



JSON Profile of XACML 3.0 Version 1.1

Candidate OASIS Standard 01

28 March 2019

Specification URIs

This version:

<https://docs.oasis-open.org/xacml/xacml-json-http/v1.1/cos01/xacml-json-http-v1.1-cos01.doc>
(Authoritative)
<https://docs.oasis-open.org/xacml/xacml-json-http/v1.1/cos01/xacml-json-http-v1.1-cos01.html>
<https://docs.oasis-open.org/xacml/xacml-json-http/v1.1/cos01/xacml-json-http-v1.1-cos01.pdf>

Previous version:

N/A

Latest version:

<https://docs.oasis-open.org/xacml/xacml-json-http/v1.1/xacml-json-http-v1.1.doc> (Authoritative)
<https://docs.oasis-open.org/xacml/xacml-json-http/v1.1/xacml-json-http-v1.1.html>
<https://docs.oasis-open.org/xacml/xacml-json-http/v1.1/xacml-json-http-v1.1.pdf>

Technical Committee:

OASIS eXtensible Access Control Markup Language (XACML) TC

Chairs:

Hal Lockhart (hal.lockhart@oracle.com), Oracle
Bill Parducci (bill@parducci.net), Individual

Editors:

David Brossard (david.brossard@axiomatics.com), Axiomatics AB
Steven Legg (steven.legg@viewds.com), Individual

Related work:

This specification replaces or supersedes:

- *JSON Profile of XACML 3.0 Version 1.0*. Edited by David Brossard. Latest version: <http://docs.oasis-open.org/xacml/xacml-json-http/v1.0/xacml-json-http-v1.0.html>.

This specification is related to:

- *eXtensible Access Control Markup Language (XACML) Version 3.0 Plus Errata 01*. Edited by Erik Rissanen. 12 July 2017. OASIS Standard incorporating Approved Errata. <http://docs.oasis-open.org/xacml/3.0/errata01/os/xacml-3.0-core-spec-errata01-os-complete.html>. Latest version: <http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-en.html>.

Abstract:

The aim of this profile is to define a standardized interface between a policy enforcement point and a policy decision point using JSON. The decision request and response structure is specified in the core XACML specification. This profile leverages it.

Status:

This document was last revised or approved by the OASIS eXtensible Access Control Markup Language (XACML) TC on the above date. The level of approval is also listed above. Check the "Latest version" location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml#technical.

TC members should send comments on this specification to the TC's email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "[Send A Comment](#)" button on the TC's web page at <https://www.oasis-open.org/committees/xacml/>.

This specification is provided under the [RF on Limited Terms](#) Mode of the [OASIS IPR Policy](#), the mode chosen when the Technical Committee was established. For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (<https://www.oasis-open.org/committees/xacml/ipr.php>).

Note that any machine-readable content ([Computer Language Definitions](#)) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

Citation format:

When referencing this specification the following citation format should be used:

[xacml-json-v1.1]

JSON Profile of XACML 3.0 Version 1.1. Edited by David Brossard and Steven Legg. 28 March 2019. Candidate OASIS Standard 01. <https://docs.oasis-open.org/xacml/xacml-json-http/v1.1/cos01/xacml-json-http-v1.1-cos01.html>. Latest version: <https://docs.oasis-open.org/xacml/xacml-json-http/v1.1/xacml-json-http-v1.1.html>.

Notices

Copyright © OASIS Open 2019. All Rights Reserved.

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

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

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

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

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

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

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The name "OASIS" is a trademark of [OASIS](#), the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see <https://www.oasis-open.org/policies-guidelines/trademark> for above guidance.

Table of Contents

1	Introduction	6
1.1	IPR Policy	6
1.2	Glossary	6
1.3	Terminology	7
1.4	Normative References	7
1.5	Non-Normative References	7
2	Vocabulary	8
3	Overview of the translation mechanisms	9
3.1	Assumed default values	9
3.2	Objects	9
3.2.1	Member names	9
3.2.2	Object order	9
3.2.3	Object cardinality	9
3.2.4	Null values	9
3.3	Data Types	10
3.3.1	Supported Data Types	10
3.3.2	Multiple values	11
3.3.3	The xpathExpression Datatype	11
3.3.3.1	Example	12
3.3.4	Special numeric values	12
3.4	Example	13
4	The XACML request	14
4.1	Class Diagram	14
4.2	Representation of the XACML request in JSON	14
4.2.1	The Request object representation	14
4.2.1.1	Example	15
4.2.2	The Category object representation	16
4.2.2.1	Shorthand notation for standard XACML categories	16
4.2.2.2	Default Category objects	16
4.2.2.3	Example	17
4.2.3	The Content object representation	17
4.2.3.1	XML Escaping	17
4.2.3.2	Base64 Encoding	17
4.2.3.3	Example	18
4.2.4	The Attribute object representation	18
4.2.4.1	Example	19
4.2.5	The MultiRequests object representation	19
4.2.6	The RequestReference object representation	19
4.2.6.1	Non-normative example	19
5	The XACML response	21
5.1	Class Diagram	21
5.2	Representation of the XACML response in JSON	21
5.2.1	The Result object representation	21
5.2.2	The Status object representation	22

5.2.3	The MissingAttributeDetail object.....	22
5.2.4	The StatusCode object representation.....	23
5.2.4.1	Example.....	23
5.2.5	The ObligationOrAdvice object representation	24
5.2.6	The AttributeAssignment object representation	24
5.2.7	The PolicyIdentifierList object representation	25
5.2.8	The IdReference object representation	25
6	Transport	26
6.1	Transport Security	26
7	IANA Registration	27
7.1	Media Type Name	27
7.2	Subtype Name	27
7.3	Required Parameters.....	27
7.4	Optional Parameters.....	27
7.5	Encoding Considerations.....	27
7.6	Security Considerations.....	27
7.7	Interoperability Considerations	27
7.8	Applications which use this media type	27
7.9	Magic number(s).....	27
7.10	File extension(s)	27
7.11	Macintosh File Type Code(s).....	28
7.12	Intended Usage	28
8	Examples.....	29
8.1	Request Example	29
8.2	Response Example.....	29
8.3	Request for Multiple Decisions Example	30
8.4	Multiple Decisions Response Example	31
9	Conformance	33
Appendix A.	Acknowledgments	34
Appendix B.	Revision History	35

1 Introduction

[All text is normative unless otherwise labeled]

{Non-normative}

The XACML architecture promotes a loose coupling between the component that enforces decisions, the policy enforcement point (PEP), and the component that decides based on XACML policies, the policy decision point (PDP).

The XACML standard defines the format of the request and the response between the PEP and the PDP. As the default representation of XACML is XML and is backed by a schema, the request and response are typically expressed as XML elements or documents. Depending on the PDP implementation, the request and response could be embedded inside a SOAP message or even a SAML assertion as described in the SAML profile of XACML.

With the rise in popularity of APIs and its consumerization, it becomes important for XACML to be easily understood in order to increase the likelihood it will be adopted.

This profile aims at defining a JSON format for the XACML request and response. It also defines the transport between client (PEP) and service (PDP).

In writing this document, the authors have kept three items in mind:

1. Equivalence: a XACML request and response expressed in XML need not be strictly equivalent in structure to a XACML request expressed in JSON so long as the meaning remains the same and so long as the JSON and XML requests would lead to the same response (decision, obligation, and advice).
2. Lossless behavior: it **MUST** be possible to translate XACML requests and responses between XML and JSON representations in either direction at any time without semantic loss.
3. Transport-agnostic nature: the JSON representation **MUST** contain all the information the XACML request and/or response contains: this means the transport layer cannot convert XACML decisions into HTTP codes, e.g. HTTP 401 for a Deny decision.

1.1 IPR Policy

This specification is provided under the [RF on Limited Terms](#) Mode of the [OASIS IPR Policy](#), the mode chosen when the Technical Committee was established. For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (<https://www.oasis-open.org/committees/xacml/ipr.php>).

1.2 Glossary

Array

An ordered sequence of zero or more JSON *elements*.

Element

In JSON, a value; either a JSON primitive type, an *object* or an *array*.

Member

A name/value pair in a JSON *object*.

Object

In JSON, an unordered collection of zero or more *members*.

41 1.3 Terminology

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

45 1.4 Normative References

- 46 [RFC2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*,
47 <http://www.ietf.org/rfc/rfc2119.txt>, IETF RFC 2119, March 1997.
- 48 [RFC4627] D. Crockford, *The application/json Media Type for JavaScript Object Notation*
49 *(JSON)*, <http://tools.ietf.org/html/rfc4627>, IETF RFC 4627, July 2006.
- 50 [XACMLMDP] *XACML v3.0 Multiple Decision Profile Version 1.0*. [http://docs.oasis-](http://docs.oasis-open.org/xacml/3.0/multiple/v1.0/xacml-3.0-multiple-v1.0.html)
51 [open.org/xacml/3.0/multiple/v1.0/xacml-3.0-multiple-v1.0.html](http://docs.oasis-open.org/xacml/3.0/multiple/v1.0/xacml-3.0-multiple-v1.0.html)
- 52 [RFC8259] T. Bray, Ed., *The JavaScript Object Notation (JSON) Data Interchange Format*,
53 <https://tools.ietf.org/html/rfc8259>, IETF RFC 8259, December 2017.
- 54 [NAMESPACES] Bray, Tim, et.al. eds, *Namespaces in XML 1.0 (Third Edition)*, W3C
55 Recommendation 8 December 2009, available at
56 <http://www.w3.org/TR/2009/REC-xml-names-20091208/>
- 57 [XACML30] *eXtensible Access Control Markup Language (XACML) Version 3.0 Plus Errata*
58 *01*. Edited by Erik Rissanen. 12 July 2018. OASIS Standard incorporating
59 Approved Errata. [http://docs.oasis-open.org/xacml/3.0/errata01/os/xacml-3.0-](http://docs.oasis-open.org/xacml/3.0/errata01/os/xacml-3.0-core-spec-errata01-os-complete.html)
60 [core-spec-errata01-os-complete.html](http://docs.oasis-open.org/xacml/3.0/errata01/os/xacml-3.0-core-spec-errata01-os-complete.html). Latest version: [http://docs.oasis-](http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-en.html)
61 [open.org/xacml/3.0/xacml-3.0-core-spec-en.html](http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-en.html).
- 62 [XML] Bray, Tim, et.al. eds, *Extensible Markup Language (XML) 1.0 (Fifth Edition)*,
63 W3C Recommendation 26 November 2008, available at
64 <http://www.w3.org/TR/2008/REC-xml-20081126/>
- 65 [XMLDatatypes] Biron, Paul et al. Eds, *XML Schema Part 2: Datatypes Second Edition*, W3C
66 Recommendation 28 October 2004, available at
67 <http://www.w3.org/TR/xmlschema-2/>
- 68 [XPATH] James Clark and Steve DeRose, *XML Path Language (XPath), Version 1.0*, W3C
69 Recommendation 16 November 1999. Available at: <http://www.w3.org/TR/xpath>
- 70 [IEEE754] Institute of Electrical and Electronics Engineers, "Standard for Floating-Point
71 Arithmetic", IEEE Standard 754, August 2008.

72 1.5 Non-Normative References

- 73 [XACMLREST] *REST Profile of XACML v3.0 Version 1.0*. Edited by Rémon Sinnema. Latest
74 version: <http://docs.oasis-open.org/xacml/xacml-rest/v1.0/xacml-rest-v1.0.doc>.
- 75 [HTTP] *Hypertext Transfer Protocol*. June 1999. IETF RFC 2616.
76 <http://tools.ietf.org/html/rfc2616>
- 77 [HTTPS] *HTTP over TLS*. May 2000. IETF RFC 2818. <http://tools.ietf.org/html/rfc2818>
- 78 [BASE64] *The Base16, Base32, and Base64 Data Encodings*. October 2006. IETF RFC
79 4648. <http://tools.ietf.org/html/rfc4648>

80 **2 Vocabulary**

81 **{Non-normative}**

82 XML introduces the notion of elements. The equivalent notion in JSON is an ***object***. XML introduces the
83 notion of attributes. The equivalent notion in JSON is a ***member***.

84 3 Overview of the translation mechanisms

85 3.1 Assumed default values

86 To avoid bloating the JSON request and response, certain parts of a request and response have default
87 values which can then be omitted. As an example, the default value for the data-type of an attribute value
88 is `String` (<http://www.w3.org/2001/XMLSchema#string>).

89 The user should refer to the XACML 3.0 specification document [XACML30] for a normative definition of
90 the request and response elements.

91 3.2 Objects

92 3.2.1 Member names

93 Unless otherwise stated, JSON *member* names MUST match the XACML XML element and/or attribute
94 names exactly, including case.

95 The following XML elements and attributes have been renamed:

- 96 • The name of the XACML XML `<Attributes>` element has been changed in JSON to the
97 *member* name "Category". It makes more sense to call the parent element that way since it
98 represents an instance of a category from a XACML sense.
- 99 • The `<AttributeValue>` element in the XML representation does not have an *object*
100 counterpart in the JSON representation. The information it bears in XML is moved to the parent
101 Attribute *object* in the JSON representation. A "Value" *member* has been introduced in the
102 Attribute *object* to bear the information contained in the XML `<AttributeValue>` element as
103 specified in section 4. The XACML request.
- 104 • The `AdviceId` and the `ObligationId` attributes of the `<Advice>` and the `<Obligation>`
105 XML elements respectively have been renamed to "Id" in JSON.

106 3.2.2 Object order

107 The order of the objects and values in XACML does not matter. Therefore, the order of *objects* and
108 values in the serialized form (JSON) does not matter.

109 3.2.3 Object cardinality

110 When in the XACML specification an XML element occurs zero or more times, the JSON equivalent is an
111 optional *member* with an *array* for the value. The *array* MAY be empty and this case is semantically
112 equivalent to the *member* being omitted from the containing *object*.

113 When in the XACML specification an XML element occurs one or more times, the JSON equivalent is a
114 mandatory *member* with an *array* for the value. The *array* MUST have at least one *element*.

115 The class diagram in section 4.1. Class Diagram states the cardinality and relationship between kinds of
116 *objects*.

117 3.2.4 Null values

118 The JSON null value is not permitted, including as an *element* in an *array*. If an optional, non-array
119 *member* has no value then it MUST be omitted from the containing *object*. A mandatory, non-array
120 *member* MUST have a non-null value.

121 **3.3 Data Types**

122 This section defines how data-types are represented and handled in the JSON representation. Chapter
 123 10, section 10.2.7 in the XACML 3.0 specification as well as section A.2 list the data-types that are
 124 defined in XACML. These are listed in the table below in section 3.3.1. It lists the shorthand value that
 125 MAY be used when creating a XACML attribute in the JSON representation.

126 **3.3.1 Supported Data Types**

127 The full XACML data type URI can also be used in JSON as the JSON shorthand type codes are a
 128 convenience, not a replacement.

129 It is also possible to omit the JSON "DataType" *member* for certain XACML data types when it can
 130 safely be inferred from the value of the attribute as shown in Table 1.

131 *Table 1. JSON shorthand and rules of inference for XACML data types.*

XACML data type identifier	JSON shorthand type code	Mapping / Inference Rule
http://www.w3.org/2001/XMLSchema#string	string	JSON string
http://www.w3.org/2001/XMLSchema#boolean	boolean	JSON boolean
http://www.w3.org/2001/XMLSchema#integer	integer	JSON number with no fractional portion and within the integer range defined by the XML schema in [XMLDatatypes] .
http://www.w3.org/2001/XMLSchema#double	double	JSON number with fractional portion or out of integer range as defined in [XMLDatatypes] .
http://www.w3.org/2001/XMLSchema#time	time	None – inference must fail.
http://www.w3.org/2001/XMLSchema#date	date	None – inference must fail.
http://www.w3.org/2001/XMLSchema#dateTime	dateTime	None – inference must fail.
http://www.w3.org/2001/XMLSchema#dayTimeDuration	dayTimeDuration	None – inference must fail.
http://www.w3.org/2001/XMLSchema#yearMonthDuration	yearMonthDuration	None – inference must fail.
http://www.w3.org/2001/XMLSchema#anyURI	anyURI	None – inference must fail.
http://www.w3.org/2001/XMLSchema#hexBinary	hexBinary	None – inference must fail.
http://www.w3.org/2001/XMLSchema#base64Binary	base64Binary	None – inference must fail.
urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name	rfc822Name	None – inference must fail.
urn:oasis:names:tc:xacml:1.0:data-type:x500Name	x500Name	None – inference must fail.
urn:oasis:names:tc:xacml:2.0:data-type:ipAddress	ipAddress	None – inference must fail.
urn:oasis:names:tc:xacml:2.0:data-type:dnsName	dnsName	None – inference must fail.
urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression	xpathExpression	None – inference

		must fail
--	--	-----------

132 For all of the XACML data types that cannot be inferred from the value, the following MUST be observed:

- 133 • The JSON "DataType" **member** MUST be specified and the attribute value expressed in the
- 134 XACML string representation of the value.
- 135 • Implementation-specific (e.g. Javascript) code may choose to parse the XACML string values into
- 136 internal numeric representations for internal use, such as for dateTime or duration
- 137 (dayTimeDuration, yearMonthDuration) values, but the JSON transport representation must
- 138 always express the attribute value in the serialized XACML string representation of the XACML
- 139 data type.

140 3.3.2 Multiple values

141 In the case of an **array** of two or more values, and if the "DataType" **member** is not specified, it may

142 not be possible to infer the "DataType" until all the values have been inspected.

143 Inference for multiple values works according to the inference rules as set in section 3.3.1. If a given data

144 type cannot be inferred and there is no "DataType" **member** specified then the **array** of values will be

145 considered as an **array** of string.

146 If an **array** of values contains integers and doubles only (excluding non-numerical values), then the

147 inference will make the **array** an **array** of double.

148 Any other combination of values will make the inference fail and the **array** will be considered as an **array**

149 of string.

150 3.3.3 The xpathExpression Datatype

151 Values of the xpathExpression data-type are represented as JSON **objects**. Each such **object** contains

152 the following **members**:

153 *Table 2 - Members of the XPathExpression Datatype*

Member name	Type	Mandatory/Optional	Default value
"XPathCategory"	A string containing a URI or the shorthand notation defined in section 4.2.2.1	Mandatory	None
"Namespaces"	An array of NamespaceDeclaration objects	Optional	None
"XPath"	String	Mandatory	None

154 The "XPath" **member** contains the XPath expression[**XPATH**] from the XACML value. The

155 "Namespaces" **member** contains namespace declarations for interpreting qualified names

156 [**NAMESPACES**] in the XPath expression.

157 A NamespaceDeclaration **object** contains the following **members**:

158 *Table 3 - Members of the NamespaceDeclaration Datatype*

Member name	Type	Mandatory/Optional	Default value
"Prefix"	String	Optional	None
"Namespace"	A string containing a URI	Mandatory	None

159 Each NamespaceDeclaration **object** describes a single XML namespace declaration [NAMESPACES].
160 The "Prefix" **member** contains the namespace prefix and the "Namespace" **member** contains the
161 namespace name. In the case of a namespace declaration for the default namespace the "Prefix"
162 **member** SHALL be absent.
163 The "Namespaces" **array** MUST contain a NamespaceDeclaration **object** for each of the namespace
164 prefixes used by the XPath expression. The "Namespaces" **array** MAY contain additional
165 NamespaceDeclaration **objects** for namespace prefixes that are not used by the XPath expression.
166 There SHALL NOT be more than one NamespaceDeclaration **object** for the same namespace prefix.

167 3.3.3.1 Example

168 {Non-normative}

169 This example shows the XML representation of an XACML attribute with a value of the xpathExpression
170 data-type and its corresponding representation in JSON.

171 As XML:

```
172 <Attribute xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"  
173   AttributeId="urn:oasis:names:tc:xacml:3.0:content-selector">  
174   <AttributeValue xmlns:md="urn:example:med:schemas:record"  
175     XPathCategory="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"  
176     DataType="urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression"  
177     >md:record/md:patient/md:patientDoB</AttributeValue>  
178 </Attribute>
```

179 As JSON:

```
180 {  
181   "Attribute": {  
182     "AttributeId": "urn:oasis:names:tc:xacml:3.0:content-selector",  
183     "DataType": "xpathExpression",  
184     "Value": {  
185       "XPathCategory":  
186         "urn:oasis:names:tc:xacml:3.0:attribute-category:resource",  
187       "Namespaces": [ {  
188         "Namespace": "urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"  
189       } , {  
190         "Prefix": "md",  
191         "Namespace": "urn:example:med:schemas:record"  
192       } ],  
193       "XPath": "md:record/md:patient/md:patientDoB"  
194     }  
195   }  
196 }
```

197 3.3.4 Special numeric values

198 The following special numeric values are not supported by the profile. Should the request contain such
199 values, the Policy Decision Point MUST reply with an Indeterminate result and a status value of
200 urn:oasis:names:tc:xacml:1.0:status:syntax-error as defined in Appendix B, section 8 of
201 [XACML30].

202 Additional behavior of the PDP when returning urn:oasis:names:tc:xacml:1.0:status:syntax-
203 error is specified in sections 5.57 and B.8 of [XACML30].

- 204 • IEEE 754-2008 NaN ("NaN")
- 205 • IEEE 754-2008 positive infinity ("INF")
- 206 • IEEE 754-2008 negative infinity ("-INF")
- 207 • IEEE 754-2008 negative zero (-0)

208 **3.4 Example**

209 {Non-normative}

210 The example below illustrates equivalent possible notations:

211 *Table 4 - Equivalent examples*

Representation explicitly stating the data-type	Representation omitting the data-type
<pre>{ "Attribute": [{ "AttributeId": "document-id" "DataType": "integer" "Value" : 123 }] }</pre>	<pre>{ "Attribute": [{ "AttributeId": "document-id" "Value": 123 }] }</pre>

212

213

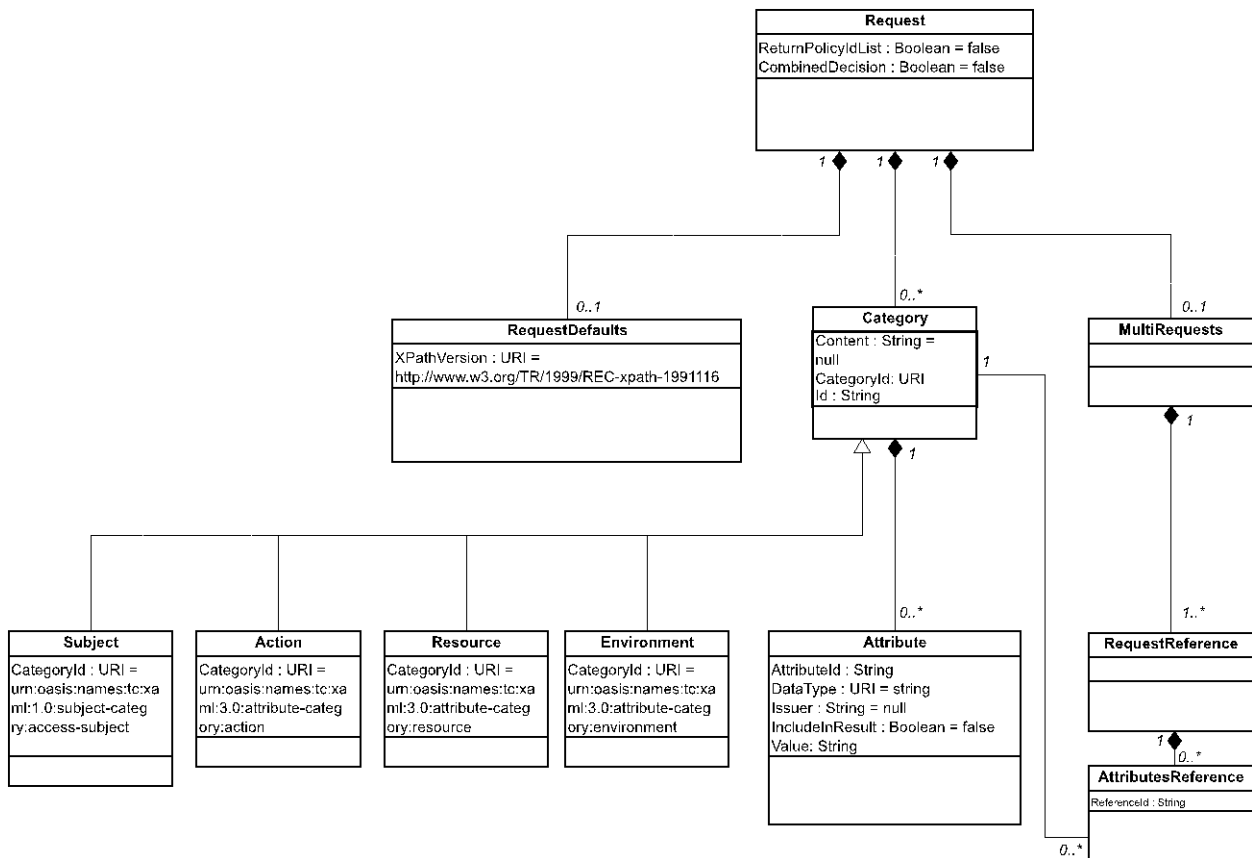
4 The XACML request

4.1 Class Diagram

215 The following class diagram represents the XACML request structure for the JSON representation. It is
216 not a representation of the XACML request as expressed in XML.

217 The key differences are:

- 218 • The <AttributeValue> element in the XML representation does not have a counterpart in the
219 JSON representation. The information it bears in XML is moved to the parent Attribute **object** in
220 the JSON representation.
- 221 • There are 4 new **objects** for attributes belonging to the most commonly used categories.
222



223

224

4.2 Representation of the XACML request in JSON

226 An XACML request is represented as an **object** with a single **member** named "Request". The value of
227 the "Request" **member** is a Request **object**.

4.2.1 The Request object representation

229 The Request **object** contains the following **members**:

230 *Table 5 - Members of the Request object*

Member name	Type	Mandatory/Optional	Default value
"ReturnPolicyIdList"	Boolean	Optional	false
"CombinedDecision"	Boolean	Optional	false
"XPathVersion"	String	Mandatory if the XACML request contains XPath expressions; otherwise, optional.	None
"Category"	An array of Category objects	Optional, but see section 4.2.2.2.	None
"Resource"	An array of Category objects	Optional, but see section 4.2.2.2.	None
"Action"	An array of Category objects	Optional, but see section 4.2.2.2.	None
"Environment"	An array of Category objects	Optional, but see section 4.2.2.2.	None
"AccessSubject"	An array of Category objects	Optional, but see section 4.2.2.2.	None
"RecipientSubject"	An array of Category objects	Optional, but see section 4.2.2.2.	None
"IntermediarySubject"	An array of Category objects	Optional, but see section 4.2.2.2.	None
"CodeBase"	An array of Category objects	Optional, but see section 4.2.2.2.	None
"RequestingMachine"	An array of Category objects	Optional, but see section 4.2.2.2.	None
"MultiRequests"	A MultiRequests object	Optional	None

231 The Category **object** corresponds to the XML <Attributes> element. Just like the <Attributes>
232 element is specific to a given XACML attribute category, the Category **object** in JSON is specific to a
233 given XACML attribute category.

234 The MultiRequests **object** serves to support the Multiple Decision Profile [XACMLMDP].

235 The representation of these **objects** is elicited in the following relevant sections.

236 Note that, in the XACML XML schema, the XML Request element contains a <RequestDefaults>
237 element. To simplify things and since the <RequestDefaults> element contains a single
238 <XPathVersion> element with a single value, the <RequestDefaults> element was flattened into a
239 single **member** called "XPathVersion" as mentioned in the above table.

240 4.2.1.1 Example

241 {Non-normative}

```

242 {
243   "Request": {
244     "Category": [...],
245     "XPathVersion": "http://www.w3.org/TR/1999/REC-xpath-19991116"
246   }

```

247 }

248

249 4.2.2 The Category object representation

250 The Category **object** contains the following **members**:

251 *Table 6 - Members of the Category object*

Member name	Type	Mandatory/Optional	Default value
"CategoryId"	A string containing a XACML category URI or the shorthand notation defined in section 4.2.2.1	Mandatory for a Category object in the "Category" member array ; otherwise, optional. See section 4.2.2.2.	None
"Id"	String	Optional	None
"Content"	String. The value must be escaped or encoded as explained in section 4.2.3.	Optional	None
"Attribute"	An array of Attribute objects	Optional	None

252 The Attribute **object** is defined in section 4.2.4, The Attribute object representation.

253 The Category **object** is the equivalent of the <Attributes> element in the XACML XML representation.

254 The structure and default values for the aforementioned are elicited in the following relevant sections.

255 4.2.2.1 Shorthand notation for standard XACML categories

256 The following table defines a shorthand notation for the standard categories defined in [XACML30].

257 *Table 7 - Shorthand notation for standard XACML categories*

Identifier	Short name
urn:oasis:names:tc:xacml:3.0:attribute-category:resource	Resource
urn:oasis:names:tc:xacml:3.0:attribute-category:action	Action
urn:oasis:names:tc:xacml:3.0:attribute-category:environment	Environment
urn:oasis:names:tc:xacml:1.0:subject-category:access-subject	AccessSubject
urn:oasis:names:tc:xacml:1.0:subject-category:recipient-subject	RecipientSubject
urn:oasis:names:tc:xacml:1.0:subject-category:intermediary-subject	IntermediarySubject
urn:oasis:names:tc:xacml:1.0:subject-category:codebase	Codebase
urn:oasis:names:tc:xacml:1.0:subject-category:requesting-machine	RequestingMachine

258 The shorthand notation MAY be used as described in sections 4.2.2.2 and 4.2.2.

259 4.2.2.2 Default Category objects

260 Category **objects** in the "Category" **member array** relate to various XACML attribute categories as indicated by their individual "CategoryId" **member**, which is a mandatory **member** only for these
261 Category **objects**. To simplify the JSON representation, this profile also defines other **members** (of the
262

263 Request **object**) with an **array** of Category **objects** for the value where the **member** names correspond
264 to the short names defined in section 4.2.2.1. The Category **objects** in these **arrays** assume a default
265 value for their "CategoryId" **member**, i.e., the short name of the containing **member**, so that it need
266 not be explicitly written. The "CategoryId" **member** is optional for these Category **objects**, but if it is
267 provided the value MUST be the same as the short name of the containing **member**.

268 The **members** with the short names have **array** values in order to cater for multiple decision requests as
269 defined in [XACMLMDP].

270 The Request **object** MUST contain at least one Category **object** in one of its **members**.

271 4.2.2.3 Example

272 {Non-normative}

```
273 {  
274   "Request": {  
275     "Category": [{  
276       "CategoryId": "custom-category",  
277       "Attribute": [...]  
278     }, {  
279       "CategoryId": "another-custom-cat",  
280       "Attribute": [...]  
281     }],  
282     "AccessSubject": [{  
283       "Attribute": [...]  
284     }],  
285     "Action": [{  
286       "Attribute": [...]  
287     }, {  
288       "Attribute": [...]  
289     }]  
290   }  
291 }
```

292 4.2.3 The Content object representation

293 There are two possible ways to represent the XML content of a XACML request in the JSON
294 representation: [XML escaping](#) or [Base64 encoding](#). The request parser must determine whether XML
295 escaping or Base 64 encoding is used. There are no **members** in the JSON request to indicate which is
296 used.

297 In both cases, any XML content sent in a JSON request MUST include all namespace definitions needed
298 to parse that content.

299 4.2.3.1 XML Escaping

300 The value of the "Content" **member** is a string which MUST contain an XML payload per the XACML
301 specification.

302 XML content must be escaped before being inserted into the JSON request. JSON dictates double
303 quotes (") be escaped using a backslash (\). This profile therefore follows this behavior.

304 In addition, since the XML content could itself contain backslashes and possibly the sequence "\", it is
305 important to also escape backslashes.

306 4.2.3.2 Base64 Encoding

307 In the case of Base64 encoding, the XML content shall be converted to its Base64 representation as per
308 [BASE64].

309 4.2.3.3 Example

310 {Non-normative}

311 The following is an example using XML escaping as defined in 4.2.3.1.

```
312 {
313   "Request":{
314     "AccessSubject":[{
315       "Content": "<?xml version=\"1.0\"?><catalog><book
316 id=\"bk101\"><author>Gambardella, Matthew</author><title>XML Developer's
317 Guide</title><genre>Computer</genre><price>44.95</price><publish_date>2000-
318 10-01</publish_date><description>An in-depth look at creating applications
319 with XML.</description></book></catalog>"
320     ]}
321   }
322 }
```

323 The following is an example using Base64 encoding as defined in 4.2.3.2.

```
324 {
325   "Request":{
326     "AccessSubject":[{
327       "Content":
328 "PD94bWwgdmVyc2lvbj0iMS4wIj8+DQo8Y2F0YWxvZz48Ym9vayBpZD0iYmsxMDEiPjxhdXRob3I+
329 R2FtYmFyZGVsbGEsIE1hdHRoZXC8L2F1dGhvcj48dG10bGU+WE1MIERldmVsb3BlcidzIEdlawRlP
330 C90aXRszT48Z2VucmU+Q29tcHV0ZXI8L2dlbnJlPjxwcm1jZT40NC45NTwvcHJpY2U+PHB1Ymxpc2
331 hfZGF0ZT4yMDAwLTEwLTAxPC9wdWJsaXNoX2RhdGU+PGRlc2NyaXB0aW9uPkJFuIGluLWRlcHRoIGx
332 vb2sgYXQgY3JlYXRpbmcgYXBwbGljYXRpb25zIHdpdGggWE1MLjwvZGVzY3JpcHRpb24+PC9ib29r
333 PjwvY2F0YWxvZz4="
334     ]}
335   }
336 }
```

337

338 4.2.4 The Attribute object representation

339 The Attribute **object** contains the following **members**:

340 *Table 8 - Members of the Attribute Object*

Member name	Type	Mandatory/ Optional	Default value
"AttributeId"	A string containing a XACML attribute URI	Mandatory	None
"Value"	An array of elements of the same type; either string, boolean, number (which maps to either a XACML integer or double as defined in Supported Data Types) or object .	Mandatory	None

"Issuer"	String	Optional	None
"DataType"	A string containing a XACML data type URI or the shorthand notation defined in section 3.3.1	Optional	The default value will be <code>http://www.w3.org/2001/XMLSchema#string</code> unless the data type can be safely assumed to be otherwise according to the rules set in sections 3.3.1 and 3.3.2.
"IncludeInResult"	Boolean	Optional	false

341 4.2.4.1 Example

342 {Non-normative}

```
343 {
344   "Attribute": [{
345     "AttributeId": "urn:oasis:names:tc:xacml:2.0:subject:role",
346     "Value": ["manager", "administrator"]
347   }]
348 }
```

349 4.2.5 The MultiRequests object representation

350 The MultiRequests **object** is optional in the JSON representation of XACML. Its purpose is to support the
351 Multiple Decision Profile [XACMLMDP]. The MultiRequests **object** contains the following **members**:

352 *Table 9 - Members of the MultiRequests object*

Member name	Type	Mandatory/Optional	Default value
"RequestReference"	An array of one or more RequestReference objects	Mandatory	None

353

354 4.2.6 The RequestReference object representation

355 The RequestReference **object** contains the following **members**:

356 *Table 10 - Members of the RequestReference object*

Member name	Type	Mandatory/Optional	Default value
"ReferenceId"	An array of one or more strings. Each string MUST be the value of a Category object's "Id" member .	Mandatory	None

357

358 4.2.6.1 Non-normative example

```
359 {
360   "MultiRequests": {
361     "RequestReference": [ {
362       "ReferenceId": ["foo1", "bar1"]
363     }, {
364       "ReferenceId": ["foo2", "bar1"]
365     }, {
366       "ReferenceId": ["foo3", "bar1"]

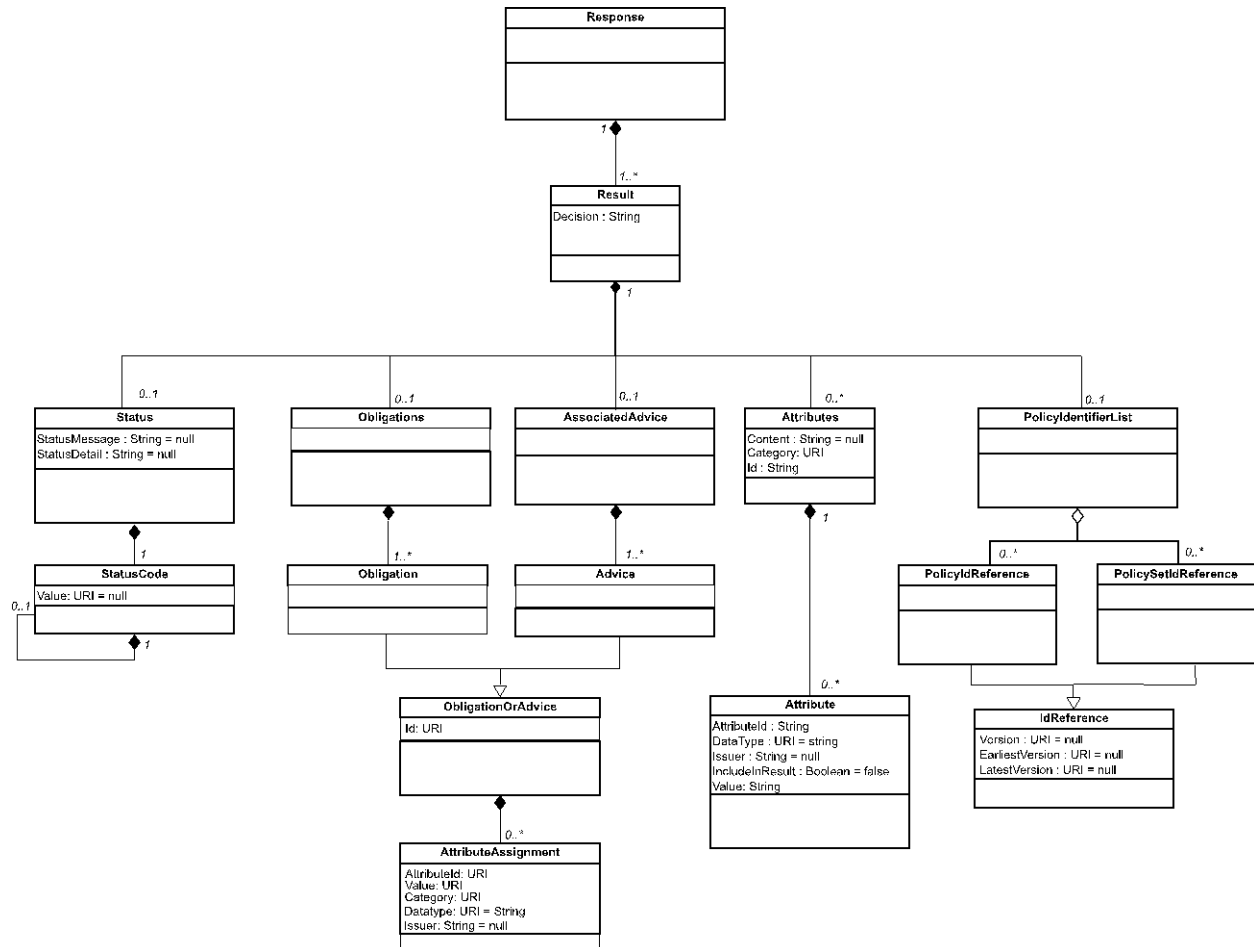
```

```
367     }}]
368   }
369 }
```

370
371
372

5 The XACML response

5.1 Class Diagram



373
374

5.2 Representation of the XACML response in JSON

376 An XACML response is represented as an **object** with a single, mandatory **member** named
377 "Response". The value of the "Response" **member** is an **array** of one or more **Result objects**. There
378 is no **Response object** as such. Instead it is replaced by the value of what would otherwise be its only
379 **member**, an **array** of **Result objects**. This eliminates the nesting of <Result> elements in the
380 <Response> element introduced in XACML's XML schema.

381 The **Result object** representation is detailed hereafter.

5.2.1 The Result object representation

383 The **Result object** contains the following **members**:

384 *Table 11 - Members of the Result object*

Member name	Type	Mandatory/Optional	Default value
"Decision"	String. There are only 4 valid values: "Permit", "Deny", "NotApplicable", and "Indeterminate". The values are case-sensitive.	Mandatory	None
"Status"	A Status object	Optional	None
"Obligations"	An array of ObligationOrAdvice objects	Optional	None
"AssociatedAdvice"	An array of ObligationOrAdvice objects	Optional	None
"Category"	An array of Category objects	Optional	None
"PolicyIdentifierList"	A PolicyIdentifierList object	Optional	None

385 The JSON representation of the <Attributes> element in a XACML response is identical to the
386 representation defined in section 4.2.2 The Category object representation.

387 5.2.2 The Status object representation

388 The Status **object** contains the following **members**:

389 *Table 12 - Members of the Status object*

Member name	Type	Mandatory/Optional	Default value
"StatusMessage"	String	Optional	None
"StatusDetail"	An array of JSON values	Optional	None
"StatusCode"	A StatusCode object	Optional	None

390 The StatusDetail **array** MAY contain arbitrary XML within strings, in which case the XML content must be
391 escaped using the same technique as specified in section 4.2.3, The Content object representation.

392 The StatusDetail **array** MAY contain MissingAttributeDetail **objects**.

393 5.2.3 The MissingAttributeDetail object

394 The MissingAttributeDetail **object** in JSON contains the following **members**:

395 *Table 13 - Members of the MissingAttributeDetail object*

Member name	Type	Mandatory / Optional	Default value
"AttributeId"	A string containing a XACML attribute URI	Mandatory	None

"Value"	An array of elements of the same type; either string, boolean, number (which maps to either a XACML integer or double as defined in Supported Data Types) or object .	Optional	None
"Issuer"	String	Optional	None
"DataType"	A string containing a XACML data type URI or the shorthand notation defined in section 3.3.1	Optional	The default value will be <code>http://www.w3.org/2001/XMLSchema#string</code> unless the data type can be safely assumed to be otherwise according to the rules set in sections 3.3.1 and 3.3.2.
"Category"	A string containing a XACML category URI or the shorthand notation defined in section 4.2.2.1	Mandatory	None

396

397 5.2.4 The StatusCode object representation

398 The StatusCode **object** in JSON contains the following **members**:

399 *Table 14 - Members of the StatusCode object*

Member name	Type	Mandatory /Optional	Default value
"Value"	A string containing a XACML status code URI	Optional	"urn:oasis:names:tc:xacml:1.0:status:ok"
"StatusCode"	A StatusCode object	Optional	None

400 Note that the StatusCode **object** may contain a "StatusCode" **member** – hence potentially creating a
401 recursive nesting of StatusCode **objects**.

402 5.2.4.1 Example

403 {Non-normative}

```

404 {
405   "Response": [{
406     "Decision": "Permit"
407     "Status": {
408       "StatusCode": {
409         "Value": "http://example.com"
410       }
411     }
412   }

```

412 }]
413 }

414 5.2.5 The ObligationOrAdvice object representation

415 The ObligationOrAdvice **object** contains the following **members**:

416 Table 15 - Members of the ObligationOrAdvice object

Member name	Type	Mandatory/Optional	Default value
"Id"	A string containing a XACML obligation or advice URI	Mandatory	None
"AttributeAssignment"	An array of AttributeAssignment objects	Optional	None

417 Note that the ObligationOrAdvice **object** maps to either an <Advice> or an <Obligation> element in
418 the XACML XML representation. While in the XML representation, each element has an attribute called
419 AdviceId or ObligationId respectively, in the JSON representation, the naming has been
420 harmonized to "Id".

421 5.2.6 The AttributeAssignment object representation

422 The AttributeAssignment object contains the following **members**:

423 Table 16 - Members of the AttributeAssignment object

Member name	Type	Mandatory/Optional	Default value
"AttributeId"	A string containing a XACML attribute URI	Mandatory	None
"Value"	Variable	Mandatory	None
"Category"	A string containing a XACML category URI or the shorthand notation defined in section 4.2.2.1	Optional	None
"DataType"	A string containing a XACML data type URI or the shorthand notation defined in section 3.3.1	Optional	The default value depends on the inference rules defined in Supported Data Types.

"Issuer"	String	Optional	None
----------	--------	----------	------

424

425 5.2.7 The PolicyIdentifierList object representation

426 The PolicyIdentifierList **object** contains the following **members**:

427 *Table 17 - Members of the PolicyIdentifierList object*

Member name	Type	Mandatory/Optional	Default value
"PolicyIdReference"	An array of IdReference objects	Optional	None
"PolicySetIdReference"	An array of IdReference objects	Optional	None

428

429 5.2.8 The IdReference object representation

430 The IdReference **object** representation contains the following **members**:

431 *Table 18 - Members of the IdReference object*

Member name	Type	Mandatory/Optional	Default value
"Id"	A string containing a XACML policy or policy set URI. Represents the value stored inside the XACML XML <PolicyIdReference> or <PolicySetIdReference> element.	Mandatory	None
"Version"	String	Optional	None

432

433 6 Transport

434 The XACML request represented in its JSON format MAY be carried from a PEP to a PDP via an HTTP
435 **[HTTP]** request as defined in the REST profile of XACML **[XACMLREST]**.

436 HTTP Headers which may be used are:

- 437 • Content-Type: application/xacml+json
- 438 • Accept: application/xacml+json

439 6.1 Transport Security

440 **{Non-normative}**

441 The use of SSL/TLS **[HTTPS]** is RECOMMENDED to protect requests and responses as they are
442 transferred across the network.

443 7 IANA Registration

444 The following section defines the information required by IANA when applying for a new media type.

445 7.1 Media Type Name

446 application

447 7.2 Subtype Name

448 xacml+json

449 7.3 Required Parameters

450 None.

451 7.4 Optional Parameters

452 version: The version parameter indicates the version of the XACML specification. Its range is the range of
453 published XACML versions. As of this writing that is: 1.0, 1.1, 2.0, and 3.0. These and future version
454 identifiers are of the form x.y, where x and y are decimal numbers with no leading zeros, with x being
455 positive and y being non-negative.

456 7.5 Encoding Considerations

457 Same as for application/xml [RFC4627].

458 7.6 Security Considerations

459 Per their specification, application/xacml+json typed objects do not contain executable content.

460 XACML requests and responses contain information for which integrity and authenticity are important.

461 To counter potential issues, the publisher may use the transport layer's security mechanisms to secure

462 xacml+json typed objects when they are in transit. For instance HTTPS, offers means to ensure the
463 confidentiality and authenticity of the publishing party and the protection of the request/response in
464 transit.

465 7.7 Interoperability Considerations

466 XACML 3.0 uses the urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 XML namespace
467 URI. XACML 2.0 uses the urn:oasis:names:tc:xacml:2.0:policy XML namespace URI.

468 7.8 Applications which use this media type

469 Potentially any application implementing XACML, as well as those applications implementing
470 specifications based on XACML or those applications requesting an authorization decision from a XACML
471 implementation.

472 7.9 Magic number(s)

473 Per [RFC4627], this section is not applicable.

474 7.10 File extension(s)

475 Per [RFC4627], .json.

476 **7.11 Macintosh File Type Code(s)**

477 Text

478 **7.12 Intended Usage**

479 Common

480 8 Examples

481 **{Non-normative}**

482 8.1 Request Example

483 **{Non-normative}**

484 The following is a sample XACML request expressed in JSON.

```
485 {
486   "Request": {
487     "AccessSubject": [ {
488       "Attribute": [ {
489         "AttributeId": "subject-id",
490         "Value": "Andreas"
491       }, {
492         "AttributeId": "location",
493         "Value": "Gamla Stan"
494       } ]
495     },
496     "Action": [ {
497       "Attribute": [ {
498         "AttributeId": "action-id",
499         "Value": "http://example.com/buy",
500         "DataType": "anyURI"
501       } ]
502     },
503     "Resource": [ {
504       "Attribute": [ {
505         "AttributeId": "book-title",
506         "Value": "Learn German in 90 days"
507       }, {
508         "AttributeId": "currency",
509         "Value": "SEK"
510       }, {
511         "AttributeId": "price",
512         "Value": 123.34
513       } ]
514     } ]
515   }
516 }
```

517

518 8.2 Response Example

519 **{Non-normative}**

520 The following is a sample XACML response expressed in JSON.

```
521 {
522   "Response": [ {
523     "Decision": "Permit"
524   } ]
525 }
```

526

527 8.3 Request for Multiple Decisions Example

528 {Non-normative}

529 The following is a sample XACML request for multiple decisions expressed in JSON.

```
530 {
531   "Request":{
532     "AccessSubject":[{
533       "Id":"s1",
534       "Attribute":[{
535         "AttributeId":"com.acme.user.employeeId",
536         "Value":"Alice"
537       }]
538     }],
539     "Resource":[{
540       "Id":"r1",
541       "Attribute":[{
542         "AttributeId":"com.acme.object.objectType",
543         "Value":"record"
544       }],{
545         "AttributeId":"com.acme.record.recordId",
546         "Value":"126",
547         "IncludeInResult":true
548       }]
549     },{
550       "Id":"r2",
551       "Attribute":[{
552         "AttributeId":"com.acme.object.objectType",
553         "Value":"record"
554       }],{
555         "AttributeId":"com.acme.record.recordId",
556         "Value":"125",
557         "IncludeInResult":true
558       }]
559     }],
560     "Action":[{
561       "Id":"a1",
562       "Attribute":[{
563         "AttributeId":"com.acme.action.actionId",
564         "Value":"view",
565         "IncludeInResult":true
566       }]
567     },{
568       "Id":"a2",
569       "Attribute":[{
570         "AttributeId":"com.acme.action.actionId",
571         "Value":"edit",
572         "IncludeInResult":true
573       }]
574     },{
575       "Id":"a3",
576       "Attribute":[{
577         "AttributeId":"com.acme.action.actionId",
578         "Value":"delete",
579         "IncludeInResult":true
580       }]
581     }],
582     "MultiRequests":{
```

```

583     "RequestReference": [{
584         "ReferenceId": [
585             "s1",
586             "a1",
587             "r1"
588         ]
589     }, {
590         "ReferenceId": [
591             "s1",
592             "a2",
593             "r1"
594         ]
595     }]
596 }
597 }
598 }
599

```

600 8.4 Multiple Decisions Response Example

601 {Non-normative}

602 The following is a sample XACML response to a request for multiple decisions expressed in JSON.

```

603 {
604     "Response": [{
605         "Decision": "Deny",
606         "Status": {
607             "StatusCode": {
608                 "Value": "urn:oasis:names:tc:xacml:1.0:status:ok",
609                 "StatusCode": {
610                     "Value": "urn:oasis:names:tc:xacml:1.0:status:ok"
611                 }
612             }
613         },
614         "Category": [{
615             "CategoryId":
616                 "urn:oasis:names:tc:xacml:3.0:attribute-category:resource",
617             "Attribute": [{
618                 "AttributeId": "com.acme.record.recordId",
619                 "Value": "126",
620                 "DataType": "http://www.w3.org/2001/XMLSchema#string"
621             }]
622         }, {
623             "CategoryId":
624                 "urn:oasis:names:tc:xacml:3.0:attribute-category:action",
625             "Attribute": [{
626                 "AttributeId": "com.acme.action.actionId",
627                 "Value": "view",
628                 "DataType": "http://www.w3.org/2001/XMLSchema#string"
629             }]
630         }]
631     }, {
632         "Decision": "Deny",
633         "Status": {
634             "StatusCode": {
635                 "Value": "urn:oasis:names:tc:xacml:1.0:status:ok",
636                 "StatusCode": {
637                     "Value": "urn:oasis:names:tc:xacml:1.0:status:ok"

```

```
638     }
639   },
640   "Category": [{
641     "CategoryId":
642       "urn:oasis:names:tc:xacml:3.0:attribute-category:resource",
643     "Attribute": [{
644       "AttributeId": "com.acme.record.recordId",
645       "Value": "126",
646       "DataType": "http://www.w3.org/2001/XMLSchema#string"
647     }]
648   }, {
649     "CategoryId":
650       "urn:oasis:names:tc:xacml:3.0:attribute-category:action",
651     "Attribute": [{
652       "AttributeId": "com.acme.action.actionId",
653       "Value": "edit",
654       "DataType": "http://www.w3.org/2001/XMLSchema#string"
655     }]
656   }]
657 }
658 }
659
```

660 9 Conformance

661 An implementation may conform to this profile if and only if both the XACML request and the response
662 are correctly encoded into JSON as previously described in sections 3 through 5 and follows the transport
663 requirements as specified in section 6.

664 **Appendix A. Acknowledgments**

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

667 **Participants:**

- 668 • Cyril Dangerville, Thales Group
- 669 • Rich Levinson, Oracle
- 670 • Hal Lockhart, Oracle
- 671 • Bill Parducci,
- 672 • Erik Rissanen, Axiomatics
- 673 • Anil Saldhana, Red Hat
- 674 • Remon Sinnema, EMC
- 675 • Danny Thorpe, Dell
- 676 • Paul Tyson, Bell Helicopters

677

Appendix B. Revision History

Revision	Date	Editor	Changes Made
WD 01	2 Jul 2012	David Brossard	Initial working draft
WD 02	9 Jul 2012	David Brossard	Integrated comments from XACML list. Enhanced the section on data-types. Added a class diagram for clarity. Changed tense to present. Removed overly explicit comparisons with XML representation.
WD 03	19 Jul 2012	David Brossard	Started work on the XACML response
WD 04	20 Aug 2012	David Brossard	Finalized work on the XACML response, added a note on HTTPS. Restructured the document to extract paragraphs common to the Request and Response section.
WD 05	20 Sep 2012	David Brossard	Took in comments from the XACML TC list (technical comments and typographical corrections)
WD 06	29 Oct 2012	David Brossard	Removed the Non-normative section in the appendix. Completed the conformance section. Added non-normative tags where needed. Also added a sample response example. Added the section on IANA registration.
WD07	15 Nov 2012	David Brossard	Removed the XPathExpression from the supported DataTypes. Fixed the examples as per Steven Legg's email. Fixed the XML encoding of XML content as per conversations on the XACML TC list.
WD08	27 Nov 2012	David Brossard	Fixed the Base64 encoding section as per Erik Rissanen's comments
WD09	24 Dec 2012	David Brossard	Addressed comments and fixed errors as per emails sent on the XACML TC list in December.
WD10	4 Feb 2013	David Brossard	Fixed the IANA registration section. Fixed inconsistent DataType spelling. DataType is always the XACML attribute and JSON property name. Data type refers to the English notion. Fixed the status XML content encoding to be consistent with the Request XML encoding technique. Fixed a non-normative section label. Fixed the formatting of JSON property names. Fixed the XACML to JSON data type inference by adding references to the relevant XML data types.

WD11	5 Feb 2013	David Brossard	Fixed the AttributeAssignment section
WD12	10 May 2013	David Brossard	Reinserted a section on the xpathExpression data type. Fixed the PolicyIdReference section (missing value). Fixed the Response example. Simplified the XPathVersion / RequestDefaults Removed unnecessary nesting in Response / Result Renamed Attributes to Category
WD13	14 June 2013	David Brossard	Fixed the final issue re. Category vs. Attributes.
WD14	12 July 2013	David Brossard	Cleaned up the documents and comments.
WD15	02 September 2013	David Brossard	Fixed document based on feedback from Steven Legg: <ul style="list-style-type: none"> • The naming of Attributes vs. Category in section 5.2.2 • Fixed the name of ObligationOrAdvice in section 5.2.6 Also fixed subjective line in introduction based on email xacml-comment from David Webber.
WD16	17 March 2014	David Brossard	<ul style="list-style-type: none"> • Fixed issues with special numerical values: based on input from the XACML TC, special values (NaN, Inf, -0) are now excluded • Rewrote section 3.4.2 and added reference to 3.4.1 • Added a section defining the shorthand notation for standard XACML categories • Added normative reference to XACML 3.0 standard • Added optional category objects for all default categories in XACML 3.0 instead of the 4 most common ones only. • Updated example in 4.2.4.1 • Fixed the Transport section to reference the REST profile. • Fixed broken samples • Added references to IEEE 754-2008 rather than Javascript for the special numerical values • Fixed the Content section to include the namespaces requirement • Fixed the default value for XPathVersion to be in accordance with

			<p>[XACML30].</p> <ul style="list-style-type: none"> Added the MissingAttributeValue object definition.
WD17	14 April 2014	David Brossard	<ul style="list-style-type: none"> Updated the profile title per conversation on the XACML TC list Updated section 3.2.1 on object names in JSON Fixed broken reference to 3.3.1 in 3.3.2 Updated the inference rule for double and integers to remove any doubt as to the potential datatypes Fixed wording in section 4.2.1 (much like vs. just like) Simplified the wording of section 4.2.2.2 Updated the example in section 4.2.2.3 Changed the shorthand name subject to access-subject to be consistent Added the Indeterminate behavior for invalid numerical values Fixed the base 64 encoding example in section 4.2.3.3. Fixed the examples (wrong attribute names, missing parents, missing curly braces) Changed the MS Word quotes into proper quotes
WD18	22 April 2014	David Brossard	<ul style="list-style-type: none"> Changed the shorthand names to use Title Case instead. resource becomes Resource, access-subject becomes AccessSubject, and so on. Updated the XPathCategory so that one can use the category shorthand notation as a valid value instead.
WD19	23 October 2014	David Brossard	<ul style="list-style-type: none"> Introduced formatting changes based on feedback received on xacml-comment Fixed section 6 content-type and accept Fixed the wording on StatusCode Added captions to tables
WD20	26 April 2018	Hal Lockhart	<ul style="list-style-type: none"> Merge in changes to references previously inserted in COS1 Correct Typos noted in Public Review Add Cyril Dangerville to Acknowledgements

WD21	24 May 2018	Steven Legg	<ul style="list-style-type: none"> • Updated the XACML 3.0 reference. • Changed JSON reference to RFC 8259. Dropped the reference to ECMA 262. • Added a glossary for JSON syntax and used the terminology from RFC 8259 throughout the profile. • Moved all the prose descriptions of object and array members into the tables. • Removed all the choices between a single object and an array of objects. Now only arrays are allowed. Updated the examples to reflect the change. • Removed null as an allowed value for the Content member. Disallowed the use of null anywhere. • Clarified that empty arrays are allowed.
WD22	18 July 2018		<ul style="list-style-type: none"> • Added Steven Legg as an editor. • Changed the "Response" member to be mandatory. • Added an example request and response for a request for multiple decisions. • Fixed the cross-references. • Reformatted the examples. • Made numerous formatting corrections.
WD23	15 August 2018		<ul style="list-style-type: none"> • Imported the class diagrams in Enhanced MetaFile format (a vector art format) to improve the document display in HTML. • Removed hyperlinks and used consistent font size in JSON examples.