc:xacml:1.0:function:any-of" function, as follows:

```
3395 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of">  
3396 <Function  
3397 FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match"/>  
3398 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">  
3399 John.*  
3400 </AttributeValue>  
3401 <SubjectAttributeDesignator  
3402 AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"  
3403 DataType="http://www.w3.org/2001/XMLSchema#string"/>  
3404 </Apply>
```

Deleted: regexp-string

Deleted: cd-03

access\_control-xacml-2.0-core-spec: cd-04

82

## 7.6. Target evaluation

The **target** value SHALL be "Match" if the **subjects**, **resources**, **actions** and **environments** specified in the **target** all match values in the request **context**. If any one of the **subjects**, **resources**, **actions** and **environments** specified in the **target** are "Indeterminate", then the **target** SHALL be "Indeterminate". Otherwise, the **target** SHALL be "No match". The **target** match table is shown in Table 1.

| Subjects value        | Resources value       | Actions value         | Environments value    | Target value    |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------|
| "Match"               | "Match"               | "Match"               | "Match"               | "Match"         |
| "No match"            | "Match" or "No match" | "Match" or "No match" | "Match" or "No match" | "No match"      |
| "Match" or "No match" | "No match"            | "Match" or "No match" | "Match" or "No match" | "No match"      |
| "Match" or "No match" | "Match" or "No match" | "No match"            | "Match" or "No match" | "No match"      |
| "Match" or "No match" | "Match" or "No match" | "Match" or "No match" | "No match"            | "No match"      |
| "Indeterminate"       | Don't care            | Don't care            | Don't care            | "Indeterminate" |
| Don't care            | "Indeterminate"       | Don't care            | Don't care            | "Indeterminate" |
| Don't care            | Don't care            | "Indeterminate"       | Don't care            | "Indeterminate" |
| Don't care            | Don't care            | Don't care            | "Indeterminate"       | "Indeterminate" |

Table 1 - Target match table

The **subjects**, **resources**, **actions** and **environments** SHALL match values in the request **context** if at least one of their <Subject>, <Resource>, <Action> or <Environment> elements, respectively, matches a value in the request **context**. The **subjects** match table is shown in Table 2. The **resources**, **actions** and **environments** match tables are analogous.

| <Subject> values                              | <Subjects> Value |
|---|------------------|
| At least one "Match"                          | "Match"          |
| None matches and at least one "Indeterminate" | "Indeterminate"  |
| All "No match"                                | "No match"       |

Table 2 - Subjects match table

A **subject**, **resource**, **action** or **environment** SHALL match a value in the request **context** if the value of all its <SubjectMatch>, <ResourceMatch>, <ActionMatch> or <EnvironmentMatch> elements, respectively, are "True".

The **subject** match table is shown in Table 3. The **resource**, **action** and **environment** match tables are analogous.

Deleted: cd-03

| <SubjectMatch> values                       | <Subject> Value |
|---|-----------------|
| All "True"                                  | "Match"         |
| No "False" and at least one "Indeterminate" | "Indeterminate" |
| At least one "False"                        | "No match"      |

**Table 3 - Subject match table**

## 7.7. VariableReference Evaluation

The <VariableReference> element references a single <VariableDefinition> element contained within the same <Policy> element. A <VariableReference> that does not reference a particular <VariableDefinition> element within the encompassing <Policy> element is called an undefined reference. **Policies** with undefined references are invalid.

In any place where a <VariableReference> occurs, it has the effect as if the text of the <Expression> element defined in the <VariableDefinition> element replaces the <VariableReference> element. Any evaluation scheme that preserves this semantic is acceptable. For instance, the expression in the <VariableDefinition> element may be evaluated to a particular value and cached for multiple references without consequence. (I.e. the value of an <Expression> element remains the same for the entire policy evaluation.) This characteristic is one of the benefits of XACML being a declarative language.

## 7.8. Condition evaluation

The **condition** value SHALL be "True" if the <Condition> element is absent, or if it evaluates to "True". Its value SHALL be "False" if the <Condition> element evaluates to "False". The **condition** value SHALL be "Indeterminate", if the expression contained in the <Condition> element evaluates to "Indeterminate".

## 7.9. Rule evaluation

A **rule** has a value that can be calculated by evaluating its contents. **Rule** evaluation involves separate evaluation of the **rule's target** and **condition**. The **rule** truth table is shown in Table 4.

| Target          | Condition       | Rule Value      |
|-----------------|-----------------|-----------------|
| "Match"         | "True"          | Effect          |
| "Match"         | "False"         | "NotApplicable" |
| "Match"         | "Indeterminate" | "Indeterminate" |
| "No-match"      | Don't care      | "NotApplicable" |
| "Indeterminate" | Don't care      | "Indeterminate" |

**Table 4 - Rule truth table**

If the **target** value is "No-match" or "Indeterminate" then the **rule** value SHALL be "NotApplicable" or "Indeterminate", respectively, regardless of the value of the **condition**. For these cases, therefore, the **condition** need not be evaluated.

Deleted: cd-03

access\_control-xacml-2.0-core-spec-[cd-04](#)

3450 If the **target** value is "Match" and the **condition** value is "True", then the **effect** specified in the  
3451 enclosing <Rule> element SHALL determine the **rule's** value.

3452 **7.10. Policy evaluation**

3453 The value of a **policy** SHALL be determined only by its contents, considered in relation to the  
3454 contents of the request **context**. A **policy's** value SHALL be determined by evaluation of the  
3455 **policy's target** and **rules**.

3456 The **policy's target** SHALL be evaluated to determine the applicability of the **policy**. If the **target**  
3457 evaluates to "Match", then the value of the **policy** SHALL be determined by evaluation of the  
3458 **policy's rules**, according to the specified **rule-combining algorithm**. If the **target** evaluates to  
3459 "No-match", then the value of the **policy** SHALL be "NotApplicable". If the **target** evaluates to  
3460 "Indeterminate", then the value of the **policy** SHALL be "Indeterminate".

3461 The **policy** truth table is shown in Table 5.

| Target          | Rule values                                | Policy Value                                     |
|-----------------|--|--|
| "Match"         | At least one rule value is its Effect      | Specified by the <b>rule-combining algorithm</b> |
| "Match"         | All rule values are "NotApplicable"        | "NotApplicable"                                  |
| "Match"         | At least one rule value is "Indeterminate" | Specified by the <b>rule-combining algorithm</b> |
| "No-match"      | Don't care                                 | "NotApplicable"                                  |
| "Indeterminate" | Don't care                                 | "Indeterminate"                                  |

3462 **Table 5 - Policy truth table**

3463 A **rules** value of "At least one rule value is its Effect" means either that the <Rule> element is  
3464 absent, or one or more of the **rules** contained in the **policy** is applicable to the **decision request**  
3465 (i.e., it returns the value of its "Effect"; see Section 7.9). A **rules** value of "All rule values are  
3466 'NotApplicable'" SHALL be used if no **rule** contained in the **policy** is applicable to the request and if  
3467 no **rule** contained in the **policy** returns a value of "Indeterminate". If no **rule** contained in the  
3468 **policy** is applicable to the request, but one or more **rule** returns a value of "Indeterminate", then the  
3469 **rules** SHALL evaluate to "At least one rule value is 'Indeterminate'".

3470 If the **target** value is "No-match" or "Indeterminate" then the **policy** value SHALL be  
3471 "NotApplicable" or "Indeterminate", respectively, regardless of the value of the **rules**. For these  
3472 cases, therefore, the **rules** need not be evaluated.

3473 If the **target** value is "Match" and the **rule** value is "At least one rule value is it's Effect" or "At least  
3474 one rule value is 'Indeterminate'", then the **rule-combining algorithm** specified in the **policy**  
3475 SHALL determine the **policy** value.

3476 Note that none of the **rule-combining algorithms** defined by XACML 2.0 take parameters.  
3477 However, non-standard **combining algorithms** MAY take parameters. In such a case, the values  
3478 of these parameters associated with the **rules**, MUST be taken into account when evaluating the  
3479 **policy**. The parameters and their types should be defined in the specification of the **combining**  
3480 **algorithm**. If the implementation supports combiner parameters and if combiner parameters are

Deleted: cd-03

3481 present in a **policy**, then the parameter values MUST be supplied to the **combining algorithm**  
3482 implementation.

3483 **7.11. Policy Set evaluation**

3484 The value of a **policy set** SHALL be determined by its contents, considered in relation to the  
3485 contents of the **request context**. A **policy set's** value SHALL be determined by evaluation of the  
3486 **policy set's target, policies** and **policy sets**, according to the specified **policy-combining**  
3487 **algorithm**.

3488 The **policy set's target** SHALL be evaluated to determine the applicability of the **policy set**. If the  
3489 **target** evaluates to "Match" then the value of the **policy set** SHALL be determined by evaluation of  
3490 the **policy set's policies** and **policy sets**, according to the specified **policy-combining algorithm**.  
3491 If the **target** evaluates to "No-match", then the value of the **policy set** shall be "NotApplicable". If  
3492 the **target** evaluates to "Indeterminate", then the value of the **policy set** SHALL be "Indeterminate".

3493 The **policy set** truth table is shown in Table 6.

| Target          | Policy values                                    | Policy Set Value                                   |
|-----------------|--|--|
| "Match"         | At least one policy value is its <b>Decision</b> | Specified by the <b>policy-combining algorithm</b> |
| "Match"         | All policy values are "NotApplicable"            | "NotApplicable"                                    |
| "Match"         | At least one policy value is "Indeterminate"     | Specified by the <b>policy-combining algorithm</b> |
| "No-match"      | Don't care                                       | "NotApplicable"                                    |
| "Indeterminate" | Don't care                                       | "Indeterminate"                                    |

3494 **Table 6 – Policy set truth table**

3495 A **policies** value of "At least one policy value is its **Decision**" SHALL be used if there are no  
3496 contained or referenced **policies** or **policy sets**, or if one or more of the **policies** or **policy sets**  
3497 contained in or referenced by the **policy set** is applicable to the **decision request** (i.e., returns a  
3498 value determined by its **combining algorithm**) A **policies** value of "All policy values are  
3499 'NotApplicable'" SHALL be used if no **policy** or **policy set** contained in or referenced by the **policy**  
3500 **set** is applicable to the request and if no **policy** or **policy set** contained in or referenced by the  
3501 **policy set** returns a value of "Indeterminate". If no **policy** or **policy set** contained in or referenced  
3502 by the **policy set** is applicable to the request but one or more **policy** or **policy set** returns a value  
3503 of "Indeterminate", then the **policies** SHALL evaluate to "At least one policy value is  
3504 'Indeterminate'".

3505 If the **target** value is "No-match" or "Indeterminate" then the **policy set** value SHALL be  
3506 "NotApplicable" or "Indeterminate", respectively, regardless of the value of the **policies**. For these  
3507 cases, therefore, the **policies** need not be evaluated.

3508 If the **target** value is "Match" and the **policies** value is "At least one policy value is its **Decision**" or  
3509 "At least one policy value is 'Indeterminate'", then the **policy-combining algorithm** specified in the  
3510 **policy set** SHALL determine the **policy set** value.

3511 Note that none of the **policy-combining algorithms** defined by XACML 2.0 take parameters.  
3512 However, non-standard **combining algorithms** MAY take parameters. In such a case, the values

Deleted: cd-03

3513 of these parameters associated with the **policies**, MUST be taken into account when evaluating the  
3514 **policy set**. The parameters and their types should be defined in the specification of the  
3515 **combining algorithm**. If the implementation supports combiner parameters and if combiner  
3516 parameters are present in a **policy**, then the parameter values MUST be supplied to the  
3517 **combining algorithm** implementation.

## 3518 7.12. Hierarchical resources

3519 It is often the case that a **resource** is organized as a hierarchy (e.g. file system, XML document).  
3520 XACML provides several optional mechanisms for supporting hierarchical resources. These are  
3521 described in the XACML Profile for Hierarchical Resources [HIER] and in the XACML Profile for  
3522 Requests for Multiple Resources [MULT].

## 3523 7.13. Authorization decision

3524 In relation to a particular **decision request**, the **PDP** is defined by a **policy-combining algorithm**  
3525 and a set of **policies** and/or **policy sets**. The **PDP** SHALL return a response **context** as if it had  
3526 evaluated a single **policy set** consisting of this **policy-combining algorithm** and the set of  
3527 **policies** and/or **policy sets**.

3528 The **PDP** MUST evaluate the **policy set** as specified in Sections 5 and 7. The **PDP** MUST return a  
3529 response **context**, with one <Decision> element of value "Permit", "Deny", "Indeterminate" or  
3530 "NotApplicable".

3531 If the **PDP** cannot make a decision, then an "Indeterminate" <Decision> element SHALL be  
3532 returned.

## 3533 7.14. Obligations

3534 A **policy** or **policy set** may contain one or more **obligations**. When such a **policy** or **policy set** is  
3535 evaluated, an **obligation** SHALL be passed up to the next level of evaluation (the enclosing or  
3536 referencing **policy**, **policy set** or **authorization decision**) only if the **effect** of the **policy** or **policy**  
3537 **set** being evaluated matches the value of the FulfillOn attribute of the **obligation**.

3538 As a consequence of this procedure, no **obligations** SHALL be returned to the **PEP** if the **policies**  
3539 or **policy sets** from which they are drawn are not evaluated, or if their evaluated result is  
3540 "Indeterminate" or "NotApplicable", or if the **decision** resulting from evaluating the **policy** or **policy**  
3541 **set** does not match the **decision** resulting from evaluating an enclosing **policy set**.

3542 If the **PDP's** evaluation is viewed as a tree of **policy sets** and **policies**, each of which returns  
3543 "Permit" or "Deny", then the set of **obligations** returned by the **PDP** to the **PEP** will include only the  
3544 **obligations** associated with those paths where the **effect** at each level of evaluation is the same as  
3545 the **effect** being returned by the **PDP**. In situations where any lack of determinism is unacceptable,  
3546 a deterministic combining algorithm, such as ordered-deny-overrides, should be used.

3547 Also, see Section 7.1.

## 3550 7.15. Exception handling

3551 XACML specifies behaviour for the **PDP** in the following situations.

Deleted: cd-03



### 7.15.1. Unsupported functionality

If the **PDP** attempts to evaluate a **policy set** or **policy** that contains an optional element type or function that the **PDP** does not support, then the **PDP** SHALL return a <Decision> value of "Indeterminate". If a <StatusCode> element is also returned, then its value SHALL be "urn:oasis:names:tc:xacml:1.0:status:syntax-error" in the case of an unsupported element type, and "urn:oasis:names:tc:xacml:1.0:status:processing-error" in the case of an unsupported function.

### 7.15.2. Syntax and type errors

If a **policy** that contains invalid syntax is evaluated by the XACML **PDP** at the time a **decision request** is received, then the result of that **policy** SHALL be "Indeterminate" with a StatusCode value of "urn:oasis:names:tc:xacml:1.0:status:syntax-error".

If a **policy** that contains invalid static data-types is evaluated by the XACML **PDP** at the time a **decision request** is received, then the result of that **policy** SHALL be "Indeterminate" with a StatusCode value of "urn:oasis:names:tc:xacml:1.0:status:processing-error".

### 7.15.3. Missing attributes

The absence of matching **attributes** in the request **context** for any of the **attribute** designators or selectors that are found in the **policy** SHALL result in a <Decision> element containing the "Indeterminate" value, as described in Sections 5.37 and 5.42. If, in this case, and a status code is supplied, then the value

"urn:oasis:names:tc:xacml:1.0:status:missing-attribute"

SHALL be used, to indicate that more information is needed in order for a definitive decision to be rendered. In this case, the <Status> element MAY list the names and data-types of any **attributes** of the **subjects**, **resource**, **action** or **environment** that are needed by the **PDP** to refine its decision (see Section 6.16). A **PEP** MAY resubmit a refined request **context** in response to a <Decision> element contents of "Indeterminate" with a status code of

"urn:oasis:names:tc:xacml:1.0:missing-attribute"

by adding **attribute** values for the **attribute** names that were listed in the previous response. When the **PDP** returns a <Decision> element contents of "Indeterminate", with a status code of

"urn:oasis:names:tc:xacml:1.0:missing-attribute",

it MUST NOT list the names and data-types of any **attribute** of the **subject**, **resource**, **action** or **environment** for which values were supplied in the original request. Note, this requirement forces the **PDP** to eventually return an **authorization decision** of "Permit", "Deny" or "Indeterminate" with some other status code, in response to successively-refined requests.

---

## 8. XACML extensibility points (non-normative)

This section describes the points within the XACML model and schema where extensions can be added



## 8.1. Extensible XML attribute types

The following XML attributes have values that are URIs. These may be extended by the creation of new URIs associated with new semantics for these attributes.

AttributeId,  
DataType,  
FunctionId,  
MatchId,  
ObligationId,  
PolicyCombiningAlgId,  
RuleCombiningAlgId,  
StatusCode,  
SubjectCategory.

See Section 5 for definitions of these attribute types.

## 8.2. Structured attributes

`<xacml:AttributeValue>` and `<xacml-context:AttributeValue>` elements MAY contain an instance of a structured XML data-type. Section 7.2.1 describes a number of standard techniques to identify data items within such a structured attribute. Listed here are some additional techniques that require XACML extensions.

1. For a given structured data-type, a community of XACML users MAY define new attribute identifiers for each leaf sub-element of the structured data-type that has a type conformant with one of the XACML-defined primitive data-types. Using these new attribute identifiers, the *PEPs* or *context handlers* used by that community of users can flatten instances of the structured data-type into a sequence of individual `<Attribute>` elements. Each such `<Attribute>` element can be compared using the XACML-defined functions. Using this method, the structured data-type itself never appears in an `<xacml-context:AttributeValue>` element.
2. A community of XACML users MAY define a new function that can be used to compare a value of the structured data-type against some other value. This method may only be used by *PDPs* that support the new function.

---

## 9. Security and privacy considerations (non-normative)

This section identifies possible security and privacy compromise scenarios that should be considered when implementing an XACML-based system. The section is informative only. It is left to the implementer to decide whether these compromise scenarios are practical in their environment and to select appropriate safeguards.

## 9.1. Threat model

We assume here that the adversary has access to the communication channel between the XACML actors and is able to interpret, insert, delete and modify messages or parts of messages.

Additionally, an actor may use information from a former message maliciously in subsequent transactions. It is further assumed that **rules** and **policies** are only as reliable as the actors that create and use them. Thus it is incumbent on each actor to establish appropriate trust in the other actors upon which it relies. Mechanisms for trust establishment are outside the scope of this specification.

The messages that are transmitted between the actors in the XACML model are susceptible to attack by malicious third parties. Other points of vulnerability include the **PEP**, the **PDP** and the **PAP**. While some of these entities are not strictly within the scope of this specification, their compromise could lead to the compromise of **access control** enforced by the **PEP**.

It should be noted that there are other components of a distributed system that may be compromised, such as an operating system and the domain-name system (DNS) that are outside the scope of this discussion of threat models. Compromise in these components may also lead to a policy violation.

The following sections detail specific compromise scenarios that may be relevant to an XACML system.

### 9.1.1. Unauthorized disclosure

XACML does not specify any inherent mechanisms to protect the confidentiality of the messages exchanged between actors. Therefore, an adversary could observe the messages in transit. Under certain security policies, disclosure of this information is a violation. Disclosure of **attributes** or the types of **decision requests** that a **subject** submits may be a breach of privacy policy. In the commercial sector, the consequences of unauthorized disclosure of personal data may range from embarrassment to the custodian to imprisonment and large fines in the case of medical or financial data.

Unauthorized disclosure is addressed by confidentiality safeguards.

### 9.1.2. Message replay

A message replay attack is one in which the adversary records and replays legitimate messages between XACML actors. This attack may lead to denial of service, the use of out-of-date information or impersonation.

Prevention of replay attacks requires the use of message freshness safeguards.

Note that encryption of the message does not mitigate a replay attack since the message is simply replayed and does not have to be understood by the adversary.

### 9.1.3. Message insertion

A message insertion attack is one in which the adversary inserts messages in the sequence of messages between XACML actors.

The solution to a message insertion attack is to use mutual authentication and message sequence integrity safeguards between the actors. It should be noted that just using SSL mutual authentication is not sufficient. This only proves that the other party is the one identified by the

Deleted: cd-03

3662 subject of the X.509 certificate. In order to be effective, it is necessary to confirm that the certificate  
3663 subject is authorized to send the message.

#### 3664 9.1.4. Message deletion

3665 A message deletion attack is one in which the adversary deletes messages in the sequence of  
3666 messages between XACML actors. Message deletion may lead to denial of service. However, a  
3667 properly designed XACML system should not render an incorrect authorization decision as a result  
3668 of a message deletion attack.

3669 The solution to a message deletion attack is to use message sequence integrity safeguards  
3670 between the actors.

#### 3671 9.1.5. Message modification

3672 If an adversary can intercept a message and change its contents, then they may be able to alter an  
3673 **authorization decision**. A message integrity safeguard can prevent a successful message  
3674 modification attack.

#### 3675 9.1.6. NotApplicable results

3676 A result of "NotApplicable" means that the **PDP** could not locate a **policy** whose **target** matched  
3677 the information in the **decision request**. In general, it is highly recommended that a "Deny" **effect**  
3678 **policy** be used, so that when a **PDP** would have returned "NotApplicable", a result of "Deny" is  
3679 returned instead.

3680 In some security models, however, such as those found in many Web Servers, an **authorization**  
3681 **decision** of "NotApplicable" is treated as equivalent to "Permit". There are particular security  
3682 considerations that must be taken into account for this to be safe. These are explained in the  
3683 following paragraphs.

3684 If "NotApplicable" is to be treated as "Permit", it is vital that the matching algorithms used by the  
3685 **policy** to match elements in the **decision request** be closely aligned with the data syntax used by  
3686 the applications that will be submitting the **decision request**. A failure to match will result in  
3687 "NotApplicable" and be treated as "Permit". So an unintended failure to match may allow  
3688 unintended **access**.

3689 Commercial http responders allow a variety of syntaxes to be treated equivalently. The "%" can be  
3690 used to represent characters by hex value. The URL path "/../" provides multiple ways of specifying  
3691 the same value. Multiple character sets may be permitted and, in some cases, the same printed  
3692 character can be represented by different binary values. Unless the matching algorithm used by  
3693 the policy is sophisticated enough to catch these variations, unintended access may be permitted.

3694 It may be safe to treat "NotApplicable" as "Permit" only in a closed environment where all  
3695 applications that formulate a **decision request** can be guaranteed to use the exact syntax  
3696 expected by the **policies**. In a more open environment, where **decision requests** may be received  
3697 from applications that use any legal syntax, it is strongly recommended that "NotApplicable" NOT  
3698 be treated as "Permit" unless matching rules have been very carefully designed to match all  
3699 possible applicable inputs, regardless of syntax or type variations. Note, however, that according to  
3700 Section 7.1, a **PEP** must deny **access** unless it receives an explicit "Permit" **authorization**  
3701 **decision**.

#### 3702 9.1.7. Negative rules

3703 A negative **rule** is one that is based on a **predicate** not being "True". If not used with care,  
3704 negative **rules** can lead to a policy violation, therefore some authorities recommend that they not

access\_control-xacml-2.0-core-spec-[cd-04](#)

Deleted: cd-03

3705 be used. However, negative **rules** can be extremely efficient in certain cases, so XACML has  
3706 chosen to include them. Nevertheless, it is recommended that they be used with care and avoided  
3707 if possible.

3708 A common use for negative **rules** is to deny **access** to an individual or subgroup when their  
3709 membership in a larger group would otherwise permit them access. For example, we might want to  
3710 write a **rule** that allows all Vice Presidents to see the unpublished financial data, except for Joe,  
3711 who is only a Ceremonial Vice President and can be indiscreet in his communications. If we have  
3712 complete control over the administration of **subject attributes**, a superior approach would be to  
3713 define "Vice President" and "Ceremonial Vice President" as distinct groups and then define **rules**  
3714 accordingly. However, in some environments this approach may not be feasible. (It is worth noting  
3715 in passing that, generally speaking, referring to individuals in **rules** does not scale well. Generally,  
3716 shared **attributes** are preferred.)

3717 If not used with care, negative **rules** can lead to policy violation in two common cases. They are:  
3718 when **attributes** are suppressed and when the base group changes. An example of suppressed  
3719 **attributes** would be if we have a policy that **access** should be permitted, *unless* the **subject** is a  
3720 credit risk. If it is possible that the **attribute** of being a credit risk may be unknown to the **PDP** for  
3721 some reason, then unauthorized **access** may result. In some environments, the **subject** may be  
3722 able to suppress the publication of **attributes** by the application of privacy controls, or the server or  
3723 repository that contains the information may be unavailable for accidental or intentional reasons.

3724 An example of a changing base group would be if there is a policy that everyone in the engineering  
3725 department may change software source code, except for secretaries. Suppose now that the  
3726 department was to merge with another engineering department and the intent is to maintain the  
3727 same policy. However, the new department also includes individuals identified as administrative  
3728 assistants, who ought to be treated in the same way as secretaries. Unless the policy is altered,  
3729 they will unintentionally be permitted to change software source code. Problems of this type are  
3730 easy to avoid when one individual administers all **policies**, but when administration is distributed,  
3731 as XACML allows, this type of situation must be explicitly guarded against.

## 3732 9.2. Safeguards

### 3733 9.2.1. Authentication

3734 Authentication provides the means for one party in a transaction to determine the identity of the  
3735 other party in the transaction. Authentication may be in one direction, or it may be bilateral.

3736 Given the sensitive nature of **access control** systems, it is important for a **PEP** to authenticate the  
3737 identity of the **PDP** to which it sends **decision requests**. Otherwise, there is a risk that an  
3738 adversary could provide false or invalid **authorization decisions**, leading to a policy violation.

3739 It is equally important for a **PDP** to authenticate the identity of the **PEP** and assess the level of trust  
3740 to determine what, if any, sensitive data should be passed. One should keep in mind that even  
3741 simple "Permit" or "Deny" responses could be exploited if an adversary were allowed to make  
3742 unlimited requests to a **PDP**.

3743 Many different techniques may be used to provide authentication, such as co-located code, a  
3744 private network, a VPN or digital signatures. Authentication may also be performed as part of the  
3745 communication protocol used to exchange the **contexts**. In this case, authentication may be  
3746 performed either at the message level or at the session level.

### 3747 9.2.2. Policy administration

3748 If the contents of **policies** are exposed outside of the **access control** system, potential **subjects**  
3749 may use this information to determine how to gain unauthorized **access**.

Deleted: cd-03

access\_control-xacml-2.0-core-spec-[cd-04](#)

3750 To prevent this threat, the repository used for the storage of **policies** may itself require **access**  
3751 **control**. In addition, the <Status> element should be used to return values of missing **attributes**  
3752 only when exposure of the identities of those **attributes** will not compromise security.

### 3753 9.2.3. Confidentiality

3754 Confidentiality mechanisms ensure that the contents of a message can be read only by the desired  
3755 recipients and not by anyone else who encounters the message while it is in transit. There are two  
3756 areas in which confidentiality should be considered: one is confidentiality during transmission; the  
3757 other is confidentiality within a <Policy> element.

#### 3758 9.2.3.1. Communication confidentiality

3759 In some environments it is deemed good practice to treat all data within an **access control** system  
3760 as confidential. In other environments, **policies** may be made freely available for distribution,  
3761 inspection and audit. The idea behind keeping **policy** information secret is to make it more difficult  
3762 for an adversary to know what steps might be sufficient to obtain unauthorized **access**. Regardless  
3763 of the approach chosen, the security of the **access control** system should not depend on the  
3764 secrecy of the **policy**.

3765 Any security considerations related to transmitting or exchanging XACML <Policy> elements are  
3766 outside the scope of the XACML standard. While it is often important to ensure that the integrity  
3767 and confidentiality of <Policy> elements is maintained when they are exchanged between two  
3768 parties, it is left to the implementers to determine the appropriate mechanisms for their  
3769 environment.

3770 Communications confidentiality can be provided by a confidentiality mechanism, such as SSL.  
3771 Using a point-to-point scheme like SSL may lead to other vulnerabilities when one of the end-points  
3772 is compromised.

#### 3773 9.2.3.2. Statement level confidentiality

3774 In some cases, an implementation may want to encrypt only parts of an XACML <Policy>  
3775 element.

3776 The XML Encryption Syntax and Processing Candidate Recommendation from W3C can be used  
3777 to encrypt all or parts of an XML document. This specification is recommended for use with  
3778 XACML.

3779 It should go without saying that if a repository is used to facilitate the communication of cleartext  
3780 (i.e., unencrypted) **policy** between the **PAP** and **PDP**, then a secure repository should be used to  
3781 store this sensitive data.

### 3782 9.2.4. Policy integrity

3783 The XACML **policy**, used by the **PDP** to evaluate the request **context**, is the heart of the system.  
3784 Therefore, maintaining its integrity is essential. There are two aspects to maintaining the integrity of  
3785 the **policy**. One is to ensure that <Policy> elements have not been altered since they were  
3786 originally created by the **PAP**. The other is to ensure that <Policy> elements have not been  
3787 inserted or deleted from the set of **policies**.

3788 In many cases, both aspects can be achieved by ensuring the integrity of the actors and  
3789 implementing session-level mechanisms to secure the communication between actors. The  
3790 selection of the appropriate mechanisms is left to the implementers. However, when **policy** is  
3791 distributed between organizations to be acted on at a later time, or when the **policy** travels with the

Deleted: cd-03

3792 protected resource, it would be useful to sign the **policy**. In these cases, the XML Signature  
3793 Syntax and Processing standard from W3C is recommended to be used with XACML.

3794 Digital signatures should only be used to ensure the integrity of the statements. Digital signatures  
3795 should not be used as a method of selecting or evaluating **policy**. That is, the **PDP** should not  
3796 request a **policy** based on who signed it or whether or not it has been signed (as such a basis for  
3797 selection would, itself, be a matter of policy). However, the **PDP** must verify that the key used to  
3798 sign the **policy** is one controlled by the purported issuer of the **policy**. The means to do this are  
3799 dependent on the specific signature technology chosen and are outside the scope of this document.

#### 3800 9.2.5. Policy identifiers

3801 Since **policies** can be referenced by their identifiers, it is the responsibility of the **PAP** to ensure  
3802 that these are unique. Confusion between identifiers could lead to misidentification of the  
3803 **applicable policy**. This specification is silent on whether a **PAP** must generate a new identifier  
3804 when a **policy** is modified or may use the same identifier in the modified **policy**. This is a matter of  
3805 administrative practice. However, care must be taken in either case. If the identifier is reused,  
3806 there is a danger that other **policies** or **policy sets** that reference it may be adversely affected.  
3807 Conversely, if a new identifier is used, these other **policies** may continue to use the prior **policy**,  
3808 unless it is deleted. In either case the results may not be what the **policy** administrator intends.

#### 3809 9.2.6. Trust model

3810 Discussions of authentication, integrity and confidentiality safeguards necessarily assume an  
3811 underlying trust model: how can one actor come to believe that a given key is uniquely associated  
3812 with a specific, identified actor so that the key can be used to encrypt data for that actor or verify  
3813 signatures (or other integrity structures) from that actor? Many different types of trust model exist,  
3814 including strict hierarchies, distributed authorities, the Web, the bridge and so on.

3815 It is worth considering the relationships between the various actors of the **access control** system in  
3816 terms of the interdependencies that do and do not exist.

- 3817 • None of the entities of the authorization system are dependent on the **PEP**. They may  
3818 collect data from it, for example authentication data, but are responsible for verifying it  
3819 themselves.
- 3820 • The correct operation of the system depends on the ability of the **PEP** to actually enforce  
3821 **policy** decisions.
- 3822 • The **PEP** depends on the **PDP** to correctly evaluate **policies**. This in turn implies that the  
3823 **PDP** is supplied with the correct inputs. Other than that, the **PDP** does not depend on the  
3824 **PEP**.
- 3825 • The **PDP** depends on the **PAP** to supply appropriate policies. The **PAP** is not dependent  
3826 on other components.

#### 3827 9.2.7. Privacy

3828 It is important to be aware that any transactions that occur with respect to **access control** may  
3829 reveal private information about the actors. For example, if an XACML **policy** states that certain  
3830 data may only be read by **subjects** with "Gold Card Member" status, then any transaction in which  
3831 a **subject** is permitted **access** to that data leaks information to an adversary about the **subject's**  
3832 status. Privacy considerations may therefore lead to encryption and/or to access control  
3833 requirements surrounding the enforcement of XACML **policy** instances themselves: confidentiality-  
3834 protected channels for the request/response protocol messages, protection of **subject attributes** in  
3835 storage and in transit, and so on.

Deleted: cd-03

3836 Selection and use of privacy mechanisms appropriate to a given environment are outside the scope  
3837 of XACML. The decision regarding whether, how and when to deploy such mechanisms is left to  
3838 the implementers associated with the environment.

## 3839 10. Conformance (normative)

### 3840 10.1. Introduction

3841 The XACML specification addresses the following aspect of conformance:

3842 The XACML specification defines a number of functions, etc. that have somewhat special  
3843 application, therefore they are not required to be implemented in an implementation that claims to  
3844 conform with the OASIS standard.

### 3845 10.2. Conformance tables

3846 This section lists those portions of the specification that MUST be included in an implementation of  
3847 a **PDP** that claims to conform with XACML v2.0. A set of test cases has been created to assist in  
3848 this process. These test cases are hosted by Sun Microsystems and can be located from the  
3849 XACML Web page. The site hosting the test cases contains a full description of the test cases and  
3850 how to execute them.

3851 Note: "M" means mandatory-to-implement. "O" means optional.

#### 3852 10.2.1. Schema elements

3853 The implementation MUST support those schema elements that are marked "M".

| Element name                         | M/O |
|--------------------------------------|-----|
| xacml-context:Action                 | M   |
| xacml-context:Attribute              | M   |
| xacml-context:AttributeValue         | M   |
| xacml-context:Decision               | M   |
| xacml-context:Environment            | M   |
| xacml-context:MissingAttributeDetail | M   |
| xacml-context:Obligations            | O   |
| xacml-context:Request                | M   |
| xacml-context:Resource               | M   |
| xacml-context:ResourceContent        | O   |
| xacml-context:Response               | M   |
| xacml-context:Result                 | M   |
| xacml-context:Status                 | M   |
| xacml-context:StatusCode             | M   |
| xacml-context:StatusDetail           | O   |
| xacml-context:StatusMessage          | O   |
| xacml-context:Subject                | M   |
| xacml:Action                         | M   |
| xacml:ActionAttributeDesignator      | M   |
| xacml:ActionMatch                    | M   |
| xacml:Actions                        | M   |
| xacml:Apply                          | M   |
| xacml:AttributeAssignment            | O   |
| xacml:AttributeSelector              | O   |
| xacml:AttributeValue                 | M   |
| xacml:CombinerParameters             | O   |

Deleted: cd-03



|                                      |   |
|--------------------------------------|---|
| xacml:CombinerParameter              | O |
| xacml:Condition                      | M |
| xacml:Description                    | M |
| xacml:Environment                    | M |
| xacml:EnvironmentMatch               | M |
| xacml:EnvironmentAttributeDesignator | M |
| xacml:Environments                   | M |
| xacml:Expression                     | M |
| xacml:Function                       | M |
| xacml:Obligation                     | O |
| xacml:Obligations                    | O |
| xacml:Policy                         | M |
| xacml:PolicyCombinerParameters       | O |
| xacml:PolicyDefaults                 | O |
| xacml:PolicyIdReference              | M |
| xacml:PolicySet                      | M |
| xacml:PolicySetDefaults              | O |
| xacml:PolicySetIdReference           | M |
| xacml:Resource                       | M |
| xacml:ResourceAttributeDesignator    | M |
| xacml:ResourceMatch                  | M |
| xacml:Resources                      | M |
| xacml:Rule                           | M |
| xacml:RuleCombinerParameters         | O |
| xacml:Subject                        | M |
| xacml:SubjectMatch                   | M |
| xacml:Subjects                       | M |
| xacml:Target                         | M |
| xacml:VariableDefinition             | M |
| xacml:VariableReference              | M |
| xacml:XPathVersion                   | O |

### 3854 10.2.2. Identifier Prefixes

3855 The following identifier prefixes are reserved by XACML.

| Identifier                                    |
|---|
| urn:oasis:names:tc:xacml:2.0                  |
| urn:oasis:names:tc:xacml:2.0:conformance-test |
| urn:oasis:names:tc:xacml:2.0:context          |
| urn:oasis:names:tc:xacml:2.0:example          |
| urn:oasis:names:tc:xacml:1.0:function         |
| urn:oasis:names:tc:xacml:2.0:function         |
| urn:oasis:names:tc:xacml:2.0:policy           |
| urn:oasis:names:tc:xacml:1.0:subject          |
| urn:oasis:names:tc:xacml:1.0:resource         |
| urn:oasis:names:tc:xacml:1.0:action           |
| urn:oasis:names:tc:xacml:1.0:environment      |
| urn:oasis:names:tc:xacml:1.0:status           |

### 3856 10.2.3. Algorithms

3857 The implementation MUST include the rule- and policy-combining algorithms associated with the  
3858 following identifiers that are marked "M".

3859

access\_control-xacml-2.0-core-spec-[cd-04](#)

Deleted: cd-03



| Algorithm  | M/O |
|--|-----|
| urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides             | M   |
| urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides           | M   |
| urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides           | M   |
| urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides         | M   |
| urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable           | M   |
| urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-applicable         | M   |
| urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:only-one-applicable      | M   |
| urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-overrides     | M   |
| urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-overrides   | M   |
| urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-overrides   | M   |
| urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-overrides | M   |

#### 10.2.4. Status Codes

Implementation support for the <StatusCode> element is optional, but if the element is supported, then the following status codes must be supported and must be used in the way XACML has specified.

| Identifier  | M/O |
|---|-----|
| urn:oasis:names:tc:xacml:1.0:status:missing-attribute | M   |
| urn:oasis:names:tc:xacml:1.0:status:ok                | M   |
| urn:oasis:names:tc:xacml:1.0:status:processing-error  | M   |
| urn:oasis:names:tc:xacml:1.0:status:syntax-error      | M   |

#### 10.2.5. Attributes

The implementation MUST support the **attributes** associated with the following identifiers as specified by XACML. If values for these **attributes** are not present in the **decision request**, then their values MUST be supplied by the **context handler**. So, unlike most other **attributes**, their semantics are not transparent to the **PDP**.

| Identifier  | M/O |
|---|-----|
| urn:oasis:names:tc:xacml:1.0:environment:current-time     | M   |
| urn:oasis:names:tc:xacml:1.0:environment:current-date     | M   |
| urn:oasis:names:tc:xacml:1.0:environment:current-dateTime | M   |

#### 10.2.6. Identifiers

The implementation MUST use the **attributes** associated with the following identifiers in the way XACML has defined. This requirement pertains primarily to implementations of a **PAP** or **PEP** that uses XACML, since the semantics of the attributes are transparent to the **PDP**.

Deleted: cd-03

| Identifier   | M/O |
|--|-----|
| urn:oasis:names:tc:xacml:1.0:subject:authn-locality:dns-name       | O   |
| urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address     | O   |
| urn:oasis:names:tc:xacml:1.0:subject:authentication-method         | O   |
| urn:oasis:names:tc:xacml:1.0:subject:authentication-time           | O   |
| urn:oasis:names:tc:xacml:1.0:subject:key-info                      | O   |
| urn:oasis:names:tc:xacml:1.0:subject:request-time                  | O   |
| urn:oasis:names:tc:xacml:1.0:subject:session-start-time            | O   |
| urn:oasis:names:tc:xacml:1.0:subject:subject-id                    | O   |
| urn:oasis:names:tc:xacml:1.0:subject:subject-id-qualifier          | O   |
| urn:oasis:names:tc:xacml:1.0:subject-category:access-subject       | M   |
| urn:oasis:names:tc:xacml:1.0:subject-category:codebase             | O   |
| urn:oasis:names:tc:xacml:1.0:subject-category:intermediary-subject | O   |
| urn:oasis:names:tc:xacml:1.0:subject-category:recipient-subject    | O   |
| urn:oasis:names:tc:xacml:1.0:subject-category:requesting-machine   | O   |
| urn:oasis:names:tc:xacml:1.0:resource:resource-location            | O   |
| urn:oasis:names:tc:xacml:1.0:resource:resource-id                  | M   |
| urn:oasis:names:tc:xacml:1.0:resource:simple-file-name             | O   |
| urn:oasis:names:tc:xacml:1.0:action:action-id                      | O   |
| urn:oasis:names:tc:xacml:1.0:action:implied-action                 | O   |

## 3877 10.2.7. Data-types

3878 The implementation MUST support the data-types associated with the following identifiers marked  
3879 "M".

| Data-type  | M/O |
|--|-----|
| http://www.w3.org/2001/XMLSchema#string                                  | M   |
| http://www.w3.org/2001/XMLSchema#boolean                                 | M   |
| http://www.w3.org/2001/XMLSchema#integer                                 | M   |
| http://www.w3.org/2001/XMLSchema#double                                  | M   |
| http://www.w3.org/2001/XMLSchema#time                                    | M   |
| http://www.w3.org/2001/XMLSchema#date                                    | M   |
| http://www.w3.org/2001/XMLSchema#dateTime                                | M   |
| http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration   | M   |
| http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration | M   |
| http://www.w3.org/2001/XMLSchema#anyURI                                  | M   |
| http://www.w3.org/2001/XMLSchema#hexBinary                               | M   |
| http://www.w3.org/2001/XMLSchema#base64Binary                            | M   |
| urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name                        | M   |
| urn:oasis:names:tc:xacml:1.0:data-type:x500Name                          | M   |

## 3880 10.2.8. Functions

3881 The implementation MUST properly process those functions associated with the identifiers marked  
3882 with an "M".

| Function  | M/O |
|---|-----|
| urn:oasis:names:tc:xacml:1.0:function:string-equal          | M   |
| urn:oasis:names:tc:xacml:1.0:function:boolean-equal         | M   |
| urn:oasis:names:tc:xacml:1.0:function:integer-equal         | M   |
| urn:oasis:names:tc:xacml:1.0:function:double-equal          | M   |
| urn:oasis:names:tc:xacml:1.0:function:date-equal            | M   |
| urn:oasis:names:tc:xacml:1.0:function:time-equal            | M   |
| urn:oasis:names:tc:xacml:1.0:function:dateTime-equal        | M   |
| urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-equal | M   |

Deleted: cd-03

access\_control-xacml-2.0-core-spec-[cd-04](#)

98

|   |   |
|---|---|
| urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-equal             | M |
| urn:oasis:names:tc:xacml:1.0:function:anyURI-equal                        | M |
| urn:oasis:names:tc:xacml:1.0:function:x500Name-equal                      | M |
| urn:oasis:names:tc:xacml:1.0:function:rfc822Name-equal                    | M |
| urn:oasis:names:tc:xacml:1.0:function:hexBinary-equal                     | M |
| urn:oasis:names:tc:xacml:1.0:function:base64Binary-equal                  | M |
| urn:oasis:names:tc:xacml:1.0:function:integer-add                         | M |
| urn:oasis:names:tc:xacml:1.0:function:double-add                          | M |
| urn:oasis:names:tc:xacml:1.0:function:integer-subtract                    | M |
| urn:oasis:names:tc:xacml:1.0:function:double-subtract                     | M |
| urn:oasis:names:tc:xacml:1.0:function:integer-multiply                    | M |
| urn:oasis:names:tc:xacml:1.0:function:double-multiply                     | M |
| urn:oasis:names:tc:xacml:1.0:function:integer-divide                      | M |
| urn:oasis:names:tc:xacml:1.0:function:double-divide                       | M |
| urn:oasis:names:tc:xacml:1.0:function:integer-mod                         | M |
| urn:oasis:names:tc:xacml:1.0:function:integer-abs                         | M |
| urn:oasis:names:tc:xacml:1.0:function:double-abs                          | M |
| urn:oasis:names:tc:xacml:1.0:function:round                               | M |
| urn:oasis:names:tc:xacml:1.0:function:floor                               | M |
| urn:oasis:names:tc:xacml:1.0:function:string-normalize-space              | M |
| urn:oasis:names:tc:xacml:1.0:function:string-normalize-to-lower-case      | M |
| urn:oasis:names:tc:xacml:1.0:function:double-to-integer                   | M |
| urn:oasis:names:tc:xacml:1.0:function:integer-to-double                   | M |
| urn:oasis:names:tc:xacml:1.0:function:or                                  | M |
| urn:oasis:names:tc:xacml:1.0:function:and                                 | M |
| urn:oasis:names:tc:xacml:1.0:function:n-of                                | M |
| urn:oasis:names:tc:xacml:1.0:function:not                                 | M |
| urn:oasis:names:tc:xacml:1.0:function:integer-greater-than                | M |
| urn:oasis:names:tc:xacml:1.0:function:integer-greater-than-or-equal       | M |
| urn:oasis:names:tc:xacml:1.0:function:integer-less-than                   | M |
| urn:oasis:names:tc:xacml:1.0:function:integer-less-than-or-equal          | M |
| urn:oasis:names:tc:xacml:1.0:function:double-greater-than                 | M |
| urn:oasis:names:tc:xacml:1.0:function:double-greater-than-or-equal        | M |
| urn:oasis:names:tc:xacml:1.0:function:double-less-than                    | M |
| urn:oasis:names:tc:xacml:1.0:function:double-less-than-or-equal           | M |
| urn:oasis:names:tc:xacml:1.0:function:dateTime-add-dayTimeDuration        | M |
| urn:oasis:names:tc:xacml:1.0:function:dateTime-add-yearMonthDuration      | M |
| urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-dayTimeDuration   | M |
| urn:oasis:names:tc:xacml:1.0:function:dateTime-subtract-yearMonthDuration | M |
| urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration          | M |
| urn:oasis:names:tc:xacml:1.0:function:date-subtract-yearMonthDuration     | M |
| urn:oasis:names:tc:xacml:1.0:function:string-greater-than                 | M |
| urn:oasis:names:tc:xacml:1.0:function:string-greater-than-or-equal        | M |
| urn:oasis:names:tc:xacml:1.0:function:string-less-than                    | M |
| urn:oasis:names:tc:xacml:1.0:function:string-less-than-or-equal           | M |
| urn:oasis:names:tc:xacml:1.0:function:time-greater-than                   | M |
| urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal          | M |
| urn:oasis:names:tc:xacml:1.0:function:time-less-than                      | M |
| urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal             | M |
| urn:oasis:names:tc:xacml:2.0:function:time-in-range                       | M |
| urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than               | M |
| urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than-or-equal      | M |
| urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than                  | M |
| urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than-or-equal         | M |
| urn:oasis:names:tc:xacml:1.0:function:date-greater-than                   | M |
| urn:oasis:names:tc:xacml:1.0:function:date-greater-than-or-equal          | M |

Deleted: cd-03

|  |   |
|--|---|
| urn:oasis:names:tc:xacml:1.0:function:date-less-than                 | M |
| urn:oasis:names:tc:xacml:1.0:function:date-less-than-or-equal        | M |
| urn:oasis:names:tc:xacml:1.0:function:string-one-and-only            | M |
| urn:oasis:names:tc:xacml:1.0:function:string-bag-size                | M |
| urn:oasis:names:tc:xacml:1.0:function:string-is-in                   | M |
| urn:oasis:names:tc:xacml:1.0:function:string-bag                     | M |
| urn:oasis:names:tc:xacml:1.0:function:boolean-one-and-only           | M |
| urn:oasis:names:tc:xacml:1.0:function:boolean-bag-size               | M |
| urn:oasis:names:tc:xacml:1.0:function:boolean-is-in                  | M |
| urn:oasis:names:tc:xacml:1.0:function:boolean-bag                    | M |
| urn:oasis:names:tc:xacml:1.0:function:integer-one-and-only           | M |
| urn:oasis:names:tc:xacml:1.0:function:integer-bag-size               | M |
| urn:oasis:names:tc:xacml:1.0:function:integer-is-in                  | M |
| urn:oasis:names:tc:xacml:1.0:function:integer-bag                    | M |
| urn:oasis:names:tc:xacml:1.0:function:double-one-and-only            | M |
| urn:oasis:names:tc:xacml:1.0:function:double-bag-size                | M |
| urn:oasis:names:tc:xacml:1.0:function:double-is-in                   | M |
| urn:oasis:names:tc:xacml:1.0:function:double-bag                     | M |
| urn:oasis:names:tc:xacml:1.0:function:time-one-and-only              | M |
| urn:oasis:names:tc:xacml:1.0:function:time-bag-size                  | M |
| urn:oasis:names:tc:xacml:1.0:function:time-is-in                     | M |
| urn:oasis:names:tc:xacml:1.0:function:time-bag                       | M |
| urn:oasis:names:tc:xacml:1.0:function:date-one-and-only              | M |
| urn:oasis:names:tc:xacml:1.0:function:date-bag-size                  | M |
| urn:oasis:names:tc:xacml:1.0:function:date-is-in                     | M |
| urn:oasis:names:tc:xacml:1.0:function:date-bag                       | M |
| urn:oasis:names:tc:xacml:1.0:function:dateTime-one-and-only          | M |
| urn:oasis:names:tc:xacml:1.0:function:dateTime-bag-size              | M |
| urn:oasis:names:tc:xacml:1.0:function:dateTime-is-in                 | M |
| urn:oasis:names:tc:xacml:1.0:function:dateTime-bag                   | M |
| urn:oasis:names:tc:xacml:1.0:function:anyURI-one-and-only            | M |
| urn:oasis:names:tc:xacml:1.0:function:anyURI-bag-size                | M |
| urn:oasis:names:tc:xacml:1.0:function:anyURI-is-in                   | M |
| urn:oasis:names:tc:xacml:1.0:function:anyURI-bag                     | M |
| urn:oasis:names:tc:xacml:1.0:function:hexBinary-one-and-only         | M |
| urn:oasis:names:tc:xacml:1.0:function:hexBinary-bag-size             | M |
| urn:oasis:names:tc:xacml:1.0:function:hexBinary-is-in                | M |
| urn:oasis:names:tc:xacml:1.0:function:hexBinary-bag                  | M |
| urn:oasis:names:tc:xacml:1.0:function:base64Binary-one-and-only      | M |
| urn:oasis:names:tc:xacml:1.0:function:base64Binary-bag-size          | M |
| urn:oasis:names:tc:xacml:1.0:function:base64Binary-is-in             | M |
| urn:oasis:names:tc:xacml:1.0:function:base64Binary-bag               | M |
| urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-one-and-only   | M |
| urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-bag-size       | M |
| urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-is-in          | M |
| urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-bag            | M |
| urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-one-and-only | M |
| urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-bag-size     | M |
| urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-is-in        | M |
| urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-bag          | M |
| urn:oasis:names:tc:xacml:1.0:function:x500Name-one-and-only          | M |
| urn:oasis:names:tc:xacml:1.0:function:x500Name-bag-size              | M |
| urn:oasis:names:tc:xacml:1.0:function:x500Name-is-in                 | M |
| urn:oasis:names:tc:xacml:1.0:function:x500Name-bag                   | M |
| urn:oasis:names:tc:xacml:1.0:function:rfc822Name-one-and-only        | M |
| urn:oasis:names:tc:xacml:1.0:function:rfc822Name-bag-size            | M |
| urn:oasis:names:tc:xacml:1.0:function:rfc822Name-is-in               | M |

Deleted: cd-03

|   |   |                            |
|---|---|----------------------------|
| urn:oasis:names:tc:xacml:1.0:function:rfc822Name-bag                  | M |                            |
| urn:oasis:names:tc:xacml:2.0:function:string-concatenate              | M |                            |
| urn:oasis:names:tc:xacml:2.0:function:uri-string-concatenate          | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:any-of                          | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:all-of                          | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:any-of-any                      | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:all-of-any                      | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:any-of-all                      | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:all-of-all                      | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:map                             | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:x500Name-match                  | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match                | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:string-regexp-match             | M | Deleted: regexp-string     |
| urn:oasis:names:tc:xacml:2.0:function:anyURI-regexp-match             | M | Deleted: regexp-anyURI     |
| urn:oasis:names:tc:xacml:2.0:function:ipAddress-regexp-match          | M | Deleted: regexp-ipAddress  |
| urn:oasis:names:tc:xacml:2.0:function:domainName-regexp-match         | M | Deleted: regexp-dnsName    |
| urn:oasis:names:tc:xacml:2.0:function:rfc822Name-regexp-match         | M | Deleted: regexp-rfc822Name |
| urn:oasis:names:tc:xacml:2.0:function:x500Name-regexp-match           | M | Deleted: regexp-           |
| urn:oasis:names:tc:xacml:1.0:function:xpath-node-count                | O |                            |
| urn:oasis:names:tc:xacml:1.0:function:xpath-node-equal                | O |                            |
| urn:oasis:names:tc:xacml:1.0:function:xpath-node-match                | O |                            |
| urn:oasis:names:tc:xacml:1.0:function:string-intersection             | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:string-at-least-one-member-of   | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:string-union                    | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:string-subset                   | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:string-set-equals               | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:boolean-intersection            | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:boolean-at-least-one-member-of  | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:boolean-union                   | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:boolean-subset                  | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:boolean-set-equals              | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:integer-intersection            | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:integer-at-least-one-member-of  | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:integer-union                   | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:integer-subset                  | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:integer-set-equals              | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:double-intersection             | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:double-at-least-one-member-of   | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:double-union                    | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:double-subset                   | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:double-set-equals               | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:time-intersection               | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:time-at-least-one-member-of     | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:time-union                      | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:time-subset                     | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:time-set-equals                 | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:date-intersection               | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:date-at-least-one-member-of     | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:date-union                      | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:date-subset                     | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:date-set-equals                 | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:dateTime-intersection           | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:dateTime-at-least-one-member-of | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:dateTime-union                  | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:dateTime-subset                 | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:dateTime-set-equals             | M |                            |
| urn:oasis:names:tc:xacml:1.0:function:anyURI-intersection             | M | Deleted: cd-03             |

|  |   |
|--|---|
| urn:oasis:names:tc:xacml:1.0:function:anyURI-at-least-one-member-of            | M |
| urn:oasis:names:tc:xacml:1.0:function:anyURI-union                             | M |
| urn:oasis:names:tc:xacml:1.0:function:anyURI-subset                            | M |
| urn:oasis:names:tc:xacml:1.0:function:anyURI-set-equals                        | M |
| urn:oasis:names:tc:xacml:1.0:function:hexBinary-intersection                   | M |
| urn:oasis:names:tc:xacml:1.0:function:hexBinary-at-least-one-member-of         | M |
| urn:oasis:names:tc:xacml:1.0:function:hexBinary-union                          | M |
| urn:oasis:names:tc:xacml:1.0:function:hexBinary-subset                         | M |
| urn:oasis:names:tc:xacml:1.0:function:hexBinary-set-equals                     | M |
| urn:oasis:names:tc:xacml:1.0:function:base64Binary-intersection                | M |
| urn:oasis:names:tc:xacml:1.0:function:base64Binary-at-least-one-member-of      | M |
| urn:oasis:names:tc:xacml:1.0:function:base64Binary-union                       | M |
| urn:oasis:names:tc:xacml:1.0:function:base64Binary-subset                      | M |
| urn:oasis:names:tc:xacml:1.0:function:base64Binary-set-equals                  | M |
| urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-intersection             | M |
| urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-at-least-one-member-of   | M |
| urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-union                    | M |
| urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-subset                   | M |
| urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-set-equals               | M |
| urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-intersection           | M |
| urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-at-least-one-member-of | M |
| urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-union                  | M |
| urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-subset                 | M |
| urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-set-equals             | M |
| urn:oasis:names:tc:xacml:1.0:function:x500Name-intersection                    | M |
| urn:oasis:names:tc:xacml:1.0:function:x500Name-at-least-one-member-of          | M |
| urn:oasis:names:tc:xacml:1.0:function:x500Name-union                           | M |
| urn:oasis:names:tc:xacml:1.0:function:x500Name-subset                          | M |
| urn:oasis:names:tc:xacml:1.0:function:x500Name-set-equals                      | M |
| urn:oasis:names:tc:xacml:1.0:function:rfc822Name-intersection                  | M |
| urn:oasis:names:tc:xacml:1.0:function:rfc822Name-at-least-one-member-of        | M |
| urn:oasis:names:tc:xacml:1.0:function:rfc822Name-union                         | M |
| urn:oasis:names:tc:xacml:1.0:function:rfc822Name-subset                        | M |
| urn:oasis:names:tc:xacml:1.0:function:rfc822Name-set-equals                    | M |

3883

## 11. References

3884

- 3885 [DS] D. Eastlake et al., *XML-Signature Syntax and Processing*,  
3886 <http://www.w3.org/TR/xmldsig-core/>, World Wide Web Consortium.
- 3887 [Hancock] Hancock, "Polymorphic Type Checking", in Simon L. Peyton Jones,  
3888 "Implementation of Functional Programming Languages", Section 8,  
3889 Prentice-Hall International, 1987
- 3890 [Haskell] Haskell, a purely functional language. Available at  
3891 <http://www.haskell.org/>
- 3892 [Hier] Anderson A, ed., *The XACML Profile for Hierarchical Resources*, OASIS  
3893 Access Control TC, Committee Draft 01, 16 Sep 2004, <http://www.oasis-open.org/committees/xacml>
- 3895 [Hinton94] Hinton, H, M, Lee, E, S, The Compatibility of Policies, Proceedings 2nd  
3896 ACM Conference on Computer and Communications Security, Nov 1994,  
3897 Fairfax, Virginia, USA.

Deleted: cd-03

|      |              |  |
|------|--------------|--|
| 3898 | [IEEE754]    | IEEE Standard for Binary Floating-Point Arithmetic 1985, ISBN 1-5593-7653-8, IEEE Product No. SH10116-TBR  |
| 3899 |              |  |
| 3900 | [ISO10181-3] | ISO/IEC 10181-3:1996 Information technology – Open Systems Interconnection -- Security frameworks for open systems: Access control framework.  |
| 3901 |              |  |
| 3902 |              |  |
| 3903 | [Kudo00]     | Kudo M and Hada S, XML document security based on provisional authorization, Proceedings of the Seventh ACM Conference on Computer and Communications Security, Nov 2000, Athens, Greece, pp 87-96.  |
| 3904 |              |  |
| 3905 |              |  |
| 3906 | [LDAP-1]     | RFC2256, A summary of the X500(96) User Schema for use with LDAPv3, Section 5, M Wahl, December 1997 <a href="http://www.ietf.org/rfc/rfc2798.txt">http://www.ietf.org/rfc/rfc2798.txt</a>   |
| 3907 |              |  |
| 3908 | [LDAP-2]     | RFC2798, Definition of the inetOrgPerson, M. Smith, April 2000 <a href="http://www.ietf.org/rfc/rfc2798.txt">http://www.ietf.org/rfc/rfc2798.txt</a>   |
| 3909 |              |  |
| 3910 | [MathML]     | Mathematical Markup Language (MathML), Version 2.0, W3C Recommendation, 21 February 2001. Available at: <a href="http://www.w3.org/TR/MathML2/">http://www.w3.org/TR/MathML2/</a>  |
| 3911 |              |  |
| 3912 |              |  |
| 3913 | [Multi]      | Anderson A, ed., <i>The XACML Profile for Requests for Multiple Resources</i> , OASIS Access Control TC, Working Draft 02, 4 June 2004, <a href="http://www.oasis-open.org/committees/xacml">http://www.oasis-open.org/committees/xacml</a>  |
| 3914 |              |  |
| 3915 |              |  |
| 3916 | [Perritt93]  | Perritt, H. Knowbots, Permissions Headers and Contract Law, Conference on Technological Strategies for Protecting Intellectual Property in the Networked Multimedia Environment, April 1993. Available at: <a href="http://www.ifla.org/documents/infopol/copyright/perh2.txt">http://www.ifla.org/documents/infopol/copyright/perh2.txt</a>     |
| 3917 |              |  |
| 3918 |              |  |
| 3919 |              |  |
| 3920 | [RBAC]       | Role-Based Access Controls, David Ferraiole and Richard Kuhn, 15th National Computer Security Conference, 1992. Available at: <a href="http://csrc.nist.gov/rbac">http://csrc.nist.gov/rbac</a>  |
| 3921 |              |  |
| 3922 |              |  |
| 3923 | [RegEx]      | XML Schema Part 0: Primer, W3C Recommendation, 2 May 2001, Appendix D. Available at: <a href="http://www.w3.org/TR/xmlschema-0/">http://www.w3.org/TR/xmlschema-0/</a>   |
| 3924 |              |  |
| 3925 | [RFC2119]    | S. Bradner, <i>Key words for use in RFCs to Indicate Requirement Levels</i> , <a href="http://www.ietf.org/rfc/rfc2119.txt">http://www.ietf.org/rfc/rfc2119.txt</a> , IETF RFC 2119, March 1997  |
| 3926 |              |  |
| 3927 | [RFC2396]    | Berners-Lee T, Fielding R, Masinter L, Uniform Resource Identifiers (URI): Generic Syntax. Available at: <a href="http://www.ietf.org/rfc/rfc2396.txt">http://www.ietf.org/rfc/rfc2396.txt</a>   |
| 3928 |              |  |
| 3929 | [RFC2732]    | Hinden R, Carpenter B, Masinter L, Format for Literal IPv6 Addresses in URL's. Available at: <a href="http://www.ietf.org/rfc/rfc2732.txt">http://www.ietf.org/rfc/rfc2732.txt</a>   |
| 3930 |              |  |
| 3931 | [RFC3198]    | IETF RFC 3198: Terminology for Policy-Based Management, November 2001. <a href="http://www.ietf.org/rfc/rfc3198.txt">http://www.ietf.org/rfc/rfc3198.txt</a>   |
| 3932 |              |  |
| 3933 | [SAML]       | Security Assertion Markup Language available from <a href="http://www.oasis-open.org/committees/security/#documents">http://www.oasis-open.org/committees/security/#documents</a>  |
| 3934 |              |  |
| 3935 | [Sloman94]   | Sloman, M. Policy Driven Management for Distributed Systems. Journal of Network and Systems Management, Volume 2, part 4. Plenum Press. 1994.  |
| 3936 |              |  |
| 3937 |              |  |
| 3938 | [XACMLv1.0]  | Extensible access control markup language (XACML) Version 1.0. OASIS Standard. 18 February 2003. Available at: <a href="http://www.oasis-open.org/apps/org/workgroup/xacml/download.php/940/oasis-xacml-1.0.pdf">http://www.oasis-open.org/apps/org/workgroup/xacml/download.php/940/oasis-xacml-1.0.pdf</a>                                     |
| 3939 |              |  |
| 3940 |              |  |
| 3941 |              |  |
| 3942 | [XACMLv1.1]  | Extensible access control markup language (XACML) Version 1.1. OASIS Committee Specification. 7 August 2003. Available at: <a href="http://www.oasis-open.org/apps/org/workgroup/xacml/download.php/4104/cs-xacml-specification-1.1.pdf">http://www.oasis-open.org/apps/org/workgroup/xacml/download.php/4104/cs-xacml-specification-1.1.pdf</a> |
| 3943 |              |  |
| 3944 |              |  |
| 3945 |              |  |

Deleted: cd-03



|      |                |   |
|------|----------------|---|
| 3946 | <b>[XF]</b>    | XQuery 1.0 and XPath 2.0 Functions and Operators, W3C Working Draft 16 August 2002. Available at: <a href="http://www.w3.org/TR/2002/WD-xquery-operators-20020816">http://www.w3.org/TR/2002/WD-xquery-operators-20020816</a> |
| 3947 |                |   |
| 3948 |                |   |
| 3949 | <b>[XS]</b>    | XML Schema, parts 1 and 2. Available at: <a href="http://www.w3.org/TR/xmlschema-1/">http://www.w3.org/TR/xmlschema-1/</a> and <a href="http://www.w3.org/TR/xmlschema-2/">http://www.w3.org/TR/xmlschema-2/</a>              |
| 3950 |                |   |
| 3951 |                |   |
| 3952 | <b>[XPath]</b> | XML Path Language (XPath), Version 1.0, W3C Recommendation 16 November 1999. Available at: <a href="http://www.w3.org/TR/xpath">http://www.w3.org/TR/xpath</a>  |
| 3953 |                |   |
| 3954 | <b>[XSLT]</b>  | XSL Transformations (XSLT) Version 1.0, W3C Recommendation 16 November 1999. Available at: <a href="http://www.w3.org/TR/xslt">http://www.w3.org/TR/xslt</a>  |
| 3955 |                |   |
| 3956 |                |   |



# Appendix A. Data-types and functions (normative)

## A.1. Introduction

This section specifies the data-types and functions used in XACML to create *predicates* for *conditions* and *target* matches.

This specification combines the various standards set forth by IEEE and ANSI for string representation of numeric values, as well as the evaluation of arithmetic functions. It describes the primitive data-types and *bags*. The standard functions are named and their operational semantics are described.

## A.2. Data-types

Although XML instances represent all data-types as strings, an XACML *PDP* must reason about types of data that, while they have string representations, are not just strings. Types such as Boolean, integer and double MUST be converted from their XML string representations to values that can be compared with values in their domain of discourse, such as numbers. The following primitive data-types are specified for use with XACML and have explicit data representations:

- <http://www.w3.org/2001/XMLSchema#string>
- <http://www.w3.org/2001/XMLSchema#boolean>
- <http://www.w3.org/2001/XMLSchema#integer>
- <http://www.w3.org/2001/XMLSchema#double>
- <http://www.w3.org/2001/XMLSchema#time>
- <http://www.w3.org/2001/XMLSchema#date>
- <http://www.w3.org/2001/XMLSchema#dateTime>
- <http://www.w3.org/2001/XMLSchema#anyURI>
- <http://www.w3.org/2001/XMLSchema#hexBinary>
- <http://www.w3.org/2001/XMLSchema#base64Binary>
- <http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration>
- <http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration>
- <urn:oasis:names:tc:xacml:1.0:data-type:x500Name>
- <urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name>
- <urn:oasis:names:tc:xacml:2.0:data-type:ipAddress>
- <urn:oasis:names:tc:xacml:2.0:data-type:dnsName>

For the sake of improved interoperability, it is RECOMMENDED that all time references be in UTC time.

Deleted: cd-03

3989 An XACML **PDP** SHALL be capable of converting string representations into various primitive data-  
3990 types. For integers and doubles, XACML SHALL use the conversions described in [IEEE754].

3991 XACML defines three data-types; these are:

3992 "urn:oasis:names:tc:xacml:1.0:data-type:x500Name",

3993 "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name"

3994 "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress"

3995 "urn:oasis:names:tc:xacml:2.0:data-type:dnsName" and

3996 These types represent identifiers for subjects or resources and appear in several standard  
3997 applications, such as TLS/SSL and electronic mail.

#### 3998 **X.500 directory name**

3999 The "urn:oasis:names:tc:xacml:1.0:data-type:x500Name" primitive type represents an ITU-T Rec.  
4000 X.520 Distinguished Name. The valid syntax for such a name is described in IETF RFC 2253  
4001 "Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names"

#### 4002 **RFC 822 name**

4003 The "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" primitive type represents an electronic  
4004 mail address. The valid syntax for such a name is described in IETF RFC 2821, Section 4.1.2,  
4005 Command Argument Syntax, under the term "Mailbox".

#### 4006 **IP address**

4007 The "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress" primitive type represents an IPv4 or IPv6  
4008 network address, with optional mask and optional port or port range. The syntax SHALL be:

4009  
4010 ipAddress = address [ "/" mask ] [ ":" [ portrange ] ]  
4011

4012 For an IPv4 address, the address and mask are formatted in accordance with the syntax for a  
4013 "host" in IETF RFC 2396 "Uniform Resource Identifiers (URI): Generic Syntax", section 3.2.  
4014 For an IPv6 address, the address and mask are formatted in accordance with the syntax for an  
4015 "ipv6reference" in IETF RFC 2732 "Format for Literal IPv6 Addresses in URL's". (Note that an IPv6  
4016 address or mask, in this syntax, is enclosed in literal "[" "]" brackets.)  
4017

#### 4018 **DNS name**

4019 The "urn:oasis:names:tc:xacml:2.0:data-type:dnsName" primitive type represents a Domain Name  
4020 Service (DNS) host name, with optional port or port range. The syntax SHALL be:

4021  
4022 dnsName = hostname [ ":" portrange ]  
4023

4024 The hostname is formatted in accordance with IETF RFC 2396 "Uniform Resource Identifiers (URI):  
4025 Generic Syntax", section 3.2, except that a wildcard "\*" may be used in the left-most component of  
4026 the hostname to indicate "any subdomain" under the domain specified to its right.  
4027

4028 For both the "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress" and  
4029 "urn:oasis:names:tc:xacml:2.0:data-type:dnsName" data-types, the port or port range syntax  
4030 SHALL be

4031  
4032 portrange = portnumber | "-"portnumber | portnumber-"["portnumber]  
4033

4034 where "portnumber" is a decimal port number. If the port number is of the form "-x", where "x" is a  
4035 port number, then the range is all ports numbered "x" and below. If the port number is of the form

Deleted: cd-03

4036 "x-", then the range is all ports numbered "x" and above. [This syntax is taken from the Java  
4037 SocketPermission.]

## 4038 A.3. Functions

4039 XACML specifies the following functions. If an argument of one of these functions were to evaluate  
4040 to "Indeterminate", then the function SHALL be set to "Indeterminate".

### 4041 A.3.1 Equality predicates

4042 The following functions are the *equality* functions for the various primitive types. Each function for a  
4043 particular data-type follows a specified standard convention for that data-type.

- 4044 • urn:oasis:names:tc:xacml:1.0:function:string-equal

4045 This function SHALL take two arguments of data-type  
4046 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an  
4047 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL return "True" if and  
4048 only if the value of both of its arguments are of equal length and each string is determined  
4049 to be equal byte-by-byte according to the function "integer-equal". Otherwise, it SHALL  
4050 return "False".

- 4051 • urn:oasis:names:tc:xacml:1.0:function:boolean-equal

4052 This function SHALL take two arguments of data-type  
4053 "http://www.w3.org/2001/XMLSchema#boolean" and SHALL return an  
4054 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL return "True" if and  
4055 only if the arguments are equal. Otherwise, it SHALL return "False".

- 4056 • urn:oasis:names:tc:xacml:1.0:function:integer-equal

4057 This function SHALL take two arguments of data-type  
4058 "http://www.w3.org/2001/XMLSchema#integer" and SHALL return an  
4059 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation on  
4060 integers according to IEEE 754 [IEEE 754].

- 4061 • urn:oasis:names:tc:xacml:1.0:function:double-equal

4062 This function SHALL take two arguments of data-type  
4063 "http://www.w3.org/2001/XMLSchema#double" and SHALL return an  
4064 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation on  
4065 doubles according to IEEE 754 [IEEE 754].

- 4066 • urn:oasis:names:tc:xacml:1.0:function:date-equal

4067 This function SHALL take two arguments of data-type  
4068 "http://www.w3.org/2001/XMLSchema#date" and SHALL return an  
4069 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation  
4070 according to the "op:date-equal" function [XF Section 8.3.11].

- 4071 • urn:oasis:names:tc:xacml:1.0:function:time-equal

4072 This function SHALL take two arguments of data-type  
4073 "http://www.w3.org/2001/XMLSchema#time" and SHALL return an  
4074 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation  
4075 according to the "op:time-equal" function [XF Section 8.3.14].

Deleted: cd-03

4076 • urn:oasis:names:tc:xacml:1.0:function:dateTime-equal

4077 This function SHALL take two arguments of data-type  
 4078 "http://www.w3.org/2001/XMLSchema#dateTime" and SHALL return an  
 4079 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation  
 4080 according to the "op:dateTime-equal" function [XF Section 8.3.8].

4081 • urn:oasis:names:tc:xacml:1.0:function:dayTimeDuration-equal

4082 This function SHALL take two arguments of data-type "http://www.w3.org/TR/2002/WD-  
 4083 xquery-operators-20020816#dayTimeDuration" and SHALL return an  
 4084 "http://www.w3.org/2001/XMLSchema#boolean". This function shall perform its evaluation  
 4085 according to the "op:dayTimeDuration-equal" function [XF Section 8.3.5]. Note that the  
 4086 lexical representation of each argument MUST be converted to a value expressed in  
 4087 fractional seconds [XF Section 8.2.2].

4088 • urn:oasis:names:tc:xacml:1.0:function:yearMonthDuration-equal

4089 This function SHALL take two arguments of data-type "http://www.w3.org/TR/2002/WD-  
 4090 xquery-operators-20020816#yearMonthDuration" and SHALL return an  
 4091 "http://www.w3.org/2001/XMLSchema#boolean". This function shall perform its evaluation  
 4092 according to the "op:yearMonthDuration-equal" function [XF Section 8.3.2]. Note that the  
 4093 lexical representation of each argument MUST be converted to a value expressed in  
 4094 integer months [XF Section 8.2.1].

4095 • urn:oasis:names:tc:xacml:1.0:function:anyURI-equal

4096 This function SHALL take two arguments of data-type  
 4097 "http://www.w3.org/2001/XMLSchema#anyURI" and SHALL return an  
 4098 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL perform its evaluation  
 4099 according to the "op:anyURI-equal" function [XF Section 10.2.1].

4100 • urn:oasis:names:tc:xacml:1.0:function:x500Name-equal

4101 This function SHALL take two arguments of "urn:oasis:names:tc:xacml:1.0:data-  
 4102 type:x500Name" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". It  
 4103 SHALL return "True" if and only if each Relative Distinguished Name (RDN) in the two  
 4104 arguments matches. Otherwise, it SHALL return "False". Two RDNs shall be said to  
 4105 match if and only if the result of the following operations is "True"<sup>3</sup>.

4106 1. Normalize the two arguments according to IETF RFC 2253 "Lightweight Directory  
 4107 Access Protocol (v3): UTF-8 String Representation of Distinguished Names".

4108 2. If any RDN contains multiple attributeTypeAndValue pairs, re-order the Attribute  
 4109 ValuePairs in that RDN in ascending order when compared as octet strings  
 4110 (described in ITU-T Rec. X.690 (1997 E) Section 11.6 "Set-of components").

4111 3. Compare RDNs using the rules in IETF RFC 3280 "Internet X.509 Public Key  
 4112 Infrastructure Certificate and Certificate Revocation List (CRL) Profile", Section  
 4113 4.1.2.4 "Issuer".

4114 • urn:oasis:names:tc:xacml:1.0:function:rfc822Name-equal

4115 This function SHALL take two arguments of data-type "urn:oasis:names:tc:xacml:1.0:data-  
 4116 type:rfc822Name" and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean".  
 4117 It SHALL return "True" if and only if the two arguments are equal. Otherwise, it SHALL

<sup>3</sup> ITU-T Rec. X.520 contains rules for matching X500 names, but these are very complex and require knowledge of the syntax of various AttributeTypes. IETF RFC 3280 contains simplified matching rules that the XACML x500Name-equal function uses.

Deleted: cd-03

4118 return "False". An RFC822 name consists of a *local-part* followed by "@" followed by a  
 4119 *domain-part*. The *local-part* is case-sensitive, while the *domain-part* (which is usually a  
 4120 DNS host name) is not case-sensitive. Perform the following operations:

- 4121 1. Normalize the *domain-part* of each argument to lower case
- 4122 2. Compare the expressions by applying the function  
 4123 "urn:oasis:names:tc:xacml:1.0:function:string-equal" to the normalized arguments.
- 4124 • urn:oasis:names:tc:xacml:1.0:function:hexBinary-equal

4125 This function SHALL take two arguments of data-type  
 4126 "http://www.w3.org/2001/XMLSchema#hexBinary" and SHALL return an  
 4127 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if the octet  
 4128 sequences represented by the value of both arguments have equal length and are equal in  
 4129 a conjunctive, point-wise, comparison using the  
 4130 "urn:oasis:names:tc:xacml:1.0:function:integer-equal" function. Otherwise, it SHALL return  
 4131 "False". The conversion from the string representation to an octet sequence SHALL be as  
 4132 specified in [XS Section 8.2.15].

- 4133 • urn:oasis:names:tc:xacml:1.0:function:base64Binary-equal

4134 This function SHALL take two arguments of data-type  
 4135 "http://www.w3.org/2001/XMLSchema#base64Binary" and SHALL return an  
 4136 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if the octet  
 4137 sequences represented by the value of both arguments have equal length and are equal in  
 4138 a conjunctive, point-wise, comparison using the  
 4139 "urn:oasis:names:tc:xacml:1.0:function:integer-equal" function. Otherwise, it SHALL return  
 4140 "False". The conversion from the string representation to an octet sequence SHALL be as  
 4141 specified in [XS Section 8.2.16].

### 4142 A.3.2 Arithmetic functions

4143 All of the following functions SHALL take two arguments of the specified *data-type*, integer or  
 4144 double, and SHALL return an element of integer or double data-type, respectively. However, the  
 4145 "add" functions MAY take more than two arguments. Each function evaluation SHALL proceed as  
 4146 specified by their logical counterparts in IEEE 754 [IEEE 754]. In an expression that contains any  
 4147 of these functions, if any argument is "Indeterminate", then the expression SHALL evaluate to  
 4148 "Indeterminate". In the case of the divide functions, if the divisor is zero, then the function SHALL  
 4149 evaluate to "Indeterminate".

- 4150 • urn:oasis:names:tc:xacml:1.0:function:integer-add
- 4151 This function MAY have two or more arguments.
- 4152 • urn:oasis:names:tc:xacml:1.0:function:double-add
- 4153 This function MAY have two or more arguments.
- 4154 • urn:oasis:names:tc:xacml:1.0:function:integer-subtract
- 4155 • urn:oasis:names:tc:xacml:1.0:function:double-subtract
- 4156 • urn:oasis:names:tc:xacml:1.0:function:integer-multiply
- 4157 • urn:oasis:names:tc:xacml:1.0:function:double-multiply
- 4158 • urn:oasis:names:tc:xacml:1.0:function:integer-divide
- 4159 • urn:oasis:names:tc:xacml:1.0:function:double-divide

Deleted: cd-03

4160 • urn:oasis:names:tc:xacml:1.0:function:integer-mod

4161 The following functions SHALL take a single argument of the specified *data-type*. The round and  
4162 floor functions SHALL take a single argument of data-type  
4163 "http://www.w3.org/2001/XMLSchema#double" and return a value of the data-type  
4164 "http://www.w3.org/2001/XMLSchema#double".

4165 • urn:oasis:names:tc:xacml:1.0:function:integer-abs

4166 • urn:oasis:names:tc:xacml:1.0:function:double-abs

4167 • urn:oasis:names:tc:xacml:1.0:function:round

4168 • urn:oasis:names:tc:xacml:1.0:function:floor

### 4169 A.3.3 String conversion functions

4170 The following functions convert between values of the data-type  
4171 "http://www.w3.org/2001/XMLSchema#string" primitive types.

4172 • urn:oasis:names:tc:xacml:1.0:function:string-normalize-space

4173 This function SHALL take one argument of data-type  
4174 "http://www.w3.org/2001/XMLSchema#string" and SHALL normalize the value by stripping  
4175 off all leading and trailing white space characters.

4176 • urn:oasis:names:tc:xacml:1.0:function:string-normalize-to-lower-case

4177 This function SHALL take one argument of data-type  
4178 "http://www.w3.org/2001/XMLSchema#string" and SHALL normalize the value by  
4179 converting each upper case character to its lower case equivalent.

### 4180 A.3.4 Numeric data-type conversion functions

4181 The following functions convert between the data-type  
4182 "http://www.w3.org/2001/XMLSchema#integer" and "http://www.w3.org/2001/XMLSchema#double"  
4183 primitive types.

4184 • urn:oasis:names:tc:xacml:1.0:function:double-to-integer

4185 This function SHALL take one argument of data-type  
4186 "http://www.w3.org/2001/XMLSchema#double" and SHALL truncate its numeric value to a  
4187 whole number and return an element of data-type  
4188 "http://www.w3.org/2001/XMLSchema#integer".

4189 • urn:oasis:names:tc:xacml:1.0:function:integer-to-double

4190 This function SHALL take one argument of data-type  
4191 "http://www.w3.org/2001/XMLSchema#integer" and SHALL promote its value to an element  
4192 of data-type "http://www.w3.org/2001/XMLSchema#double" with the same numeric value.

### 4193 A.3.5 Logical functions

4194 This section contains the specification for logical functions that operate on arguments of data-type  
4195 "http://www.w3.org/2001/XMLSchema#boolean".

4196 • urn:oasis:names:tc:xacml:1.0:function:or

Deleted: cd-03

4197 This function SHALL return "False" if it has no arguments and SHALL return "True" if at  
4198 least one of its arguments evaluates to "True". The order of evaluation SHALL be from first  
4199 argument to last. The evaluation SHALL stop with a result of "True" if any argument  
4200 evaluates to "True", leaving the rest of the arguments unevaluated.

4201 • urn:oasis:names:tc:xacml:1.0:function:and

4202 This function SHALL return "True" if it has no arguments and SHALL return "False" if one of  
4203 its arguments evaluates to "False". The order of evaluation SHALL be from first argument  
4204 to last. The evaluation SHALL stop with a result of "False" if any argument evaluates to  
4205 "False", leaving the rest of the arguments unevaluated.

4206 • urn:oasis:names:tc:xacml:1.0:function:n-of

4207 The first argument to this function SHALL be of data-type  
4208 <http://www.w3.org/2001/XMLSchema#integer>. The remaining arguments SHALL be of  
4209 data-type <http://www.w3.org/2001/XMLSchema#boolean>. The first argument specifies the  
4210 minimum number of the remaining arguments that MUST evaluate to "True" for the  
4211 expression to be considered "True". If the first argument is 0, the result SHALL be "True".  
4212 If the number of arguments after the first one is less than the value of the first argument,  
4213 then the expression SHALL result in "Indeterminate". The order of evaluation SHALL be:  
4214 first evaluate the integer value, then evaluate each subsequent argument. The evaluation  
4215 SHALL stop and return "True" if the specified number of arguments evaluate to "True". The  
4216 evaluation of arguments SHALL stop if it is determined that evaluating the remaining  
4217 arguments will not satisfy the requirement.

4218 • urn:oasis:names:tc:xacml:1.0:function:not

4219 This function SHALL take one argument of data-type  
4220 "<http://www.w3.org/2001/XMLSchema#boolean>". If the argument evaluates to "True", then  
4221 the result of the expression SHALL be "False". If the argument evaluates to "False", then  
4222 the result of the expression SHALL be "True".

4223 Note: When evaluating and, or, or n-of, it MAY NOT be necessary to attempt a full evaluation of  
4224 each argument in order to determine whether the evaluation of the argument would result in  
4225 "Indeterminate". Analysis of the argument regarding the availability of its attributes, or other  
4226 analysis regarding errors, such as "divide-by-zero", may render the argument error free. Such  
4227 arguments occurring in the expression in a position after the evaluation is stated to stop need not  
4228 be processed.

### 4229 A.3.6 Numeric comparison functions

4230 These functions form a minimal set for comparing two numbers, yielding a Boolean result. They  
4231 SHALL comply with the rules governed by IEEE 754 [IEEE 754].

4232 • urn:oasis:names:tc:xacml:1.0:function:integer-greater-than

4233 • urn:oasis:names:tc:xacml:1.0:function:integer-greater-than-or-equal

4234 • urn:oasis:names:tc:xacml:1.0:function:integer-less-than

4235 • urn:oasis:names:tc:xacml:1.0:function:integer-less-than-or-equal

4236 • urn:oasis:names:tc:xacml:1.0:function:double-greater-than

4237 • urn:oasis:names:tc:xacml:1.0:function:double-greater-than-or-equal

4238 • urn:oasis:names:tc:xacml:1.0:function:double-less-than

Deleted: cd-03



4239 • urn:oasis:names:tc:xacml:1.0:function:double-less-than-or-equal

### 4240 **A.3.7 Date and time arithmetic functions**

4241 These functions perform arithmetic operations with date and time.

4242 • urn:oasis:names:tc:xacml:1.0:function:date-time-add-dayTimeDuration

4243 This function SHALL take two arguments, the first SHALL be of data-type  
4244 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be of data-type  
4245 "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration". It SHALL  
4246 return a result of "http://www.w3.org/2001/XMLSchema#dateTime". This function SHALL  
4247 return the value by adding the second argument to the first argument according to the  
4248 specification of adding durations to date and time [XS Appendix E].

4249 • urn:oasis:names:tc:xacml:1.0:function:date-time-add-yearMonthDuration

4250 This function SHALL take two arguments, the first SHALL be a  
4251 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be a  
4252 "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration". It  
4253 SHALL return a result of "http://www.w3.org/2001/XMLSchema#dateTime". This function  
4254 SHALL return the value by adding the second argument to the first argument according to  
4255 the specification of adding durations to date and time [XS Appendix E].

4256 • urn:oasis:names:tc:xacml:1.0:function:date-time-subtract-dayTimeDuration

4257 This function SHALL take two arguments, the first SHALL be a  
4258 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be a  
4259 "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration". It SHALL  
4260 return a result of "http://www.w3.org/2001/XMLSchema#dateTime". If the second argument  
4261 is a positive duration, then this function SHALL return the value by adding the  
4262 corresponding negative duration, as per the specification [XS Appendix E]. If the second  
4263 argument is a negative duration, then the result SHALL be as if the function  
4264 "urn:oasis:names:tc:xacml:1.0:function:date-time-add-dayTimeDuration" had been applied  
4265 to the corresponding positive duration.

4266 • urn:oasis:names:tc:xacml:1.0:function:date-time-subtract-yearMonthDuration

4267 This function SHALL take two arguments, the first SHALL be a  
4268 "http://www.w3.org/2001/XMLSchema#dateTime" and the second SHALL be a  
4269 "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration". It  
4270 SHALL return a result of "http://www.w3.org/2001/XMLSchema#dateTime". If the second  
4271 argument is a positive duration, then this function SHALL return the value by adding the  
4272 corresponding negative duration, as per the specification [XS Appendix E]. If the second  
4273 argument is a negative duration, then the result SHALL be as if the function  
4274 "urn:oasis:names:tc:xacml:1.0:function:date-time-add-yearMonthDuration" had been  
4275 applied to the corresponding positive duration.

4276 • urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration

4277 This function SHALL take two arguments, the first SHALL be a  
4278 "http://www.w3.org/2001/XMLSchema#date" and the second SHALL be a  
4279 "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration". It  
4280 SHALL return a result of "http://www.w3.org/2001/XMLSchema#date". This function  
4281 SHALL return the value by adding the second argument to the first argument according to  
4282 the specification of adding duration to date [XS Appendix E].

4283 • urn:oasis:names:tc:xacml:1.0:function:date-subtract-yearMonthDuration

Deleted: cd-03



4284 This function SHALL take two arguments, the first SHALL be a  
4285 "http://www.w3.org/2001/XMLSchema#date" and the second SHALL be a  
4286 "http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration". It  
4287 SHALL return a result of "http://www.w3.org/2001/XMLSchema#date". If the second  
4288 argument is a positive duration, then this function SHALL return the value by adding the  
4289 corresponding negative duration, as per the specification [XS Appendix E]. If the second  
4290 argument is a negative duration, then the result SHALL be as if the function  
4291 "urn:oasis:names:tc:xacml:1.0:function:date-add-yearMonthDuration" had been applied to  
4292 the corresponding positive duration.

### 4293 A.3.8 Non-numeric comparison functions

4294 These functions perform comparison operations on two arguments of non-numerical types.

- 4295 • urn:oasis:names:tc:xacml:1.0:function:string-greater-than

4296 This function SHALL take two arguments of data-type  
4297 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an  
4298 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the  
4299 arguments are compared byte by byte and, after an initial prefix of corresponding bytes  
4300 from both arguments that are considered equal by  
4301 "urn:oasis:names:tc:xacml:1.0:function:integer-equal", the next byte by byte comparison is  
4302 such that the byte from the first argument is greater than the byte from the second  
4303 argument by the use of the function "urn:oasis:names:tc:xacml:2.0:function:integer-greater-  
4304 then". Otherwise, it SHALL return "False".

- 4305 • urn:oasis:names:tc:xacml:1.0:function:string-greater-than-or-equal

4306 This function SHALL take two arguments of data-type  
4307 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an  
4308 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return a result as if evaluated  
4309 with the logical function "urn:oasis:names:tc:xacml:1.0:function:or" with two arguments  
4310 containing the functions "urn:oasis:names:tc:xacml:1.0:function:string-greater-than" and  
4311 "urn:oasis:names:tc:xacml:1.0:function:string-equal" containing the original arguments

- 4312 • urn:oasis:names:tc:xacml:1.0:function:string-less-than

4313 This function SHALL take two arguments of data-type  
4314 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an  
4315 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the  
4316 arguments are compared byte by byte and, after an initial prefix of corresponding bytes  
4317 from both arguments that are considered equal by  
4318 "urn:oasis:names:tc:xacml:1.0:function:integer-equal", the next byte by byte comparison is  
4319 such that the byte from the first argument is less than the byte from the second argument  
4320 by the use of the function "urn:oasis:names:tc:xacml:1.0:function:integer-less-than".  
4321 Otherwise, it SHALL return "False".

- 4322 • urn:oasis:names:tc:xacml:1.0:function:string-less-than-or-equal

4323 This function SHALL take two arguments of data-type  
4324 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an  
4325 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return a result as if evaluated  
4326 with the function "urn:oasis:names:tc:xacml:1.0:function:or" with two arguments containing  
4327 the functions "urn:oasis:names:tc:xacml:1.0:function:string-less-than" and  
4328 "urn:oasis:names:tc:xacml:1.0:function:string-equal" containing the original arguments.

- 4329 • urn:oasis:names:tc:xacml:1.0:function:time-greater-than

Deleted: cd-03

4330 This function SHALL take two arguments of data-type  
4331 "http://www.w3.org/2001/XMLSchema#time" and SHALL return an  
4332 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the  
4333 first argument is greater than the second argument according to the order relation specified  
4334 for "http://www.w3.org/2001/XMLSchema#time" [XS Section 3.2.8]. Otherwise, it SHALL  
4335 return "False". Note: it is illegal to compare a time that includes a time-zone value with one  
4336 that does not. In such cases, the time-in-range function should be used.

4337 • urn:oasis:names:tc:xacml:1.0:function:time-greater-than-or-equal

4338 This function SHALL take two arguments of data-type  
4339 "http://www.w3.org/2001/XMLSchema#time" and SHALL return an  
4340 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the  
4341 first argument is greater than or equal to the second argument according to the order  
4342 relation specified for "http://www.w3.org/2001/XMLSchema#time" [XS Section 3.2.8].  
4343 Otherwise, it SHALL return "False". Note: it is illegal to compare a time that includes a  
4344 time-zone value with one that does not. In such cases, the time-in-range function should  
4345 be used.

4346 • urn:oasis:names:tc:xacml:1.0:function:time-less-than

4347 This function SHALL take two arguments of data-type  
4348 "http://www.w3.org/2001/XMLSchema#time" and SHALL return an  
4349 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the  
4350 first argument is less than the second argument according to the order relation specified for  
4351 "http://www.w3.org/2001/XMLSchema#time" [XS Section 3.2.8]. Otherwise, it SHALL  
4352 return "False". Note: it is illegal to compare a time that includes a time-zone value with one  
4353 that does not. In such cases, the time-in-range function should be used.

4354 • urn:oasis:names:tc:xacml:1.0:function:time-less-than-or-equal

4355 This function SHALL take two arguments of data-type  
4356 "http://www.w3.org/2001/XMLSchema#time" and SHALL return an  
4357 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the  
4358 first argument is less than or equal to the second argument according to the order relation  
4359 specified for "http://www.w3.org/2001/XMLSchema#time" [XS Section 3.2.8]. Otherwise, it  
4360 SHALL return "False". Note: it is illegal to compare a time that includes a time-zone value  
4361 with one that does not. In such cases, the time-in-range function should be used.

4362 • urn:oasis:names:tc:xacml:1.0:function:time-in-range

4363 This function SHALL take three arguments of data-type  
4364 "http://www.w3.org/2001/XMLSchema#time" and SHALL return an  
4365 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if the first  
4366 argument falls in the range defined inclusively by the second and third arguments.  
4367 Otherwise, it SHALL return "False". Regardless of its value, the third argument SHALL be  
4368 interpreted as a time that is equal to, or later than by less than twenty-four hours, the  
4369 second argument. If no time zone is provided for the first argument, it SHALL use the  
4370 default time zone at the context handler. If no time zone is provided for the second or third  
4371 arguments, then they SHALL use the time zone from the first argument.

4372 • urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than

4373 This function SHALL take two arguments of data-type  
4374 "http://www.w3.org/2001/XMLSchema#dateTime" and SHALL return an  
4375 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the  
4376 first argument is greater than the second argument according to the order relation specified  
4377 for "http://www.w3.org/2001/XMLSchema#dateTime" by [XF Section 3.2.7]. Otherwise, it

Deleted: cd-03

4378 SHALL return "False". Note: if a dateTime value does not include a time-zone value, then  
4379 an implicit time-zone value SHALL be assigned, as described in [XF].

- 4380 • urn:oasis:names:tc:xacml:1.0:function:dateTime-greater-than-or-equal

4381 This function SHALL take two arguments of data-type  
4382 "http://www.w3.org/2001/XMLSchema#dateTime" and SHALL return an  
4383 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the  
4384 first argument is greater than or equal to the second argument according to the order  
4385 relation specified for "http://www.w3.org/2001/XMLSchema#dateTime" by [XF Section  
4386 3.2.7]. Otherwise, it SHALL return "False". Note: if a dateTime value does not include a  
4387 time-zone value, then an implicit time-zone value SHALL be assigned, as described in  
4388 [XF].

- 4389 • urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than

4390 This function SHALL take two arguments of data-type  
4391 "http://www.w3.org/2001/XMLSchema#dateTime" and SHALL return an  
4392 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the  
4393 first argument is less than the second argument according to the order relation specified for  
4394 "http://www.w3.org/2001/XMLSchema#dateTime" by [XF Section 3.2.7]. Otherwise, it  
4395 SHALL return "False". Note: if a dateTime value does not include a time-zone value, then  
4396 an implicit time-zone value SHALL be assigned, as described in [XF].

- 4397 • urn:oasis:names:tc:xacml:1.0:function:dateTime-less-than-or-equal

4398 This function SHALL take two arguments of data-type  
4399 "http://www.w3.org/2001/XMLSchema#dateTime" and SHALL return an  
4400 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the  
4401 first argument is less than or equal to the second argument according to the order relation  
4402 specified for "http://www.w3.org/2001/XMLSchema#dateTime" by [XF Section 3.2.7].  
4403 Otherwise, it SHALL return "False". Note: if a dateTime value does not include a time-zone  
4404 value, then an implicit time-zone value SHALL be assigned, as described in [XF].

- 4405 • urn:oasis:names:tc:xacml:1.0:function:date-greater-than

4406 This function SHALL take two arguments of data-type  
4407 "http://www.w3.org/2001/XMLSchema#date" and SHALL return an  
4408 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the  
4409 first argument is greater than the second argument according to the order relation specified  
4410 for "http://www.w3.org/2001/XMLSchema#date" by [XF Section 3.2.9]. Otherwise, it SHALL  
4411 return "False". Note: if a date value does not include a time-zone value, then an implicit  
4412 time-zone value SHALL be assigned, as described in [XF].

- 4413 • urn:oasis:names:tc:xacml:1.0:function:date-greater-than-or-equal

4414 This function SHALL take two arguments of data-type  
4415 "http://www.w3.org/2001/XMLSchema#date" and SHALL return an  
4416 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the  
4417 first argument is greater than or equal to the second argument according to the order  
4418 relation specified for "http://www.w3.org/2001/XMLSchema#date" by [XF Section 3.2.9].  
4419 Otherwise, it SHALL return "False". Note: if a date value does not include a time-zone  
4420 value, then an implicit time-zone value SHALL be assigned, as described in [XF].

- 4421 • urn:oasis:names:tc:xacml:1.0:function:date-less-than

4422 This function SHALL take two arguments of data-type  
4423 "http://www.w3.org/2001/XMLSchema#date" and SHALL return an  
4424 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the

Deleted: cd-03

4425 first argument is less than the second argument according to the order relation specified for  
4426 "http://www.w3.org/2001/XMLSchema#date" by [XF Section 3.2.9]. Otherwise, it SHALL  
4427 return "False". Note: if a date value does not include a time-zone value, then an implicit  
4428 time-zone value SHALL be assigned, as described in [XF].

- 4429 • urn:oasis:names:tc:xacml:1.0:function:date-less-than-or-equal

4430 This function SHALL take two arguments of data-type  
4431 "http://www.w3.org/2001/XMLSchema#date" and SHALL return an  
4432 "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and only if the  
4433 first argument is less than or equal to the second argument according to the order relation  
4434 specified for "http://www.w3.org/2001/XMLSchema#date" by [XF Section 3.2.9]. Otherwise,  
4435 it SHALL return "False". Note: if a date value does not include a time-zone value, then an  
4436 implicit time-zone value SHALL be assigned, as described in [XF].

### 4437 A.3.9 String functions

4438 The following functions operate on strings and URIs.

- 4439 • urn:oasis:names:tc:xacml:2.0:function:string-concatenate

4440

4441 This function SHALL take two or more arguments of data-type  
4442 "http://www.w3.org/2001/XMLSchema#string" and SHALL return a  
4443 "http://www.w3.org/2001/XMLSchema#string". The result SHALL be the concatenation, in  
4444 order, of the arguments.

- 4445 • urn:oasis:names:tc:xacml:2.0:function:url-string-concatenate

4446 This function SHALL take one argument of data-type  
4447 "http://www.w3.org/2001/XMLSchema#anyURI" and one or more arguments of type  
4448 "http://www.w3.org/2001/XMLSchema#string", and SHALL return a  
4449 "http://www.w3.org/2001/XMLSchema#anyURI". The result SHALL be the URI constructed  
4450 by appending, in order, the "string" arguments to the "anyURI" argument.

### 4451 A.3.10 Bag functions

4452 These functions operate on a **bag** of 'type' values, where *type* is one of the primitive data-types.  
4453 Some additional conditions defined for each function below SHALL cause the expression to  
4454 evaluate to "Indeterminate".

- 4455 • urn:oasis:names:tc:xacml:1.0:function:type-one-and-only

4456 This function SHALL take a **bag** of 'type' values as an argument and SHALL return a value  
4457 of 'type'. It SHALL return the only value in the **bag**. If the **bag** does not have one and only  
4458 one value, then the expression SHALL evaluate to "Indeterminate".

- 4459 • urn:oasis:names:tc:xacml:1.0:function:type-bag-size

4460 This function SHALL take a **bag** of 'type' values as an argument and SHALL return an  
4461 "http://www.w3.org/2001/XMLSchema#integer" indicating the number of values in the **bag**.

- 4462 • urn:oasis:names:tc:xacml:1.0:function:type-is-in

4463 This function SHALL take an argument of 'type' as the first argument and a **bag** of *type*  
4464 values as the second argument and SHALL return an  
4465 "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL evaluate to "True" if

Deleted: cd-03

4466 and only if the first argument matches by the "urn:oasis:names:tc:xacml:x.x:function:type-  
4467 equal" any value in the **bag**. Otherwise, it SHALL return "False".

4468 • urn:oasis:names:tc:xacml:1.0:function:type-bag

4469 This function SHALL take any number of arguments of 'type' and return a **bag** of 'type'  
4470 values containing the values of the arguments. An application of this function to zero  
4471 arguments SHALL produce an empty **bag** of the specified data-type.

### 4472 A.3.11 Set functions

4473 These functions operate on **bags** mimicking sets by eliminating duplicate elements from a **bag**.

4474 • urn:oasis:names:tc:xacml:1.0:function:type-intersection

4475 This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL  
4476 return a **bag** of 'type' values such that it contains only elements that are common between  
4477 the two **bags**, which is determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal".  
4478 No duplicates, as determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal",  
4479 SHALL exist in the result.

4480 • urn:oasis:names:tc:xacml:1.0:function:type-at-least-one-member-of

4481 This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL  
4482 return a "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL evaluate to  
4483 "True" if and only if at least one element of the first argument is contained in the second  
4484 argument as determined by "urn:oasis:names:tc:xacml:x.x:function:type-is-in".

4485 • urn:oasis:names:tc:xacml:1.0:function:type-union

4486 This function SHALL take two arguments that are both a **bag** of 'type' values. The  
4487 expression SHALL return a **bag** of 'type' such that it contains all elements of both **bags**.  
4488 No duplicates, as determined by "urn:oasis:names:tc:xacml:x.x:function:type-equal",  
4489 SHALL exist in the result.

4490 • urn:oasis:names:tc:xacml:1.0:function:type-subset

4491 This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL  
4492 return a "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return "True" if and  
4493 only if the first argument is a subset of the second argument. Each argument SHALL be  
4494 considered to have had its duplicates removed, as determined by  
4495 "urn:oasis:names:tc:xacml:x.x:function:type-equal", before the subset calculation.

4496 • urn:oasis:names:tc:xacml:1.0:function:type-set-equals

4497 This function SHALL take two arguments that are both a **bag** of 'type' values. It SHALL  
4498 return a "http://www.w3.org/2001/XMLSchema#boolean". It SHALL return the result of  
4499 applying "urn:oasis:names:tc:xacml:1.0:function:and" to the application of  
4500 "urn:oasis:names:tc:xacml:x.x:function:type-subset" to the first and second arguments and  
4501 the application of "urn:oasis:names:tc:xacml:x.x:function:type-subset" to the second and  
4502 first arguments.

### 4503 A.3.12 Higher-order bag functions

4504 This section describes functions in XACML that perform operations on **bags** such that functions  
4505 may be applied to the **bags** in general.

Deleted: cd-03

4506 In this section, a general-purpose functional language called Haskell [\[Haskell\]](#) is used to formally  
4507 specify the semantics of these functions. Although the English description is adequate, a formal  
4508 specification of the semantics is helpful.

4509 For a quick summary, in the following Haskell notation, a function definition takes the form of  
4510 clauses that are applied to patterns of structures, namely lists. The symbol "[]" denotes the empty  
4511 list, whereas the expression "(x:xs)" matches against an argument of a non-empty list of which "x"  
4512 represents the first element of the list, and "xs" is the rest of the list, which may be an empty list.  
4513 We use the Haskell notion of a list, which is an ordered collection of elements, to model the XACML  
4514 **bags** of values.

4515 A simple Haskell definition of a familiar function "urn:oasis:names:tc:xacml:1.0:function:and" that  
4516 takes a list of values of type Boolean is defined as follows:

```
4517     and:: [Bool]  -> Bool
4518     and []        = True
4519     and (x:xs)    = x && (and xs)
```

4520 The first definition line denoted by a "::-" formally describes the data-type of the function, which takes  
4521 a list of Booleans, denoted by "[Bool]", and returns a Boolean, denoted by "Bool". The second  
4522 definition line is a clause that states that the function "and" applied to the empty list is "True". The  
4523 third definition line is a clause that states that for a non-empty list, such that the first element is "x",  
4524 which is a value of data-type Bool, the function "and" applied to x SHALL be combined with, using  
4525 the logical conjunction function, which is denoted by the infix symbol "&&", the result of recursively  
4526 applying the function "and" to the rest of the list. Of course, an application of the "and" function is  
4527 "True" if and only if the list to which it is applied is empty or every element of the list is "True". For  
4528 example, the evaluation of the following Haskell expressions,

4529 (and []), (and [True]), (and [True,True]), (and [True,True,False])

4530 evaluate to "True", "True", "True", and "False", respectively.

4531 • urn:oasis:names:tc:xacml:1.0:function:any-of

4532 This function applies a Boolean function between a specific primitive value and a **bag** of  
4533 values, and SHALL return "True" if and only if the predicate is "True" for at least one  
4534 element of the **bag**.

4535 This function SHALL take three arguments. The first argument SHALL be an  
4536 <xacml:Function> element that names a Boolean function that takes two arguments of  
4537 primitive types. The second argument SHALL be a value of a primitive data-type. The third  
4538 argument SHALL be a **bag** of a primitive data-type. The expression SHALL be evaluated  
4539 as if the function named in the <xacml:Function> argument were applied to the second  
4540 argument and each element of the third argument (the **bag**) and the results are combined  
4541 with "urn:oasis:names:tc:xacml:1.0:function:or".

4542 In Haskell, the semantics of this operation are as follows:

```
4543     any_of :: ( a -> b -> Bool )-> a -> [b] -> Bool
4544     any_of f a []      = False
4545     any_of f a (x:xs)  = (f a x) || (any_of f a xs)
```

4546 In the above notation, "f" is the function to be applied, "a" is the primitive value, and "(x:xs)"  
4547 represents the first element of the list as "x" and the rest of the list as "xs".

4548 For example, the following expression SHALL return "True":

Deleted: cd-03



```

4549 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of">
4550   <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>
4551   <AttributeValue
4552     DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>
4553   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
4554     <AttributeValue
4555       DataType="http://www.w3.org/2001/XMLSchema#string">John</AttributeValue>
4556     <AttributeValue
4557       DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>
4558     <AttributeValue
4559       DataType="http://www.w3.org/2001/XMLSchema#string">George</AttributeValue>
4560     <AttributeValue
4561       DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>
4562   </Apply>
4563 </Apply>

```

4564 This expression is "True" because the first argument is equal to at least one of the  
 4565 elements of the **bag**, according to the function.

4566 • urn:oasis:names:tc:xacml:1.0:function:all-of

4567 This function applies a Boolean function between a specific primitive value and a **bag** of  
 4568 values, and returns "True" if and only if the predicate is "True" for every element of the **bag**.

4569 This function SHALL take three arguments. The first argument SHALL be an  
 4570 <xacml:Function> element that names a Boolean function that takes two arguments of  
 4571 primitive types. The second argument SHALL be a value of a primitive data-type. The third  
 4572 argument SHALL be a **bag** of a primitive data-type. The expression SHALL be evaluated  
 4573 as if the function named in the <xacml:Function> argument were applied to the second  
 4574 argument and each element of the third argument (the **bag**) and the results were combined  
 4575 using "urn:oasis:names:tc:xacml:1.0:function:and".

4576 In Haskell, the semantics of this operation are as follows:

```

4577 all_of :: ( a -> b -> Bool ) -> a -> [b] -> Bool
4578 all_of f a []           = False
4579 all_of f a (x:xs)       = (f a x) && (all_of f a xs)

```

4580 In the above notation, "f" is the function to be applied, "a" is the primitive value, and "(x:xs)"  
 4581 represents the first element of the list as "x" and the rest of the list as "xs".

4582 For example, the following expression SHALL evaluate to "True":

```

4583 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of">
4584   <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-greater"/>
4585   <AttributeValue
4586     DataType="http://www.w3.org/2001/XMLSchema#integer">10</AttributeValue>
4587   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4588     <AttributeValue
4589       DataType="http://www.w3.org/2001/XMLSchema#integer">9</AttributeValue>
4590     <AttributeValue
4591       DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4592     <AttributeValue
4593       DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>
4594     <AttributeValue
4595       DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>
4596   </Apply>
4597 </Apply>

```

4598 This expression is "True" because the first argument (10) is greater than *all* of the elements  
 4599 of the **bag** (9,3,4 and 2).

4600 • urn:oasis:names:tc:xacml:1.0:function:any-of-any

Deleted: cd-03

4601 This function applies a Boolean function between each element of a **bag** of values and  
 4602 each element of another **bag** of values, and returns "True" if and only if the predicate is  
 4603 "True" for at least one comparison.

4604 This function SHALL take three arguments. The first argument SHALL be an  
 4605 `<xacml:Function>` element that names a Boolean function that takes two arguments of  
 4606 primitive types. The second argument SHALL be a **bag** of a primitive data-type. The third  
 4607 argument SHALL be a **bag** of a primitive data-type. The expression SHALL be evaluated  
 4608 as if the function named in the `<xacml:Function>` argument were applied between  
 4609 every element of the second argument and every element of the third argument and the  
 4610 results were combined using "urn:oasis:names:tc:xacml:1.0:function:or". The semantics  
 4611 are that the result of the expression SHALL be "True" if and only if the applied predicate is  
 4612 "True" for *at least one* comparison of elements from the two **bags**.

4613 In Haskell, taking advantage of the "any\_of" function defined above, the semantics of the  
 4614 "any\_of\_any" function are as follows:

```
4615 any_of_any :: ( a -> b -> Bool ) -> [a]-> [b] -> Bool
4616 any_of_any f [] ys = False
4617 any_of_any f (x:xs) ys = (any_of f x ys) || (any_of_any f xs ys)
```

4618 In the above notation, "f" is the function to be applied and "(x:xs)" represents the first  
 4619 element of the list as "x" and the rest of the list as "xs".

4620 For example, the following expression SHALL evaluate to "True":

```
4621 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of-any">
4622   <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal" />
4623   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
4624     <AttributeValue
4625       DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>
4626     <AttributeValue
4627       DataType="http://www.w3.org/2001/XMLSchema#string">Mary</AttributeValue>
4628   </Apply>
4629   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
4630     <AttributeValue
4631       DataType="http://www.w3.org/2001/XMLSchema#string">John</AttributeValue>
4632     <AttributeValue
4633       DataType="http://www.w3.org/2001/XMLSchema#string">Paul</AttributeValue>
4634     <AttributeValue
4635       DataType="http://www.w3.org/2001/XMLSchema#string">George</AttributeValue>
4636     <AttributeValue
4637       DataType="http://www.w3.org/2001/XMLSchema#string">Ringo</AttributeValue>
4638   </Apply>
4639 </Apply>
```

4640 This expression is "True" because at least one of the elements of the first **bag**, namely  
 4641 "Ringo", is equal to at least one of the elements of the second **bag**.

4642 • urn:oasis:names:tc:xacml:1.0:function:all-of-any

4643 This function applies a Boolean function between the elements of two **bags**. The  
 4644 expression SHALL be "True" if and only if the supplied predicate is 'True' between each  
 4645 element of the first **bag** and any element of the second **bag**.

4646 This function SHALL take three arguments. The first argument SHALL be an  
 4647 `<xacml:Function>` element that names a Boolean function that takes two arguments of  
 4648 primitive types. The second argument SHALL be a **bag** of a primitive data-type. The third  
 4649 argument SHALL be a **bag** of a primitive data-type. The expression SHALL be evaluated  
 4650 as if the "urn:oasis:names:tc:xacml:1.0:function:any-of" function had been applied to each

Deleted: cd-03



4651 value of the first **bag** and the whole of the second **bag** using the supplied xacml:Function,  
4652 and the results were then combined using "urn:oasis:names:tc:xacml:1.0:function:and".

4653 In Haskell, taking advantage of the "any\_of" function defined in Haskell above, the  
4654 semantics of the "all\_of\_any" function are as follows:

```
4655 all_of_any :: ( a -> b -> Bool ) -> [a]-> [b] -> Bool
4656 all_of_any f [] ys = True
4657 all_of_any f (x:xs) ys = (any_of f x ys) && (all_of_any f xs ys)
```

4658 In the above notation, "f" is the function to be applied and "(x:xs)" represents the first  
4659 element of the list as "x" and the rest of the list as "xs".

4660 For example, the following expression SHALL evaluate to "True":

```
4661 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of-any">
4662   <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-greater"/>
4663   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4664     <AttributeValue
4665       DataType="http://www.w3.org/2001/XMLSchema#integer">10</AttributeValue>
4666     <AttributeValue
4667       DataType="http://www.w3.org/2001/XMLSchema#integer">20</AttributeValue>
4668   </Apply>
4669   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4670     <AttributeValue
4671       DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
4672     <AttributeValue
4673       DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4674     <AttributeValue
4675       DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>
4676     <AttributeValue
4677       DataType="http://www.w3.org/2001/XMLSchema#integer">19</AttributeValue>
4678   </Apply>
4679 </Apply>
```

4680 This expression is "True" because each of the elements of the first **bag** is greater than at  
4681 least one of the elements of the second **bag**.

4682 • urn:oasis:names:tc:xacml:1.0:function:any-of-all

4683 This function applies a Boolean function between the elements of two **bags**. The  
4684 expression SHALL be "True" if and only if the supplied predicate is "True" between each  
4685 element of the second **bag** and any element of the first **bag**.

4686 This function SHALL take three arguments. The first argument SHALL be an  
4687 <xacml:Function> element that names a Boolean function that takes two arguments of  
4688 primitive types. The second argument SHALL be a **bag** of a primitive data-type. The third  
4689 argument SHALL be a **bag** of a primitive data-type. The expression SHALL be evaluated  
4690 as if the "urn:oasis:names:tc:xacml:1.0:function:any-of" function had been applied to each  
4691 value of the second **bag** and the whole of the first **bag** using the supplied xacml:Function,  
4692 and the results were then combined using "urn:oasis:names:tc:xacml:1.0:function:and".

4693 In Haskell, taking advantage of the "all\_of" function defined in Haskell above, the semantics  
4694 of the "any\_of\_all" function are as follows:

```
4695 any_of_all :: ( a -> b -> Bool ) -> [a]-> [b] -> Bool
4696 any_of_all f [] ys = False
4697 any_of_all f (x:xs) ys = (all_of f x ys) || (any_of_all f xs ys)
```

4698 In the above notation, "f" is the function name to be applied and "(x:xs)" represents the first  
4699 element of the list as "x" and the rest of the list as "xs".

4700 For example, the following expression SHALL evaluate to "True":

Deleted: cd-03

access\_control-xacml-2.0-core-spec: [cd-04](#)

121

```

4701 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:any-of-all">
4702   <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-greater"/>
4703   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4704     <AttributeValue
4705       DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4706     <AttributeValue
4707       DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>
4708   </Apply>
4709   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4710     <AttributeValue
4711       DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
4712     <AttributeValue
4713       DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>
4714     <AttributeValue
4715       DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4716     <AttributeValue
4717       DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>
4718   </Apply>
4719 </Apply>

```

4720 This expression is "True" because, for all of the values in the second **bag**, there is a value  
 4721 in the first **bag** that is greater.

4722 • urn:oasis:names:tc:xacml:1.0:function:all-of-all

4723 This function applies a Boolean function between the elements of two **bags**. The  
 4724 expression SHALL be "True" if and only if the supplied predicate is "True" between each  
 4725 and every element of the first **bag** collectively against all the elements of the second **bag**.

4726 This function SHALL take three arguments. The first argument SHALL be an  
 4727 <xacml:Function> element that names a Boolean function that takes two arguments of  
 4728 primitive types. The second argument SHALL be a **bag** of a primitive data-type. The third  
 4729 argument SHALL be a **bag** of a primitive data-type. The expression is evaluated as if the  
 4730 function named in the <xacml:Function> element were applied between every element  
 4731 of the second argument and every element of the third argument and the results were  
 4732 combined using "urn:oasis:names:tc:xacml:1.0:function:and". The semantics are that the  
 4733 result of the expression is "True" if and only if the applied predicate is "True" for all  
 4734 elements of the first **bag** compared to all the elements of the second **bag**.

4735 In Haskell, taking advantage of the "all\_of" function defined in Haskell above, the semantics  
 4736 of the "all\_of\_all" function is as follows:

```

4737 all_of_all :: ( a -> b -> Bool ) -> [a] -> [b] -> Bool
4738 all_of_all f [] ys = False
4739 all_of_all f (x:xs) ys = (all_of f x ys) && (all_of_all f xs ys)

```

4740 In the above notation, "f" is the function to be applied and "(x:xs)" represents the first  
 4741 element of the list as "x" and the rest of the list as "xs".

4742 For example, the following expression SHALL evaluate to "True":

Deleted: cd-03

```

4743 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:all-of-all">
4744   <Function FunctionId="urn:oasis:names:tc:xacml:2.0:function:integer-greater"/>
4745   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4746     <AttributeValue
4747       DataType="http://www.w3.org/2001/XMLSchema#integer">6</AttributeValue>
4748     <AttributeValue
4749       DataType="http://www.w3.org/2001/XMLSchema#integer">5</AttributeValue>
4750   </Apply>
4751   <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-bag">
4752     <AttributeValue
4753       DataType="http://www.w3.org/2001/XMLSchema#integer">1</AttributeValue>
4754     <AttributeValue
4755       DataType="http://www.w3.org/2001/XMLSchema#integer">2</AttributeValue>
4756     <AttributeValue
4757       DataType="http://www.w3.org/2001/XMLSchema#integer">3</AttributeValue>
4758     <AttributeValue
4759       DataType="http://www.w3.org/2001/XMLSchema#integer">4</AttributeValue>
4760   </Apply>
4761 </Apply>

```

This expression is "True" because all elements of the first **bag**, "5" and "6", are each greater than all of the integer values "1", "2", "3", "4" of the second **bag**.

- urn:oasis:names:tc:xacml:1.0:function:map

This function converts a **bag** of values to another **bag** of values.

This function SHALL take two arguments. The first function SHALL be an `<xacml:Function>` element naming a function that takes a single argument of a primitive data-type and returns a value of a primitive data-type. The second argument SHALL be a **bag** of a primitive data-type. The expression SHALL be evaluated as if the function named in the `<xacml:Function>` element were applied to each element in the **bag** resulting in a **bag** of the converted value. The result SHALL be a **bag** of the primitive data-type that is returned by the function named in the `<xacml:Function>` element.

In Haskell, this function is defined as follows:

```

4774 map:: (a -> b) -> [a] -> [b]
4775 map f []      = []
4776 map f (x:xs)  = (f x) : (map f xs)

```

In the above notation, "f" is the function to be applied and "(x:xs)" represents the first element of the list as "x" and the rest of the list as "xs".

For example, the following expression,

```

4780 <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:map">
4781   <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-normalize-
4782     to-lower-case">
4783     <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
4784       <AttributeValue
4785         DataType="http://www.w3.org/2001/XMLSchema#string">Hello</AttributeValue>
4786       <AttributeValue
4787         DataType="http://www.w3.org/2001/XMLSchema#string">World!</AttributeValue>
4788     </Apply>
4789   </Apply>

```

evaluates to a **bag** containing "hello" and "world!".

Deleted: cd-03

### 4791 A.3.13 Regular-expression-based functions

4792 These functions operate on various types using regular expressions and evaluate to  
4793 "http://www.w3.org/2001/XMLSchema#boolean".

- 4794 • urn:oasis:names:tc:xacml:1.0:function:~~string-regexp-match~~

Deleted: regexp-string

4795 This function decides a regular expression match. It SHALL take two arguments of  
4796 "http://www.w3.org/2001/XMLSchema#string" and SHALL return an  
4797 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular  
4798 expression and the second argument SHALL be a general string. The function  
4799 specification SHALL be that of the "xf:matches" function with the arguments reversed [XF  
4800 Section 6.3.15].

- 4801 • urn:oasis:names:tc:xacml:2.0:function:~~anyURI-regexp-match~~

Deleted: regexp-anyURI

4802 This function decides a regular expression match. It SHALL take two arguments; the first is  
4803 of type "http://www.w3.org/2001/XMLSchema#string" and the second is of type  
4804 "http://www.w3.org/2001/XMLSchema#anyURI". It SHALL return an  
4805 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular  
4806 expression and the second argument SHALL be a URI. The function SHALL convert the  
4807 second argument to type "http://www.w3.org/2001/XMLSchema#string", then apply  
4808 "urn:oasis:names:tc:xacml:1.0:function:~~string-regexp-match~~".

Deleted: regexp-string

- 4809 • urn:oasis:names:tc:xacml:2.0:function:~~ipAddress-regexp-match~~

Deleted: regexp-ipAddress

4810 This function decides a regular expression match. It SHALL take two arguments; the first is  
4811 of type "http://www.w3.org/2001/XMLSchema#string" and the second is of type  
4812 "urn:oasis:names:tc:xacml:2.0:data-type:ipAddress". It SHALL return an  
4813 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular  
4814 expression and the second argument SHALL be an IPv4 or IPv6 address. The function  
4815 SHALL convert the second argument to type "http://www.w3.org/2001/XMLSchema#string",  
4816 then apply "urn:oasis:names:tc:xacml:1.0:function:~~string-regexp-match~~".

Deleted: regexp-string

- 4817 • urn:oasis:names:tc:xacml:2.0:function:~~dnsName-regexp-match~~

Deleted: regexp-dnsName

4818 This function decides a regular expression match. It SHALL take two arguments; the first is  
4819 of type "http://www.w3.org/2001/XMLSchema#string" and the second is of type  
4820 "urn:oasis:names:tc:xacml:2.0:data-type:dnsName". It SHALL return an  
4821 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular  
4822 expression and the second argument SHALL be a DNS name. The function SHALL  
4823 convert the second argument to type "http://www.w3.org/2001/XMLSchema#string", then  
4824 apply "urn:oasis:names:tc:xacml:1.0:function:~~string-regexp-match~~".

Deleted: regexp-string

- 4825 • urn:oasis:names:tc:xacml:2.0:function:~~rfc822Name-regexp-match~~

Deleted: regexp-rfc822Name

4826 This function decides a regular expression match. It SHALL take two arguments; the first is  
4827 of type "http://www.w3.org/2001/XMLSchema#string" and the second is of type  
4828 "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name". It SHALL return an  
4829 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular  
4830 expression and the second argument SHALL be an RFC 822 name. The function SHALL  
4831 convert the second argument to type "http://www.w3.org/2001/XMLSchema#string", then  
4832 apply "urn:oasis:names:tc:xacml:1.0:function:~~string-regexp-match~~".

Deleted: regexp-string

- 4833 • urn:oasis:names:tc:xacml:2.0:function:~~x500Name-regexp-match~~

Deleted: regexp-

4834 This function decides a regular expression match. It SHALL take two arguments; the first is  
4835 of type "http://www.w3.org/2001/XMLSchema#string" and the second is of type  
4836 "urn:oasis:names:tc:xacml:1.0:data-type:x500Name". It SHALL return an

Deleted: cd-03

access\_control-xacml-2.0-core-spec-~~cd-04~~

124

4837 "http://www.w3.org/2001/XMLSchema#boolean". The first argument SHALL be a regular  
4838 expression and the second argument SHALL be an X.500 directory name. The function  
4839 SHALL convert the second argument to type "http://www.w3.org/2001/XMLSchema#string",  
4840 then apply "urn:oasis:names:tc:xacml:1.0:function:~~string-regexp-match~~".

Deleted: regexp-string

### 4841 A.3.14 Special match functions

4842 These functions operate on various types and evaluate to  
4843 "http://www.w3.org/2001/XMLSchema#boolean" based on the specified standard matching  
4844 algorithm.

- 4845 • urn:oasis:names:tc:xacml:1.0:function:x500Name-match

4846 This function shall take two arguments of "urn:oasis:names:tc:xacml:2.0:data-  
4847 type:x500Name" and shall return an "http://www.w3.org/2001/XMLSchema#boolean". It  
4848 shall return "True" if and only if the first argument matches some terminal sequence of  
4849 RDNs from the second argument when compared using x500Name-equal.

- 4850 • urn:oasis:names:tc:xacml:1.0:function:rfc822Name-match

4851 This function SHALL take two arguments, the first is of data-type  
4852 "http://www.w3.org/2001/XMLSchema#string" and the second is of data-type  
4853 "urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name" and SHALL return an  
4854 "http://www.w3.org/2001/XMLSchema#boolean". This function SHALL evaluate to "True" if  
4855 the first argument matches the second argument according to the following specification.

4856 An RFC822 name consists of a local-part followed by "@" followed by a domain-part. The  
4857 local-part is case-sensitive, while the domain-part (which is usually a DNS name) is not  
4858 case-sensitive.<sup>4</sup>

4859 The second argument contains a complete rfc822Name. The first argument is a complete  
4860 or partial rfc822Name used to select appropriate values in the second argument as follows.

4861 In order to match a particular address in the second argument, the first argument must  
4862 specify the complete mail address to be matched. For example, if the first argument is  
4863 "Anderson@sun.com", this matches a value in the second argument of  
4864 "Anderson@sun.com" and "Anderson@SUN.COM", but not "Anne.Anderson@sun.com",  
4865 "anderson@sun.com" or "Anderson@east.sun.com".

4866 In order to match any address at a particular domain in the second argument, the first  
4867 argument must specify only a domain name (usually a DNS name). For example, if the first  
4868 argument is "sun.com", this matches a value in the first argument of "Anderson@sun.com"  
4869 or "Baxter@SUN.COM", but not "Anderson@east.sun.com".

4870 In order to match any address in a particular domain in the second argument, the first  
4871 argument must specify the desired domain-part with a leading ".". For example, if the first  
4872 argument is ".east.sun.com", this matches a value in the second argument of  
4873 "Anderson@east.sun.com" and "anne.anderson@ISRG.EAST.SUN.COM" but not  
4874 "Anderson@sun.com".

---

4 According to IETF RFC822 and its successor specifications [RFC2821], case is significant in the *local-part*. Many mail systems, as well as the IETF PKIX specification, treat the *local-part* as case-insensitive. This anomaly is considered an error by mail-system designers and is not encouraged. For this reason, rfc822Name-match treats *local-part* as case sensitive.

Deleted: cd-03

### A.3.15 XPath-based functions

This section specifies functions that take XPath expressions for arguments. An XPath expression evaluates to a *node-set*, which is a set of XML nodes that match the expression. A node or node-set is not in the formal data-type system of XACML. All comparison or other operations on node-sets are performed in isolation of the particular function specified. That is, the XPath expressions in these functions are restricted to the XACML request **context**. The `<xacml-context:Request>` element is the context node for every XPath expression. The following functions are defined:

- urn:oasis:names:tc:xacml:1.0:function:xpath-node-count

This function SHALL take an "http://www.w3.org/2001/XMLSchema#string" as an argument, which SHALL be interpreted as an XPath expression, and evaluates to an "http://www.w3.org/2001/XMLSchema#integer". The value returned from the function SHALL be the count of the nodes within the node-set that match the given XPath expression.

- urn:oasis:names:tc:xacml:1.0:function:xpath-node-equal

This function SHALL take two "http://www.w3.org/2001/XMLSchema#string" arguments, which SHALL be interpreted as XPath expressions, and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". The function SHALL return "True" if any of the XML nodes in the node-set matched by the first argument equals, according to the "op:node-equal" function [XF Section 13.1.6], any of the XML nodes in the node-set matched by the second argument.

- urn:oasis:names:tc:xacml:1.0:function:xpath-node-match

This function SHALL take two "http://www.w3.org/2001/XMLSchema#string" arguments, which SHALL be interpreted as XPath expressions and SHALL return an "http://www.w3.org/2001/XMLSchema#boolean". This function SHALL evaluate to "True" if one of the following two conditions is satisfied: (1) Any of the XML nodes in the node-set matched by the first argument is equal, according to "op:node-equal" [XF Section 13.1.6], to any of the XML nodes in the node-set matched by the second argument; (2) any attribute and element node below any of the XML nodes in the node-set matched by the first argument is equal, according to "op:node-equal" [XF Section 13.1.6], to any of the XML nodes in the node-set matched by the second argument.

NOTE: The first condition is equivalent to "xpath-node-equal", and guarantees that "xpath-node-equal" is a special case of "xpath-node-match".

### A.3.16 Extension functions and primitive types

Functions and primitive types are specified by string identifiers allowing for the introduction of functions in addition to those specified by XACML. This approach allows one to extend the XACML module with special functions and special primitive data-types.

In order to preserve the integrity of the XACML evaluation strategy, the result of an extension function SHALL depend only on the values of its arguments. Global and hidden parameters SHALL NOT affect the evaluation of an expression. Functions SHALL NOT have side effects, as evaluation order cannot be guaranteed in a standard way.

Deleted: cd-03

## Appendix B. XACML identifiers (normative)

This section defines standard identifiers for commonly used entities.

### B.1. XACML namespaces

There are currently two defined XACML namespaces.

Policies are defined using this identifier.

urn:oasis:names:tc:xacml:2.0:policy:schema:cd:04

Deleted: cd:03

Request and response **contexts** are defined using this identifier.

urn:oasis:names:tc:xacml:2.0:context:schema:cd:04

Deleted: cd:03

### B.2. Access subject categories

This identifier indicates the system entity that initiated the **access** request. That is, the initial entity in a request chain. If **subject** category is not specified, this is the default value.

urn:oasis:names:tc:xacml:1.0:subject-category:access-subject

This identifier indicates the system entity that will receive the results of the request (used when it is distinct from the access-subject).

urn:oasis:names:tc:xacml:1.0:subject-category:recipient-subject

This identifier indicates a system entity through which the **access** request was passed. There may be more than one. No means is provided to specify the order in which they passed the message.

urn:oasis:names:tc:xacml:1.0:subject-category:intermediary-subject

This identifier indicates a system entity associated with a local or remote codebase that generated the request. Corresponding **subject attributes** might include the URL from which it was loaded and/or the identity of the code-signer. There may be more than one. No means is provided to specify the order in which they processed the request.

urn:oasis:names:tc:xacml:1.0:subject-category:codebase

This identifier indicates a system entity associated with the computer that initiated the **access** request. An example would be an IPsec identity.

urn:oasis:names:tc:xacml:1.0:subject-category:requesting-machine

### B.3. Data-types

The following identifiers indicate data-types that are defined in Section A.2.

urn:oasis:names:tc:xacml:1.0:data-type:x500Name.

urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name

urn:oasis:names:tc:xacml:2.0:data-type:ipAddress

urn:oasis:names:tc:xacml:2.0:data-type:dnsName

The following data-type identifiers are defined by XML Schema [XS].

http://www.w3.org/2001/XMLSchema#string

http://www.w3.org/2001/XMLSchema#boolean

http://www.w3.org/2001/XMLSchema#integer

Deleted: cd-03

access\_control-xacml-2.0-core-spec-cd-04



4951 <http://www.w3.org/2001/XMLSchema#double>  
4952 <http://www.w3.org/2001/XMLSchema#time>  
4953 <http://www.w3.org/2001/XMLSchema#date>  
4954 <http://www.w3.org/2001/XMLSchema#dateTime>  
4955 <http://www.w3.org/2001/XMLSchema#anyURI>  
4956 <http://www.w3.org/2001/XMLSchema#hexBinary>  
4957 <http://www.w3.org/2001/XMLSchema#base64Binary>  
4958 The following data-type identifiers correspond to the `dayTimeDuration` and `yearMonthDuration`  
4959 data-types defined in [XF Sections 8.2.2 and 8.2.1, respectively].  
4960 <http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration>  
4961 <http://www.w3.org/TR/2002/WD-xquery-operators-20020816#yearMonthDuration>

## 4962 B.4. Subject attributes

4963 These identifiers indicate **attributes** of a **subject**. When used, they SHALL appear within a  
4964 `<Subject>` element of the request **context**. They SHALL be accessed by means of a  
4965 `<SubjectAttributeDesignator>` element, or an `<AttributeSelector>` element that points  
4966 into a `<Subject>` element of the request **context**.

4967 At most one of each of these attributes is associated with each subject. Each attribute associated  
4968 with authentication included within a single `<Subject>` element relates to the same authentication  
4969 event.

4970 This identifier indicates the name of the **subject**. The default format is  
4971 "<http://www.w3.org/2001/XMLSchema#string>". To indicate other formats, use the `DataType`  
4972 attributes listed in B.3

4973 `urn:oasis:names:tc:xacml:1.0:subject:subject-id`

4974 This identifier indicates the **subject** category. "access-subject" is the default value.

4975 `urn:oasis:names:tc:xacml:1.0:subject:category`

4976 This identifier indicates the security domain of the **subject**. It identifies the administrator and policy  
4977 that manages the name-space in which the **subject** id is administered.

4978 `urn:oasis:names:tc:xacml:1.0:subject:subject-id-qualifier`

4979 This identifier indicates a public key used to confirm the **subject's** identity.

4980 `urn:oasis:names:tc:xacml:1.0:subject:key-info`

4981 This identifier indicates the time at which the **subject** was authenticated.

4982 `urn:oasis:names:tc:xacml:1.0:subject:authentication-time`

4983 This identifier indicates the method used to authenticate the **subject**.

4984 `urn:oasis:names:tc:xacml:1.0:subject:authn-locality:authentication-method`

4985 This identifier indicates the time at which the **subject** initiated the **access** request, according to the  
4986 **PEP**.

4987 `urn:oasis:names:tc:xacml:1.0:subject:request-time`

4988 This identifier indicates the time at which the **subject's** current session began, according to the  
4989 **PEP**.

4990 `urn:oasis:names:tc:xacml:1.0:subject:session-start-time`

4991 The following identifiers indicate the location where authentication credentials were activated. They  
4992 are intended to support the corresponding entities from the SAML authentication statement  
4993 [SAML].

4994 This identifier indicates that the location is expressed as an IP address.

| `access_control-xacml-2.0-core-spec-cd-04`

Deleted: cd-03

4995 urn:oasis:names:tc:xacml:1.0:subject:authn-locality:ip-address  
4996 The corresponding attribute SHALL be of data-type "http://www.w3.org/2001/XMLSchema#string".  
4997 This identifier indicates that the location is expressed as a DNS name.  
4998 urn:oasis:names:tc:xacml:1.0:subject:authn-locality:dns-name  
4999 The corresponding attribute SHALL be of data-type "http://www.w3.org/2001/XMLSchema#string".  
5000 Where a suitable attribute is already defined in LDAP [LDAP-1, LDAP-2], the XACML identifier  
5001 SHALL be formed by adding the **attribute** name to the URI of the LDAP specification. For  
5002 example, the **attribute** name for the userPassword defined in the RFC 2256 SHALL be:  
5003 http://www.ietf.org/rfc/rfc2256.txt#userPassword

## 5004 B.6. Resource attributes

5005 These identifiers indicate **attributes** of the **resource**. The corresponding **attributes** MAY appear in  
5006 the <Resource> element of the request **context** and be accessed by means of a  
5007 <ResourceAttributeDesignator> element, or by an <AttributeSelector> element that  
5008 points into the <Resource> element of the request **context**.  
5009 This **attribute** identifies the **resource** to which access is requested. If an <xacml-  
5010 context:ResourceContent> element is provided, then the resource to which access is  
5011 requested SHALL be all or a portion of the resource supplied in the <xacml-  
5012 context:ResourceContent> element.  
5013 urn:oasis:names:tc:xacml:1.0:resource:resource-id  
5014 This **attribute** identifies the namespace of the top element of the contents of the <xacml-  
5015 context:ResourceContent> element. In the case where the **resource** content is supplied in the  
5016 request **context** and the **resource** namespace is defined in the **resource**, the PDP SHALL confirm  
5017 that the namespace defined by this **attribute** is the same as that defined in the **resource**. The type  
5018 of the corresponding **attribute** SHALL be "http://www.w3.org/2001/XMLSchema#anyURI".  
5019 urn:oasis:names:tc:xacml:2.0:resource:target-namespace

## 5020 B.7. Action attributes

5021 These identifiers indicate **attributes** of the **action** being requested. When used, they SHALL  
5022 appear within the <Action> element of the request **context**. They SHALL be accessed by means  
5023 of an <ActionAttributeDesignator> element, or an <AttributeSelector> element that  
5024 points into the <Action> element of the request **context**.  
5025 This **attribute** identifies the **action** for which **access** is requested.  
5026 urn:oasis:names:tc:xacml:1.0:action:action-id  
5027 Where the **action** is implicit, the value of the action-id **attribute** SHALL be  
5028 urn:oasis:names:tc:xacml:1.0:action:implied-action  
5029 This **attribute** identifies the namespace in which the action-id **attribute** is defined.  
5030 urn:oasis:names:tc:xacml:1.0:action:action-namespace

Deleted: cd-03

## B.8. Environment attributes

These identifiers indicate **attributes** of the **environment** within which the **decision request** is to be evaluated. When used in the **decision request**, they SHALL appear in the <Environment> element of the request **context**. They SHALL be accessed by means of an <EnvironmentAttributeDesignator> element, or an <AttributeSelector> element that points into the <Environment> element of the request **context**.

This identifier indicates the current time at the **context handler**. In practice it is the time at which the request **context** was created. For this reason, if these identifiers appear in multiple places within a <Policy> or <PolicySet>, then the same value SHALL be assigned to each occurrence in the evaluation procedure, regardless of how much time elapses between the processing of the occurrences.

urn:oasis:names:tc:xacml:1.0:environment:current-time

The corresponding **attribute** SHALL be of data-type  
"http://www.w3.org/2001/XMLSchema#time".

urn:oasis:names:tc:xacml:1.0:environment:current-date

The corresponding **attribute** SHALL be of data-type  
"http://www.w3.org/2001/XMLSchema#date".

urn:oasis:names:tc:xacml:1.0:environment:current-dateTime

The corresponding **attribute** SHALL be of data-type  
"http://www.w3.org/2001/XMLSchema#dateTime".

## B.9. Status codes

The following status code values are defined.

This identifier indicates success.

urn:oasis:names:tc:xacml:1.0:status:ok

This identifier indicates that all the attributes necessary to make a policy decision were not available (see Section 6.16).

urn:oasis:names:tc:xacml:1.0:status:missing-attribute

This identifier indicates that some attribute value contained a syntax error, such as a letter in a numeric field.

urn:oasis:names:tc:xacml:1.0:status:syntax-error

This identifier indicates that an error occurred during policy evaluation. An example would be division by zero.

urn:oasis:names:tc:xacml:1.0:status:processing-error

## B.10. Combining algorithms

The deny-overrides rule-combining algorithm has the following value for the ruleCombiningAlgId attribute:

urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides

access\_control-xacml-2.0-core-spec-[cd-04](#)

Deleted: cd-03

5068 The deny-overrides policy-combining algorithm has the following value for the  
 5069 policyCombiningAlgId attribute:  
 5070 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:deny-overrides  
 5071 The permit-overrides rule-combining algorithm has the following value for the  
 5072 ruleCombiningAlgId attribute:  
 5073 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides  
 5074 The permit-overrides policy-combining algorithm has the following value for the  
 5075 policyCombiningAlgId attribute:  
 5076 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides  
 5077 The first-applicable rule-combining algorithm has the following value for the  
 5078 ruleCombiningAlgId attribute:  
 5079 urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable  
 5080 The first-applicable policy-combining algorithm has the following value for the  
 5081 policyCombiningAlgId attribute:  
 5082 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-applicable  
 5083 The only-one-applicable-policy policy-combining algorithm has the following value for the  
 5084 policyCombiningAlgId attribute:  
 5085 urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:only-one-  
 5086 applicable  
 5087 The ordered-deny-overrides rule-combining algorithm has the following value for the  
 5088 ruleCombiningAlgId attribute:  
 5089 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-deny-  
 5090 overrides  
 5091 The ordered-deny-overrides policy-combining algorithm has the following value for the  
 5092 policyCombiningAlgId attribute:  
 5093 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-deny-  
 5094 overrides  
 5095 The ordered-permit-overrides rule-combining algorithm has the following value for the  
 5096 ruleCombiningAlgId attribute:  
 5097 urn:oasis:names:tc:xacml:1.1:rule-combining-algorithm:ordered-permit-  
 5098 overrides  
 5099 The ordered-permit-overrides policy-combining algorithm has the following value for the  
 5100 policyCombiningAlgId attribute:  
 5101 urn:oasis:names:tc:xacml:1.1:policy-combining-algorithm:ordered-permit-  
 5102 overrides

Deleted: cd-03

## Appendix C. Combining algorithms (normative)

This section contains a description of the **rule**- and **policy-combining algorithms** specified by XACML.

### C.1. Deny-overrides

The following specification defines the "Deny-overrides" **rule-combining algorithm** of a **policy**.

In the entire set of **rules** in the **policy**, if any **rule** evaluates to "Deny", then the result of the **rule** combination SHALL be "Deny". If any **rule** evaluates to "Permit" and all other **rules** evaluate to "NotApplicable", then the result of the **rule** combination SHALL be "Permit". In other words, "Deny" takes precedence, regardless of the result of evaluating any of the other **rules** in the combination. If all **rules** are found to be "NotApplicable" to the **decision request**, then the **rule** combination SHALL evaluate to "NotApplicable".

If an error occurs while evaluating the **target** or **condition** of a **rule** that contains an **effect** value of "Deny" then the evaluation SHALL continue to evaluate subsequent **rules**, looking for a result of "Deny". If no other **rule** evaluates to "Deny", then the combination SHALL evaluate to "Indeterminate", with the appropriate error status.

If at least one **rule** evaluates to "Permit", all other **rules** that do not have evaluation errors evaluate to "Permit" or "NotApplicable" and all **rules** that do have evaluation errors contain **effects** of "Permit", then the result of the combination SHALL be "Permit".

The following pseudo-code represents the evaluation strategy of this **rule-combining algorithm**.

```
Decision denyOverridesRuleCombiningAlgorithm(Rule rule[])
{
    Boolean atLeastOneError = false;
    Boolean potentialDeny = false;
    Boolean atLeastOnePermit = false;
    for( i=0 ; i < lengthOf(rules) ; i++ )
    {
        Decision decision = evaluate(rule[i]);
        if (decision == Deny)
        {
            return Deny;
        }
        if (decision == Permit)
        {
            atLeastOnePermit = true;
            continue;
        }
        if (decision == NotApplicable)
        {
            continue;
        }
        if (decision == Indeterminate)
        {
            atLeastOneError = true;
            if (effect(rule[i]) == Deny)
            {
                potentialDeny = true;
            }
            continue;
        }
    }
}
```

Deleted: cd-03

```

5152     }
5153   }
5154   if (potentialDeny)
5155   {
5156     return Indeterminate;
5157   }
5158   if (atLeastOnePermit)
5159   {
5160     return Permit;
5161   }
5162   if (atLeastOneError)
5163   {
5164     return Indeterminate;
5165   }
5166   return NotApplicable;
5167 }

```

5168 The following specification defines the "Deny-overrides" **policy-combining algorithm** of a **policy**  
5169 **set**.

5170 In the entire set of **policies** in the **policy set**, if any **policy** evaluates to "Deny", then the  
5171 result of the **policy** combination SHALL be "Deny". In other words, "Deny" takes  
5172 precedence, regardless of the result of evaluating any of the other **policies** in the **policy**  
5173 **set**. If all **policies** are found to be "NotApplicable" to the **decision request**, then the  
5174 **policy set** SHALL evaluate to "NotApplicable".

5175 If an error occurs while evaluating the **target** of a **policy**, or a reference to a **policy** is  
5176 considered invalid or the **policy** evaluation results in "Indeterminate", then the **policy set**  
5177 SHALL evaluate to "Deny".

5178 The following pseudo-code represents the evaluation strategy of this **policy-combining algorithm**.

```

5179 Decision denyOverridesPolicyCombiningAlgorithm(Policy policy[])
5180 {
5181   Boolean atLeastOnePermit = false;
5182   for( i=0 ; i < lengthOf(policy) ; i++ )
5183   {
5184     Decision decision = evaluate(policy[i]);
5185     if (decision == Deny)
5186     {
5187       return Deny;
5188     }
5189     if (decision == Permit)
5190     {
5191       atLeastOnePermit = true;
5192       continue;
5193     }
5194     if (decision == NotApplicable)
5195     {
5196       continue;
5197     }
5198     if (decision == Indeterminate)
5199     {
5200       return Deny;
5201     }
5202   }
5203   if (atLeastOnePermit)
5204   {
5205     return Permit;
5206   }
5207   return NotApplicable;
5208 }

```

5209 **Obligations** of the individual **policies** shall be combined as described in Section 7.14.

Deleted: cd-03

## C.2. Ordered-deny-overrides

The following specification defines the "Ordered-deny-overrides" *rule-combining algorithm* of a *policy*.

The behavior of this algorithm is identical to that of the Deny-overrides *rule-combining algorithm* with one exception. The order in which the collection of *rules* is evaluated SHALL match the order as listed in the *policy*.

The following specification defines the "Ordered-deny-overrides" *policy-combining algorithm* of a *policy set*.

The behavior of this algorithm is identical to that of the Deny-overrides *policy-combining algorithm* with one exception. The order in which the collection of *policies* is evaluated SHALL match the order as listed in the *policy set*.

## C.3. Permit-overrides

The following specification defines the "Permit-overrides" *rule-combining algorithm* of a *policy*.

In the entire set of *rules* in the *policy*, if any *rule* evaluates to "Permit", then the result of the *rule* combination SHALL be "Permit". If any *rule* evaluates to "Deny" and all other *rules* evaluate to "NotApplicable", then the *policy* SHALL evaluate to "Deny". In other words, "Permit" takes precedence, regardless of the result of evaluating any of the other *rules* in the *policy*. If all *rules* are found to be "NotApplicable" to the *decision request*, then the *policy* SHALL evaluate to "NotApplicable".

If an error occurs while evaluating the *target* or *condition* of a *rule* that contains an *effect* of "Permit" then the evaluation SHALL continue looking for a result of "Permit". If no other *rule* evaluates to "Permit", then the *policy* SHALL evaluate to "Indeterminate", with the appropriate error status.

If at least one *rule* evaluates to "Deny", all other *rules* that do not have evaluation errors evaluate to "Deny" or "NotApplicable" and all *rules* that do have evaluation errors contain an *effect* value of "Deny", then the *policy* SHALL evaluate to "Deny".

The following pseudo-code represents the evaluation strategy of this *rule-combining algorithm*.

```
Decision permitOverridesRuleCombiningAlgorithm(Rule rule[])
{
    Boolean atLeastOneError = false;
    Boolean potentialPermit = false;
    Boolean atLeastOneDeny = false;
    for( i=0 ; i < lengthOf(rule) ; i++ )
    {
        Decision decision = evaluate(rule[i]);
        if (decision == Deny)
        {
            atLeastOneDeny = true;
            continue;
        }
        if (decision == Permit)
        {
            return Permit;
        }
        if (decision == NotApplicable)
        {
            continue;
        }
    }
}
```

Deleted: cd-03



```

5257     }
5258     if (decision == Indeterminate)
5259     {
5260         atLeastOneError = true;
5261
5262         if (effect(rule[i]) == Permit)
5263         {
5264             potentialPermit = true;
5265         }
5266         continue;
5267     }
5268 }
5269 if (potentialPermit)
5270 {
5271     return Indeterminate;
5272 }
5273 if (atLeastOneDeny)
5274 {
5275     return Deny;
5276 }
5277 if (atLeastOneError)
5278 {
5279     return Indeterminate;
5280 }
5281 return NotApplicable;
5282 }

```

5283 The following specification defines the "Permit-overrides" **policy-combining algorithm** of a **policy**  
5284 **set**.

5285 In the entire set of **policies** in the **policy set**, if any **policy** evaluates to "Permit", then the  
5286 result of the **policy** combination SHALL be "Permit". In other words, "Permit" takes  
5287 precedence, regardless of the result of evaluating any of the other **policies** in the **policy**  
5288 **set**. If all **policies** are found to be "NotApplicable" to the **decision request**, then the  
5289 **policy set** SHALL evaluate to "NotApplicable".

5290 If an error occurs while evaluating the **target** of a **policy**, a reference to a **policy** is  
5291 considered invalid or the **policy** evaluation results in "Indeterminate", then the **policy set**  
5292 SHALL evaluate to "Indeterminate", with the appropriate error status, provided no other  
5293 **policies** evaluate to "Permit" or "Deny".

5294 The following pseudo-code represents the evaluation strategy of this **policy-combining algorithm**.

```

5295 Decision permitOverridesPolicyCombiningAlgorithm(Policy policy[])
5296 {
5297     Boolean atLeastOneError = false;
5298     Boolean atLeastOneDeny = false;
5299     for( i=0 ; i < lengthOf(policy) ; i++ )
5300     {
5301         Decision decision = evaluate(policy[i]);
5302         if (decision == Deny)
5303         {
5304             atLeastOneDeny = true;
5305             continue;
5306         }
5307         if (decision == Permit)
5308         {
5309             return Permit;
5310         }
5311         if (decision == NotApplicable)
5312         {
5313             continue;
5314         }
5315     }
5316 }

```

Deleted: cd-03

```

5315     if (decision == Indeterminate)
5316     {
5317         atLeastOneError = true;
5318         continue;
5319     }
5320 }
5321 if (atLeastOneDeny)
5322 {
5323     return Deny;
5324 }
5325 if (atLeastOneError)
5326 {
5327     return Indeterminate;
5328 }
5329 return NotApplicable;
5330 }

```

5331 **Obligations** of the individual **policies** shall be combined as described in Section 7.14.

## 5332 C.4. Ordered-permit-overrides

5333 The following specification defines the "Ordered-permit-overrides" **rule-combining algorithm** of a  
5334 **policy**.

5335 The behavior of this algorithm is identical to that of the Permit-overrides **rule-combining**  
5336 **algorithm** with one exception. The order in which the collection of **rules** is evaluated SHALL  
5337 match the order as listed in the **policy**.

5338 The following specification defines the "Ordered-permit-overrides" **policy-combining algorithm** of  
5339 a **policy set**.

5340 The behavior of this algorithm is identical to that of the Permit-overrides **policy-combining**  
5341 **algorithm** with one exception. The order in which the collection of **policies** is evaluated  
5342 SHALL match the order as listed in the **policy set**.

## 5343 C.5. First-applicable

5344 The following specification defines the "First-Applicable" **rule-combining algorithm** of a **policy**.

5345 Each **rule** SHALL be evaluated in the order in which it is listed in the **policy**. For a  
5346 particular **rule**, if the **target** matches and the **condition** evaluates to "True", then the  
5347 evaluation of the **policy** SHALL halt and the corresponding **effect** of the **rule** SHALL be the  
5348 result of the evaluation of the **policy** (i.e. "Permit" or "Deny"). For a particular **rule** selected  
5349 in the evaluation, if the **target** evaluates to "False" or the **condition** evaluates to "False",  
5350 then the next **rule** in the order SHALL be evaluated. If no further **rule** in the order exists,  
5351 then the **policy** SHALL evaluate to "NotApplicable".

5352 If an error occurs while evaluating the **target** or **condition** of a **rule**, then the evaluation  
5353 SHALL halt, and the **policy** shall evaluate to "Indeterminate", with the appropriate error  
5354 status.

5355 The following pseudo-code represents the evaluation strategy of this **rule-combining algorithm**.

5356

5357

Deleted: cd-03

```

5358 Decision firstApplicableEffectRuleCombiningAlgorithm(Rule rule[])
5359 {
5360     for( i = 0 ; i < lengthOf(rule) ; i++ )
5361     {
5362         Decision decision = evaluate(rule[i]);
5363         if (decision == Deny)
5364         {
5365             return Deny;
5366         }
5367         if (decision == Permit)
5368         {
5369             return Permit;
5370         }
5371         if (decision == NotApplicable)
5372         {
5373             continue;
5374         }
5375         if (decision == Indeterminate)
5376         {
5377             return Indeterminate;
5378         }
5379     }
5380     return NotApplicable;
5381 }

```

5382 The following specification defines the "First-applicable" *policy-combining algorithm* of a *policy*  
5383 *set*.

5384 Each *policy* is evaluated in the order that it appears in the *policy set*. For a particular  
5385 *policy*, if the *target* evaluates to "True" and the *policy* evaluates to a determinate value of  
5386 "Permit" or "Deny", then the evaluation SHALL halt and the *policy set* SHALL evaluate to  
5387 the *effect* value of that *policy*. For a particular *policy*, if the *target* evaluate to "False", or  
5388 the *policy* evaluates to "NotApplicable", then the next *policy* in the order SHALL be  
5389 evaluated. If no further *policy* exists in the order, then the *policy set* SHALL evaluate to  
5390 "NotApplicable".

5391 If an error were to occur when evaluating the *target*, or when evaluating a specific *policy*,  
5392 the reference to the *policy* is considered invalid, or the *policy* itself evaluates to  
5393 "Indeterminate", then the evaluation of the *policy-combining algorithm* shall halt, and the  
5394 *policy set* shall evaluate to "Indeterminate" with an appropriate error status.

5395 The following pseudo-code represents the evaluation strategy of this *policy-combination*  
5396 *algorithm*.

```

5397 Decision firstApplicableEffectPolicyCombiningAlgorithm(Policy policy[])
5398 {
5399     for( i = 0 ; i < lengthOf(policy) ; i++ )
5400     {
5401         Decision decision = evaluate(policy[i]);
5402         if (decision == Deny)
5403         {
5404             return Deny;
5405         }
5406         if (decision == Permit)
5407         {
5408             return Permit;
5409         }
5410         if (decision == NotApplicable)
5411         {
5412             continue;
5413         }
5414         if (decision == Indeterminate)
5415         {

```

Deleted: cd-03

```

5416         return Indeterminate;
5417     }
5418 }
5419 return NotApplicable;
5420 }

```

5421 **Obligations** of the individual **policies** shall be combined as described in Section 7.14.

## 5422 C.6. Only-one-applicable

5423 The following specification defines the "Only-one-applicable" **policy-combining algorithm** of a  
5424 **policy set**.

5425 In the entire set of **policies** in the **policy set**, if no **policy** is considered applicable by virtue  
5426 of its **target**, then the result of the **policy** combination algorithm SHALL be "NotApplicable".  
5427 If more than one **policy** is considered applicable by virtue of its **target**, then the result of  
5428 the **policy** combination algorithm SHALL be "Indeterminate".

5429 If only one **policy** is considered applicable by evaluation of its **target**, then the result of the  
5430 **policy-combining algorithm** SHALL be the result of evaluating the **policy**.

5431 If an error occurs while evaluating the **target** of a **policy**, or a reference to a **policy** is  
5432 considered invalid or the **policy** evaluation results in "Indeterminate", then the **policy set**  
5433 SHALL evaluate to "Indeterminate", with the appropriate error status.

5434 The following pseudo-code represents the evaluation strategy of this policy combining algorithm.

```

5435 Decision onlyOneApplicablePolicyPolicyCombiningAlgorithm(Policy policy[])
5436 {
5437     Boolean          atLeastOne      = false;
5438     Policy           selectedPolicy = null;
5439     ApplicableResult appResult;
5440
5441     for ( i = 0; i < lengthOf(policy) ; i++ )
5442     {
5443         appResult = isApplicable(policy[i]);
5444
5445         if ( appResult == Indeterminate )
5446         {
5447             return Indeterminate;
5448         }
5449         if( appResult == Applicable )
5450         {
5451             if ( atLeastOne )
5452             {
5453                 return Indeterminate;
5454             }
5455             else
5456             {
5457                 atLeastOne      = true;
5458                 selectedPolicy = policy[i];
5459             }
5460         }
5461         if ( appResult == NotApplicable )
5462         {
5463             continue;
5464         }
5465     }
5466     if ( atLeastOne )
5467     {

```

Deleted: cd-03

```
5468         return evaluate(selectedPolicy);
5469     }
5470     else
5471     {
5472         return NotApplicable;
5473     }
5474 }
5475
```

## Appendix D. Acknowledgments

The following individuals contributed to the development of the specification:

Anne Anderson  
Anthony Nadalin  
Bill Parducci  
Carlisle Adams  
Daniel Engovatov  
Don Flinn  
Ed Coyne  
Ernesto Damiani  
Frank Siebenlist  
Gerald Brose  
Hal Lockhart  
Haruyuki Kawabe  
James MacLean  
John Merrells  
Ken Yagen  
Konstantin Beznosov  
Michiharu Kudo  
Michael McIntosh  
Pierangela Samarati  
Pirasenna Velandai Thiyagarajan  
Polar Humenn  
Rebekah Metz  
Ron Jacobson  
Satoshi Hada  
Sekhar Vajjhala  
Seth Proctor  
Simon Godik  
Steve Anderson  
Steve Crocker  
Suresh Damodaran  
Tim Moses  
Von Welch

5511

## Appendix E. Revision history

| Rev          | Date              | By whom                  | What   |
|--------------|-------------------|--------------------------|--|
| CD 01        | 16 Sep 2004       | Access Control TC        | First committee draft  |
| CD 02        | 30 Sep 2004       | Access Control TC        | Updated list of editors  |
| CD 03        | 10 Nov 2004       | Access control TC        | Editorial corrections to Haskell source code for higher-order bag functions and corrected identifiers for data-types and functions introduced in Version 2.0   |
| <u>CD 04</u> | <u>6 Dec 2004</u> | <u>Access control TC</u> | <u>Changed function names such as "... string-regexp-match" to "... string-regexp-match", etc., in order to conform with the naming convention used elsewhere in the document.</u><br><u>Corrected Figure 1.</u> |

5512

access\_control-xacml-2.0-core-spec-cd-04

141

Deleted: cd-03



---

## Appendix F. Notices

5513

5514 OASIS takes no position regarding the validity or scope of any intellectual property or other rights  
5515 that might be claimed to pertain to the implementation or use of the technology described in this  
5516 document or the extent to which any license under such rights might or might not be available;  
5517 neither does it represent that it has made any effort to identify any such rights. Information on  
5518 OASIS's procedures with respect to rights in OASIS specifications can be found at the OASIS  
5519 website. Copies of claims of rights made available for publication and any assurances of licenses to  
5520 be made available, or the result of an attempt made to obtain a general license or permission for  
5521 the use of such proprietary rights by implementers or users of this specification, can be obtained  
5522 from the OASIS Executive Director.

5523 OASIS has been notified of intellectual property rights claimed in regard to some or all of the  
5524 contents of this specification. For more information consult the online list of claimed rights.

5525 OASIS invites any interested party to bring to its attention any copyrights, patents or patent  
5526 applications, or other proprietary rights which may cover technology that may be required to  
5527 implement this specification. Please address the information to the OASIS Executive Director.

5528 Copyright (C) OASIS Open 2004. All Rights Reserved.

5529 This document and translations of it may be copied and furnished to others, and derivative works  
5530 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,  
5531 published and distributed, in whole or in part, without restriction of any kind, provided that the above  
5532 copyright notice and this paragraph are included on all such copies and derivative works. However,  
5533 this document itself may not be modified in any way, such as by removing the copyright notice or  
5534 references to OASIS, except as needed for the purpose of developing OASIS specifications, in  
5535 which case the procedures for copyrights defined in the OASIS Intellectual Property Rights  
5536 document must be followed, or as required to translate it into languages other than English.

5537 The limited permissions granted above are perpetual and will not be revoked by OASIS or its  
5538 successors or assigns.

5539 This document and the information contained herein is provided on an "AS IS" basis and OASIS  
5540 DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO  
5541 ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY  
5542 RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A  
5543 PARTICULAR PURPOSE.

Deleted: cd-03

|                     |   |                   |
|---------------------|---|-------------------|
| Page 8: [1] Deleted | Entrust User  | 12/6/2004 1:07 PM |
| 1.                  | Introduction (non-normative) .....  | 9                 |
| 1.1.                | Glossary .....  | 9                 |
| 1.1.1               | Preferred terms.....  | 9                 |
| 1.1.2               | Related terms.....  | 10                |
| 1.2.                | Notation.....   | 11                |
| 1.3.                | Schema organization and namespaces .....                                    | 11                |
| 2.                  | Background (non-normative).....   | 12                |
| 2.1.                | Requirements .....  | 12                |
| 2.2.                | Rule and policy combining .....   | 13                |
| 2.3.                | Combining algorithms.....   | 13                |
| 2.4.                | Multiple subjects .....   | 14                |
| 2.5.                | Policies based on subject and resource attributes.....                      | 14                |
| 2.6.                | Multi-valued attributes .....   | 15                |
| 2.7.                | Policies based on resource contents .....                                   | 15                |
| 2.8.                | Operators.....  | 15                |
| 2.9.                | Policy distribution .....   | 16                |
| 2.10.               | Policy indexing .....   | 16                |
| 2.11.               | Abstraction layer .....   | 16                |
| 2.12.               | Actions performed in conjunction with enforcement.....                      | 17                |
| 3.                  | Models (non-normative) .....  | 17                |
| 3.1.                | Data-flow model.....  | 17                |
| 3.2.                | XACML context.....  | 19                |
| 3.3.                | Policy language model .....   | 19                |
| 3.3.1               | Rule.....   | 21                |
| 3.3.2               | Policy.....   | 22                |
| 3.3.3               | Policy set .....  | 23                |
| 4.                  | Examples (non-normative) .....  | 23                |
| 4.1.                | Example one .....   | 24                |
| 4.1.1               | Example policy .....  | 24                |
| 4.1.2               | Example request context .....   | 26                |
| 4.1.3               | Example response context.....   | 27                |
| 4.2.                | Example two .....   | 27                |
| 4.2.1               | Example medical record instance .....                                       | 27                |
| 4.2.2               | Example request context .....   | 29                |
| 4.2.3               | Example plain-language rules.....   | 30                |
| 4.2.4               | Example XACML rule instances.....   | 31                |
| 5.                  | Policy syntax (normative, with the exception of the schema fragments) ..... | 43                |
| 5.1.                | Element <PolicySet>.....  | 43                |
| 5.2.                | Element <Description> .....   | 46                |
| 5.3.                | Element <PolicySetDefaults>.....  | 46                |
| 5.4.                | Element <XPathVersion> .....  | 46                |
| 5.5.                | Element <Target>.....   | 46                |
| 5.6.                | Element <Subjects> .....  | 47                |
| 5.7.                | Element <Subject> .....   | 47                |
| 5.8.                | Element <SubjectMatch> .....  | 48                |
| 5.9.                | Element <Resources> .....   | 48                |

|       |   |    |
|-------|---|----|
| 5.10. | Element <Resource> .....  | 49 |
| 5.11. | Element <ResourceMatch> .....   | 49 |
| 5.12. | Element <Actions> .....   | 50 |
| 5.13. | Element <Action> .....  | 50 |
| 5.14. | Element <ActionMatch> .....   | 50 |
| 5.15. | Element <Environments> .....  | 51 |
| 5.16. | Element <Environment> .....   | 51 |
| 5.17. | Element <EnvironmentMatch> .....  | 52 |
| 5.18. | Element <PolicySetIdReference> .....  | 52 |
| 5.19. | Element <PolicyIdReference> .....   | 53 |
| 5.20. | Simple type VersionType .....   | 53 |
| 5.21. | Simple type VersionMatchType .....  | 53 |
| 5.22. | Element <Policy> .....  | 54 |
| 5.23. | Element <PolicyDefaults> .....  | 56 |
| 5.24. | Element <CombinerParameters> .....  | 56 |
| 5.25. | Element <CombinerParameter> .....   | 56 |
| 5.26. | Element <RuleCombinerParameters> .....                                      | 57 |
| 5.27. | Element <PolicyCombinerParameters> .....                                    | 57 |
| 5.28. | Element <PolicySetCombinerParameters> .....                                 | 58 |
| 5.29. | Element <Rule> .....  | 59 |
| 5.30. | Simple type EffectType .....  | 59 |
| 5.31. | Element <VariableDefinition> .....  | 60 |
| 5.32. | Element <VariableReference> .....   | 60 |
| 5.33. | Element <Expression> .....  | 61 |
| 5.34. | Element <Condition> .....   | 61 |
| 5.35. | Element <Apply> .....   | 61 |
| 5.36. | Element <Function> .....  | 62 |
| 5.37. | Complex type AttributeDesignatorType .....                                  | 62 |
| 5.38. | Element <SubjectAttributeDesignator> .....                                  | 63 |
| 5.39. | Element <ResourceAttributeDesignator> .....                                 | 64 |
| 5.40. | Element <ActionAttributeDesignator> .....                                   | 64 |
| 5.41. | Element <EnvironmentAttributeDesignator> .....                              | 65 |
| 5.42. | Element <AttributeSelector> .....   | 65 |
| 5.43. | Element <AttributeValue> .....  | 67 |
| 5.44. | Element <Obligations> .....   | 67 |
| 5.45. | Element <Obligation> .....  | 67 |
| 5.46. | Element <AttributeAssignment> .....   | 68 |
| 6.    | Context syntax (normative with the exception of the schema fragments) ..... | 69 |
| 6.1.  | Element <Request> .....   | 69 |
| 6.2.  | Element <Subject> .....   | 70 |
| 6.3.  | Element <Resource> .....  | 70 |
| 6.4.  | Element <ResourceContent> .....   | 71 |
| 6.5.  | Element <Action> .....  | 71 |
| 6.6.  | Element <Environment> .....   | 72 |
| 6.7.  | Element <Attribute> .....   | 72 |
| 6.8.  | Element <AttributeValue> .....  | 73 |

|         |   |    |
|---------|---|----|
| 6.9.    | Element <Response>.....                                   | 73 |
| 6.10.   | Element <Result> .....                                    | 74 |
| 6.11.   | Element <Decision> .....                                  | 74 |
| 6.12.   | Element <Status> .....                                    | 75 |
| 6.13.   | Element <StatusCode>.....                                 | 75 |
| 6.14.   | Element <StatusMessage>.....                              | 76 |
| 6.15.   | Element <StatusDetail> .....                              | 76 |
| 6.16.   | Element <MissingAttributeDetail> .....                    | 77 |
| 7.      | Functional requirements (normative).....                  | 77 |
| 7.1.    | Policy enforcement point .....                            | 77 |
| 7.1.1.  | Base PEP .....  | 78 |
| 7.1.2.  | Deny-biased PEP .....                                     | 78 |
| 7.1.3.  | Permit-biased PEP .....                                   | 78 |
| 7.2.    | Attribute evaluation.....                                 | 78 |
| 7.2.1.  | Structured attributes.....                                | 78 |
| 7.2.2.  | Attribute bags.....                                       | 79 |
| 7.2.3.  | Multivalued attributes.....                               | 79 |
| 7.2.4.  | Attribute Matching .....                                  | 79 |
| 7.2.5.  | Attribute Retrieval .....                                 | 80 |
| 7.2.6.  | Environment Attributes .....                              | 80 |
| 7.3.    | Expression evaluation .....                               | 80 |
| 7.4.    | Arithmetic evaluation.....                                | 81 |
| 7.5.    | Match evaluation .....                                    | 81 |
| 7.6.    | Target evaluation.....                                    | 83 |
| 7.7.    | VariableReference Evaluation.....                         | 84 |
| 7.8.    | Condition evaluation .....                                | 84 |
| 7.9.    | Rule evaluation .....                                     | 84 |
| 7.10.   | Policy evaluation .....                                   | 85 |
| 7.11.   | Policy Set evaluation.....                                | 86 |
| 7.12.   | Hierarchical resources.....                               | 87 |
| 7.13.   | Authorization decision .....                              | 87 |
| 7.14.   | Obligations.....  | 87 |
| 7.15.   | Exception handling.....                                   | 87 |
| 7.15.1. | Unsupported functionality.....                            | 88 |
| 7.15.2. | Syntax and type errors .....                              | 88 |
| 7.15.3. | Missing attributes.....                                   | 88 |
| 8.      | XACML extensibility points (non-normative).....           | 88 |
| 8.1.    | Extensible XML attribute types.....                       | 89 |
| 8.2.    | Structured attributes .....                               | 89 |
| 9.      | Security and privacy considerations (non-normative) ..... | 89 |
| 9.1.    | Threat model.....   | 90 |
| 9.1.1.  | Unauthorized disclosure .....                             | 90 |
| 9.1.2.  | Message replay.....                                       | 90 |
| 9.1.3.  | Message insertion .....                                   | 90 |
| 9.1.4.  | Message deletion.....                                     | 91 |
| 9.1.5.  | Message modification.....                                 | 91 |

|             |   |     |
|-------------|---|-----|
| 9.1.6.      | NotApplicable results .....                   | 91  |
| 9.1.7.      | Negative rules .....                          | 91  |
| 9.2.        | Safeguards.....                               | 92  |
| 9.2.1.      | Authentication.....                           | 92  |
| 9.2.2.      | Policy administration.....                    | 92  |
| 9.2.3.      | Confidentiality .....                         | 93  |
| 9.2.4.      | Policy integrity.....                         | 93  |
| 9.2.5.      | Policy identifiers .....                      | 94  |
| 9.2.6.      | Trust model .....                             | 94  |
| 9.2.7.      | Privacy.....                                  | 94  |
| 10.         | Conformance (normative).....                  | 95  |
| 10.1.       | Introduction.....                             | 95  |
| 10.2.       | Conformance tables.....                       | 95  |
| 10.2.1.     | Schema elements.....                          | 95  |
| 10.2.2.     | Identifier Prefixes .....                     | 96  |
| 10.2.3.     | Algorithms.....                               | 96  |
| 10.2.4.     | Status Codes .....                            | 97  |
| 10.2.5.     | Attributes .....                              | 97  |
| 10.2.6.     | Identifiers .....                             | 97  |
| 10.2.7.     | Data-types .....                              | 98  |
| 10.2.8.     | Functions .....                               | 98  |
| 11.         | References.....                               | 102 |
| Appendix A. | Data-types and functions (normative).....     | 105 |
| A.1.        | Introduction.....                             | 105 |
| A.2.        | Data-types .....                              | 105 |
| A.3.        | Functions.....                                | 107 |
| A.3.1       | Equality predicates .....                     | 107 |
| A.3.2       | Arithmetic functions .....                    | 109 |
| A.3.3       | String conversion functions .....             | 110 |
| A.3.4       | Numeric data-type conversion functions.....   | 110 |
| A.3.5       | Logical functions .....                       | 110 |
| A.3.6       | Numeric comparison functions.....             | 111 |
| A.3.7       | Date and time arithmetic functions.....       | 112 |
| A.3.8       | Non-numeric comparison functions .....        | 113 |
| A.3.9       | String functions.....                         | 116 |
| A.3.10      | Bag functions .....                           | 116 |
| A.3.11      | Set functions.....                            | 117 |
| A.3.12      | Higher-order bag functions .....              | 117 |
| A.3.13      | Regular-expression-based functions.....       | 124 |
| A.3.14      | Special match functions.....                  | 125 |
| A.3.15      | XPath-based functions.....                    | 126 |
| A.3.16      | Extension functions and primitive types ..... | 126 |
| Appendix B. | XACML identifiers (normative).....            | 127 |
| B.1.        | XACML namespaces .....                        | 127 |
| B.2.        | Access subject categories.....                | 127 |
| B.3.        | Data-types .....                              | 127 |

|   |     |
|---|-----|
| B.4. Subject attributes .....                     | 128 |
| B.6. Resource attributes .....                    | 129 |
| B.7. Action attributes .....                      | 129 |
| B.8. Environment attributes.....                  | 130 |
| B.9. Status codes .....                           | 130 |
| B.10. Combining algorithms .....                  | 130 |
| Appendix C. Combining algorithms (normative)..... | 132 |
| C.1. Deny-overrides.....                          | 132 |
| C.2. Ordered-deny-overrides.....                  | 134 |
| C.3. Permit-overrides.....                        | 134 |
| C.4. Ordered-permit-overrides.....                | 136 |
| C.5. First-applicable .....                       | 136 |
| C.6. Only-one-applicable .....                    | 138 |
| Appendix D. Acknowledgments.....                  | 140 |
| Appendix E. Revision history .....                | 141 |
| Appendix F. Notices .....                         | 142 |