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.

Related work:
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]

# 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
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.
1.3 Terminology

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

1.4 Normative References


1.5 Non-Normative References


2 Vocabulary

{Non-normative}

XML introduces the notion of elements. The equivalent notion in JSON is an object. XML introduces the notion of attributes. The equivalent notion in JSON is a member.
3 Overview of the translation mechanisms

3.1 Assumed default values

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

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

3.2 Objects

3.2.1 Member names

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

The following XML elements and attributes have been renamed:

- The name of the XACML XML <Attributes> element has been changed in JSON to the member name "Category". It makes more sense to call the parent element that way since it represents an instance of a category from a XACML sense.

- The <AttributeValue> element in the XML representation does not have an object counterpart in the JSON representation. The information it bears in XML is moved to the parent Attribute object in the JSON representation. A "Value" member has been introduced in the Attribute object to bear the information contained in the XML <AttributeValue> element as specified in section 4. The XACML request.

- The AdviceId and the ObligationId attributes of the <Advice> and the <Obligation> XML elements respectively have been renamed to "Id" in JSON.

3.2.2 Object order

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

3.2.3 Object cardinality

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

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

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

3.2.4 Null values

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

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

3.3.1 Supported Data Types

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

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

<table>
<thead>
<tr>
<th>XACML data type identifier</th>
<th>JSON shorthand type code</th>
<th>Mapping / Inference Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.w3.org/2001/XMLSchema#string">http://www.w3.org/2001/XMLSchema#string</a></td>
<td>string</td>
<td>JSON string</td>
</tr>
<tr>
<td><a href="http://www.w3.org/2001/XMLSchema#boolean">http://www.w3.org/2001/XMLSchema#boolean</a></td>
<td>boolean</td>
<td>JSON boolean</td>
</tr>
<tr>
<td><a href="http://www.w3.org/2001/XMLSchema#integer">http://www.w3.org/2001/XMLSchema#integer</a></td>
<td>integer</td>
<td>JSON number with no fractional portion and within the integer range defined by the XML schema in [XMLDatatypes].</td>
</tr>
<tr>
<td><a href="http://www.w3.org/2001/XMLSchema#double">http://www.w3.org/2001/XMLSchema#double</a></td>
<td>double</td>
<td>JSON number with fractional portion or out of integer range as defined in [XMLDatatypes].</td>
</tr>
<tr>
<td><a href="http://www.w3.org/2001/XMLSchema#time">http://www.w3.org/2001/XMLSchema#time</a></td>
<td>time</td>
<td>None – inference must fail.</td>
</tr>
<tr>
<td><a href="http://www.w3.org/2001/XMLSchema#date">http://www.w3.org/2001/XMLSchema#date</a></td>
<td>date</td>
<td>None – inference must fail.</td>
</tr>
<tr>
<td><a href="http://www.w3.org/2001/XMLSchema#dateTime">http://www.w3.org/2001/XMLSchema#dateTime</a></td>
<td>dateTime</td>
<td>None – inference must fail.</td>
</tr>
<tr>
<td><a href="http://www.w3.org/2001/XMLSchema#dayTimeDuration">http://www.w3.org/2001/XMLSchema#dayTimeDuration</a></td>
<td>dayTimeDuration</td>
<td>None – inference must fail.</td>
</tr>
<tr>
<td><a href="http://www.w3.org/2001/XMLSchema#yearMonthDuration">http://www.w3.org/2001/XMLSchema#yearMonthDuration</a></td>
<td>yearMonthDuration</td>
<td>None – inference must fail.</td>
</tr>
<tr>
<td><a href="http://www.w3.org/2001/XMLSchema#anyURI">http://www.w3.org/2001/XMLSchema#anyURI</a></td>
<td>anyURI</td>
<td>None – inference must fail.</td>
</tr>
<tr>
<td><a href="http://www.w3.org/2001/XMLSchema#hexBinary">http://www.w3.org/2001/XMLSchema#hexBinary</a></td>
<td>hexBinary</td>
<td>None – inference must fail.</td>
</tr>
<tr>
<td>urn:oasis:names:tc:xacml:1.0:data-type:base64Binary</td>
<td>base64Binary</td>
<td>None – inference must fail.</td>
</tr>
<tr>
<td>urn:oasis:names:tc:xacml:1.0:data-type:rfc822Name</td>
<td>rfc822Name</td>
<td>None – inference must fail.</td>
</tr>
<tr>
<td>urn:oasis:names:tc:xacml:1.0:data-type:x500Name</td>
<td>x500Name</td>
<td>None – inference must fail.</td>
</tr>
<tr>
<td>urn:oasis:names:tc:xacml:2.0:data-type:dnsName</td>
<td>dnsName</td>
<td>None – inference must fail.</td>
</tr>
</tbody>
</table>
For all of the XACML data types that cannot be inferred from the value, the following MUST be observed:

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

### 3.3.2 Multiple values

In the case of an array of two or more values, and if the "DataType" member is not specified, it may not be possible to infer the "DataType" until all the values have been inspected.

Inference for multiple values works according to the inference rules as set in section 3.3.1. If a given data type cannot be inferred and there is no "DataType" member specified then the array of values will be considered as an array of string.

If an array of values contains integers and doubles only (excluding non-numerical values), then the inference will make the array an array of double.

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

### 3.3.3 The xpathExpression Datatype

Values of the xpathExpression data-type are represented as JSON objects. Each such object contains the following members:

**Table 2 - Members of the XPathExpression Datatype**

<table>
<thead>
<tr>
<th>Member name</th>
<th>Type</th>
<th>Mandatory/Optional</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;XPathCategory&quot;</td>
<td>A string containing a URI or the shorthand notation defined in section 4.2.2.1</td>
<td>Mandatory</td>
<td>None</td>
</tr>
<tr>
<td>&quot;Namespaces&quot;</td>
<td>An array of NamespaceDeclaration objects</td>
<td>Optional</td>
<td>None</td>
</tr>
<tr>
<td>&quot;XPath&quot;</td>
<td>String</td>
<td>Mandatory</td>
<td>None</td>
</tr>
</tbody>
</table>

The "XPath" member contains the XPath expression[XPATH] from the XACML value. The "Namespaces" member contains namespace declarations for interpreting qualified names [NAMESPACE] in the XPath expression.

A NamespaceDeclaration object contains the following members:

**Table 3 - Members of the NamespaceDeclaration Datatype**

<table>
<thead>
<tr>
<th>Member name</th>
<th>Type</th>
<th>Mandatory/Optional</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Prefix&quot;</td>
<td>String</td>
<td>Optional</td>
<td>None</td>
</tr>
<tr>
<td>&quot;Namespace&quot;</td>
<td>A string containing a URI</td>
<td>Mandatory</td>
<td>None</td>
</tr>
</tbody>
</table>
Each NamespaceDeclaration object describes a single XML namespace declaration [NAMESPACES]. The "Prefix" member contains the namespace prefix and the "Namespace" member contains the namespace name. In the case of a namespace declaration for the default namespace the "Prefix" member SHALL be absent. The "Namespaces" array MUST contain a NamespaceDeclaration object for each of the namespace prefixes used by the XPath expression. The "Namespaces" array MAY contain additional NamespaceDeclaration objects for namespace prefixes that are not used by the XPath expression. There SHALL NOT be more than one NamespaceDeclaration object for the same namespace prefix.

3.3.3.1 Example

{Non-normative}

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

As XML:

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

As JSON:

```json
{
    "Attribute": {
        "AttributeId": "urn:oasis:names:tc:xacml:3.0:content-selector",
        "DataType": "xpathExpression",
        "Value": {
            "XPathCategory": 
            "urn:oasis:names:tc:xacml:3.0:attribute-category:resource",
            "Namespaces": [
                {
                    "Namespace": "urn:oasis:names:tc:xacml:3.0:core:schema:wd-17",
                    "Prefix": "md",
                    "Namespace": "urn:example:med:schemas:record"
                },
                
                "XPath": "md:record/md:patient/md:patientDoB"
            ]
        }
    }
}
```

3.3.4 Special numeric values

The following special numeric values are not supported by the profile. Should the request contain such values, the Policy Decision Point MUST reply with an Indeterminate result and a status value of urn:oasis:names:tc:xacml:1.0:status:syntax-error as defined in Appendix B, section 8 of [XACML30]. Additional behavior of the PDP when returning `urn:oasis:names:tc:xacml:1.0:status:syntax-error` is specified in sections 5.57 and 8.8 of [XACML30].

- IEEE 754-2008 NaN ("NaN")
- IEEE 754-2008 positive infinity ("INF")
- IEEE 754-2008 negative infinity ("-INF")
- IEEE 754-2008 negative zero (-0)
### 3.4 Example

{Non-normative}

The example below illustrates equivalent possible notations:

**Table 4 - Equivalent examples**

<table>
<thead>
<tr>
<th>Representation explicitly stating the data-type</th>
<th>Representation omitting the data-type</th>
</tr>
</thead>
<tbody>
<tr>
<td>`{ &quot;Attribute&quot;: [{</td>
<td>`{ &quot;Attribute&quot;: [{</td>
</tr>
<tr>
<td>&quot;AttributeId&quot;: &quot;document-id&quot;</td>
<td>&quot;AttributeId&quot;: &quot;document-id&quot;</td>
</tr>
<tr>
<td>&quot;DataType&quot;: &quot;integer&quot;</td>
<td>&quot;Value&quot;: 123</td>
</tr>
<tr>
<td>&quot;Value&quot;: 123</td>
<td></td>
</tr>
<tr>
<td>} }</td>
<td>}</td>
</tr>
</tbody>
</table>

212
4 The XACML request

4.1 Class Diagram

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

The key differences are:

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

4.2 Representation of the XACML request in JSON

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

4.2.1 The Request object representation

The Request object contains the following members:

Table 5 - Members of the Request object
<table>
<thead>
<tr>
<th>Member name</th>
<th>Type</th>
<th>Mandatory/Optional</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;ReturnPolicyIdList&quot;</td>
<td>Boolean</td>
<td>Optional</td>
<td>false</td>
</tr>
<tr>
<td>&quot;CombinedDecision&quot;</td>
<td>Boolean</td>
<td>Optional</td>
<td>false</td>
</tr>
<tr>
<td>&quot;XPathVersion&quot;</td>
<td>String</td>
<td>Mandatory if the XACML request contains XPath expressions; otherwise, optional.</td>
<td>None</td>
</tr>
<tr>
<td>&quot;Category&quot;</td>
<td>An <strong>array</strong> of <strong>Category objects</strong></td>
<td>Optional, but see section 4.2.2.2.</td>
<td>None</td>
</tr>
<tr>
<td>&quot;Resource&quot;</td>
<td>An <strong>array</strong> of <strong>Category objects</strong></td>
<td>Optional, but see section 4.2.2.2.</td>
<td>None</td>
</tr>
<tr>
<td>&quot;Action&quot;</td>
<td>An <strong>array</strong> of <strong>Category objects</strong></td>
<td>Optional, but see section 4.2.2.2.</td>
<td>None</td>
</tr>
<tr>
<td>&quot;Environment&quot;</td>
<td>An <strong>array</strong> of <strong>Category objects</strong></td>
<td>Optional, but see section 4.2.2.2.</td>
<td>None</td>
</tr>
<tr>
<td>&quot;AccessSubject&quot;</td>
<td>An <strong>array</strong> of <strong>Category objects</strong></td>
<td>Optional, but see section 4.2.2.2.</td>
<td>None</td>
</tr>
<tr>
<td>&quot;RecipientSubject&quot;</td>
<td>An <strong>array</strong> of <strong>Category objects</strong></td>
<td>Optional, but see section 4.2.2.2.</td>
<td>None</td>
</tr>
<tr>
<td>&quot;IntermediarySubject&quot;</td>
<td>An <strong>array</strong> of <strong>Category objects</strong></td>
<td>Optional, but see section 4.2.2.2.</td>
<td>None</td>
</tr>
<tr>
<td>&quot;CodeBase&quot;</td>
<td>An <strong>array</strong> of <strong>Category objects</strong></td>
<td>Optional, but see section 4.2.2.2.</td>
<td>None</td>
</tr>
<tr>
<td>&quot;RequestingMachine&quot;</td>
<td>An <strong>array</strong> of <strong>Category objects</strong></td>
<td>Optional, but see section 4.2.2.2.</td>
<td>None</td>
</tr>
<tr>
<td>&quot;MultiRequests&quot;</td>
<td>A <strong>MultiRequests object</strong></td>
<td>Optional</td>
<td>None</td>
</tr>
</tbody>
</table>

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

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

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

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

### 4.2.1.1 Example

```json
{Non-normative}
{
    "Request":{
        "Category":[...],
        "XPathVersion":"http://www.w3.org/TR/1999/REC-xpath-19991116"
    }
}
```
4.2.2 The Category object representation

The Category object contains the following members:

Table 6 - Members of the Category object

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

The Category object is the equivalent of the <Attributes> element in the XACML XML representation. The structure and default values for the aforementioned are elicited in the following relevant sections.

4.2.2.1 Shorthand notation for standard XACML categories

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

Table 7 - Shorthand notation for standard XACML categories

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Short name</th>
</tr>
</thead>
<tbody>
<tr>
<td>urn:oasis:names:tc:xacml:3.0:attribute-category:resource</td>
<td>Resource</td>
</tr>
<tr>
<td>urn:oasis:names:tc:xacml:3.0:attribute-category:action</td>
<td>Action</td>
</tr>
<tr>
<td>urn:oasis:names:tc:xacml:3.0:attribute-category:environment</td>
<td>Environment</td>
</tr>
<tr>
<td>urn:oasis:names:tc:xacml:1.0:subject-category:access-subject</td>
<td>AccessSubject</td>
</tr>
<tr>
<td>urn:oasis:names:tc:xacml:1.0:subject-category:recipient-subject</td>
<td>RecipientSubject</td>
</tr>
<tr>
<td>urn:oasis:names:tc:xacml:1.0:subject-category:intermediary-subject</td>
<td>IntermediarySubject</td>
</tr>
<tr>
<td>urn:oasis:names:tc:xacml:1.0:subject-category:codebase</td>
<td>Codebase</td>
</tr>
</tbody>
</table>

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

4.2.2.2 Default Category objects

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 Category objects. To simplify the JSON representation, this profile also defines other members (of the
Request object) with an array of Category objects for the value where the member names correspond
to the short names defined in section 4.2.2.1. The Category objects in these arrays assume a default
value for their "CategoryId" member, i.e., the short name of the containing member, so that it need
not be explicitly written. The "CategoryId" member is optional for these Category objects, but if it is
provided the value MUST be the same as the short name of the containing member.

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

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

4.2.2.3 Example

{Non-normative}

"Request":{
    "Category":[
        "CategoryId":"custom-category",
        "Attribute":[]
    ],
    "CategoryId":"another-custom-cat",
    "Attribute":[]
},
"AccessSubject":[
    "Attribute":[]
],
"Action":[
    "Attribute":[]
]
}

4.2.3 The Content object representation

There are two possible ways to represent the XML content of a XACML request in the JSON
representation: XML escaping or Base64 encoding. The request parser must determine whether XML
escaping or Base64 encoding is used. There are no members in the JSON request to indicate which is
used.

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

4.2.3.1 XML Escaping

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

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

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

4.2.3.2 Base64 Encoding

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

{Non-normative}
The following is an example using XML escaping as defined in 4.2.3.1.

```
"Request":{
  "AccessSubject":[
    "Content": "<catalog><book
      id="bk101"><author>Gambardella, Matthew</author><title>XML Developer's
      Guide</title><genre>Computer</genre><price>44.95</price><publish_date>2000-10-01</publish_date><description>An in-depth look at creating applications with XML.</description></book></catalog>"
  ]
}
```

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

```
"Request":{
  "AccessSubject":[
    "Content": "PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0idXNlciIsIyI6Ig0KTS5W Trent Turpin
    (London) for all queries regarding this page."
  ]
}
```

4.2.4 The Attribute object representation

The Attribute object contains the following members:

Table 8 - Members of the Attribute Object

<table>
<thead>
<tr>
<th>Member name</th>
<th>Type</th>
<th>Mandatory/Optional</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;AttributeId&quot;</td>
<td>A string containing a XACML attribute URI</td>
<td>Mandatory</td>
<td>None</td>
</tr>
<tr>
<td>&quot;Value&quot;</td>
<td>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.</td>
<td>Mandatory</td>
<td>None</td>
</tr>
</tbody>
</table>
4.2.4.1 Example

{Non-normative}

{
    "Attribute": [{
        "AttributeId": "urn:oasis:names:tc:xacml:2.0:subject:role",
        "Value": ["manager", "administrator"]
    }]
}

4.2.5 The MultiRequests object representation

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

<table>
<thead>
<tr>
<th>Table 9 - Members of the MultiRequests object</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Member name</strong></td>
</tr>
<tr>
<td>&quot;RequestReference&quot;</td>
</tr>
</tbody>
</table>

4.2.6 The RequestReference object representation

The RequestReference object contains the following members:

<table>
<thead>
<tr>
<th>Table 10 - Members of the RequestReference object</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Member name</strong></td>
</tr>
<tr>
<td>&quot;ReferenceId&quot;</td>
</tr>
</tbody>
</table>

4.2.6.1 Non-normative example

{
    "MultiRequests": {
        "RequestReference": [{
            "ReferenceId": ["foo1", "bar1"]
        }, {
            "ReferenceId": ["foo2", "bar1"]
        }, {
            "ReferenceId": ["foo3", "bar1"]
        }
    }
}
}]}
368  }
369  }
5 The XACML response

5.1 Class Diagram

5.2 Representation of the XACML response in JSON

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

The Result object representation is detailed hereafter.

5.2.1 The Result object representation

The Result object contains the following members:

Table 11 - Members of the Result object
### 5.2.2 The Status object representation

The Status object contains the following members:

Table 12 - Members of the Status object

<table>
<thead>
<tr>
<th>Member name</th>
<th>Type</th>
<th>Mandatory/Optional</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;StatusMessage&quot;</td>
<td>String</td>
<td>Optional</td>
<td>None</td>
</tr>
<tr>
<td>&quot;StatusDetail&quot;</td>
<td>An array of JSON values</td>
<td>Optional</td>
<td>None</td>
</tr>
<tr>
<td>&quot;StatusCode&quot;</td>
<td>A StatusCode object</td>
<td>Optional</td>
<td>None</td>
</tr>
</tbody>
</table>

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

### 5.2.3 The MissingAttributeDetail object

The MissingAttributeDetail object in JSON contains the following members:

Table 13 - Members of the MissingAttributeDetail object

<table>
<thead>
<tr>
<th>Member name</th>
<th>Type</th>
<th>Mandatory / Optional</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;AttributeId&quot;</td>
<td>A string containing a XACML attribute URI</td>
<td>Mandatory</td>
<td>None</td>
</tr>
</tbody>
</table>
"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 http://www.w3.org/2001/XMLSchema#string 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

### 5.2.4 The StatusCode object representation

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

**Table 14 - Members of the StatusCode object**

<table>
<thead>
<tr>
<th>Member name</th>
<th>Type</th>
<th>Mandatory /Optional</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Value&quot;</td>
<td>A string containing a XACML status code URI</td>
<td>Optional</td>
<td>&quot;urn:oasis:names:tc:xacml:1.0:status:ok&quot;</td>
</tr>
<tr>
<td>&quot;StatusCode&quot;</td>
<td>A StatusCode object</td>
<td>Optional</td>
<td>None</td>
</tr>
</tbody>
</table>

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

#### 5.2.4.1 Example

```
{Non-normative}
{
   "Response":{
      "Decision": "Permit"
      "Status":{
         "StatusCode":{
            "Value": "http://example.com"
         }
      }
   }
```

5.2.5 The ObligationOrAdvice object representation

The ObligationOrAdvice object contains the following members:

Table 15 - Members of the ObligationOrAdvice object

<table>
<thead>
<tr>
<th>Member name</th>
<th>Type</th>
<th>Mandatory/Optional</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Id&quot;</td>
<td>A string containing a XACML obligation or advice URI</td>
<td>Mandatory</td>
<td>None</td>
</tr>
<tr>
<td>&quot;AttributeAssignment&quot;</td>
<td>An array of AttributeAssignment objects</td>
<td>Optional</td>
<td>None</td>
</tr>
</tbody>
</table>

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

5.2.6 The AttributeAssignment object representation

The AttributeAssignment object contains the following members:

Table 16 - Members of the AttributeAssignment object

<table>
<thead>
<tr>
<th>Member name</th>
<th>Type</th>
<th>Mandatory/Optional</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;AttributeId&quot;</td>
<td>A string containing a XACML attribute URI</td>
<td>Mandatory</td>
<td>None</td>
</tr>
<tr>
<td>&quot;Value&quot;</td>
<td>Variable</td>
<td>Mandatory</td>
<td>None</td>
</tr>
<tr>
<td>&quot;Category&quot;</td>
<td>A string containing a XACML category URI or the shorthand notation defined in section 4.2.2.1</td>
<td>Optional</td>
<td>None</td>
</tr>
<tr>
<td>&quot;DataType&quot;</td>
<td>A string containing a XACML data type URI or the shorthand notation defined in section 3.3.1</td>
<td>Optional</td>
<td>The default value depends on the inference rules defined in Supported Data Types.</td>
</tr>
</tbody>
</table>
5.2.7 The PolicyIdentifierList object representation

The PolicyIdentifierList object contains the following members:

Table 17 - Members of the PolicyIdentifierList object

<table>
<thead>
<tr>
<th>Member name</th>
<th>Type</th>
<th>Mandatory/Optional</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;PolicyIdReference&quot;</td>
<td>An array of IdReference objects</td>
<td>Optional</td>
<td>None</td>
</tr>
<tr>
<td>&quot;PolicySetIdReference&quot;</td>
<td>An array of IdReference objects</td>
<td>Optional</td>
<td>None</td>
</tr>
</tbody>
</table>

5.2.8 The IdReference object representation

The IdReference object representation contains the following members:

Table 18 - Members of the IdReference object

<table>
<thead>
<tr>
<th>Member name</th>
<th>Type</th>
<th>Mandatory/Optional</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Id&quot;</td>
<td>A string containing a XACML policy or policy set URI. Represents the value stored inside the XACML XML &lt;PolicyIdReference&gt; or &lt;PolicySetIdReference&gt; element.</td>
<td>Mandatory</td>
<td>None</td>
</tr>
<tr>
<td>&quot;Version&quot;</td>
<td>String</td>
<td>Optional</td>
<td>None</td>
</tr>
</tbody>
</table>
6 Transport

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

HTTP Headers which may be used are:

- Content-Type: application/xacml+json
- Accept: application/xacml+json

6.1 Transport Security

{Non-normative}

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

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

7.1 Media Type Name

application

7.2 Subtype Name

xacml+json

7.3 Required Parameters

None.

7.4 Optional Parameters

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

7.5 Encoding Considerations

Same as for application/xml [RFC4627].

7.6 Security Considerations

Per their specification, application/xacml+json typed objects do not contain executable content. XACML requests and responses contain information for which integrity and authenticity are important. To counter potential issues, the publisher may use the transport layer’s security mechanisms to secure xacml+json typed objects when they are in transit. For instance HTTPS, offers means to ensure the confidentiality and authenticity of the publishing party and the protection of the request/response in transit.

7.7 Interoperability Considerations

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

7.8 Applications which use this media type

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

7.9 Magic number(s)

Per [RFC4627], this section is not applicable.

7.10 File extension(s)

Per [RFC4627], .json.
7.11 Macintosh File Type Code(s)

Text

7.12 Intended Usage

Common
8 Examples

{Non-normative}

8.1 Request Example

{Non-normative}

The following is a sample XACML request expressed in JSON.

```
{ "Request":{
  "AccessSubject":[
    "Attribute":{
      "AttributeId":"subject-id",
      "Value":"Andreas"
    },
    "Attribute":{
      "AttributeId":"location",
      "Value":"Gamla Stan"
    }
  ],
  "Action":[
    "Attribute":{
      "AttributeId":"action-id",
      "Value":"http://example.com/buy",
      "DataType":"anyURI"
    }
  ],
  "Resource":[
    "Attribute":{
      "AttributeId":"book-title",
      "Value":"Learn German in 90 days"
    },
    "Attribute":{
      "AttributeId":"currency",
      "Value": "SEK"
    },
    "Attribute":{
      "AttributeId":"price",
      "Value": 123.34
    }
  ]
}
```

8.2 Response Example

{Non-normative}

The following is a sample XACML response expressed in JSON.

```
{ "Response":{
  "Decision":"Permit"
}
```
8.3 Request for Multiple Decisions Example

{Non-normative}

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

```
{
    "Request":{
        "AccessSubject":{
            "Id":"s1",
            "Attribute":{
                "AttributeId":"com.acme.user.employeeId",
                "Value":"Alice"
            }
        },
        "Resource":{
            "Id":"r1",
            "Attribute":{
                "AttributeId":"com.acme.object.objectType",
                "Value":"record",
                "IncludeInResult":true
            },
            "Id":"r2",
            "Attribute":{
                "AttributeId":"com.acme.object.objectType",
                "Value":"record",
                "IncludeInResult":true
            }
        },
        "Action":{
            "Id":"a1",
            "Attribute":{
                "AttributeId":"com.acme.action.actionId",
                "Value":"view",
                "IncludeInResult":true
            }
        },
        "Id":"a2",
        "Attribute":{
            "AttributeId":"com.acme.action.actionId",
            "Value":"edit",
            "IncludeInResult":true
        }
    }
}
```
8.4 Multiple Decisions Response Example

{Non-normative}

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

```json
{
  "Response":{
    "Decision":"Deny",
    "Status":{
      "StatusCode":{
        "Value":"urn:oasis:names:tc:xacml:1.0:status:ok",
        "StatusCode":{
          "Value":"urn:oasis:names:tc:xacml:1.0:status:ok"
        }
      }
    }
  },
  "Category":{
    "CategoryId":
      "urn:oasis:names:tc:xacml:3.0:attribute-category:resource",
    "Attribute":{
      "AttributeId":"com.acme.record.recordId",
      "Value":"126",
      "DataType":"http://www.w3.org/2001/XMLSchema#string"
    }
  },
  "CategoryId":
    "urn:oasis:names:tc:xacml:3.0:attribute-category:action",
  "Attribute":{
    "AttributeId":"com.acme.action.actionId",
    "Value":"view",
    "DataType":"http://www.w3.org/2001/XMLSchema#string"
  }
},
  "Decision":"Deny",
  "Status":{
    "StatusCode":{
      "Value":"urn:oasis:names:tc:xacml:1.0:status:ok",
      "StatusCode":{
        "Value":"urn:oasis:names:tc:xacml:1.0:status:ok"
      }
    }
  }
}
```
"Category":{{
  "CategoryId": "urn:oasis:names:tc:xacml:3.0:attribute-category:resource",
  "Attribute":{{
    "AttributeId": "com.acme.record.recordId",
    "Value": "126",
    "DataType": "http://www.w3.org/2001/XMLSchema#string"
  }}
},
  "CategoryId": "urn:oasis:names:tc:xacml:3.0:attribute-category:action",
  "Attribute":{{
    "AttributeId": "com.acme.action.actionId",
    "Value": "edit",
    "DataType": "http://www.w3.org/2001/XMLSchema#string"
  }}
}}
9 Conformance

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

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

Participants:
- Cyril Dangerville, Thales Group
- Rich Levinson, Oracle
- Hal Lockhart, Oracle
- Bill Parducci,
- Erik Rissanen, Axiomatics
- Anil Saldhana, Red Hat
- Remon Sinnema, EMC
- Danny Thorpe, Dell
- Paul Tyson, Bell Helicopters
# Appendix B. Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Editor</th>
<th>Changes Made</th>
</tr>
</thead>
<tbody>
<tr>
<td>WD 01</td>
<td>2 Jul 2012</td>
<td>David Brossard</td>
<td>Initial working draft</td>
</tr>
<tr>
<td>WD 02</td>
<td>9 Jul 2012</td>
<td>David Brossard</td>
<td>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.</td>
</tr>
<tr>
<td>WD 03</td>
<td>19 Jul 2012</td>
<td>David Brossard</td>
<td>Started work on the XACML response</td>
</tr>
<tr>
<td>WD 04</td>
<td>20 Aug 2012</td>
<td>David Brossard</td>
<td>Finalized work on the XACML response, added a note on HTTPS. Restructured the document to extract paragraphs common to the Request and Response section.</td>
</tr>
<tr>
<td>WD 05</td>
<td>20 Sep 2012</td>
<td>David Brossard</td>
<td>Took in comments from the XACML TC list (technical comments and typographical corrections)</td>
</tr>
<tr>
<td>WD 06</td>
<td>29 Oct 2012</td>
<td>David Brossard</td>
<td>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.</td>
</tr>
<tr>
<td>WD07</td>
<td>15 Nov 2012</td>
<td>David Brossard</td>
<td>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.</td>
</tr>
<tr>
<td>WD08</td>
<td>27 Nov 2012</td>
<td>David Brossard</td>
<td>Fixed the Base64 encoding section as per Erik Rissanen’s comments</td>
</tr>
<tr>
<td>WD09</td>
<td>24 Dec 2012</td>
<td>David Brossard</td>
<td>Addressed comments and fixed errors as per emails sent on the XACML TC list in December.</td>
</tr>
<tr>
<td>WD10</td>
<td>4 Feb 2013</td>
<td>David Brossard</td>
<td>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.</td>
</tr>
<tr>
<td>WD</td>
<td>Date</td>
<td>Author</td>
<td>Description</td>
</tr>
<tr>
<td>----</td>
<td>--------------</td>
<td>-----------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>WD11</td>
<td>5 Feb 2013</td>
<td>David Brossard</td>
<td>Fixed the AttributeAssignment section</td>
</tr>
<tr>
<td>WD12</td>
<td>10 May 2013</td>
<td>David Brossard</td>
<td>Reinserted a section on the xpathExpression data type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fixed the PolicyIdReference section (missing value).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fixed the Response example.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Simplified the XPathVersion / RequestDefaults</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Removed unnecessary nesting in Response / Result</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Renamed Attributes to Category</td>
</tr>
<tr>
<td>WD13</td>
<td>14 June 2013</td>
<td>David Brossard</td>
<td>Fixed the final issue re. Category vs. Attributes.</td>
</tr>
<tr>
<td>WD14</td>
<td>12 July 2013</td>
<td>David Brossard</td>
<td>Cleaned up the documents and comments.</td>
</tr>
<tr>
<td>WD15</td>
<td>02 September 2013</td>
<td>David Brossard</td>
<td>Fixed document based on feedback from Steven Legg:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• The naming of Attributes vs. Category in section 5.2.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Fixed the name of ObligationOrAdvice in section 5.2.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Also fixed subjective line in introduction based on email xacml-comment from David Webber.</td>
</tr>
<tr>
<td>WD16</td>
<td>17 March 2014</td>
<td>David Brossard</td>
<td>Fixed issues with special numerical values: based on input from the XACML TC, special values (NaN, Inf, -0) are now excluded</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Rewrote section 3.4.2 and added reference to 3.4.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Added a section defining the shorthand notation for standard XACML categories</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Added normative reference to XACML 3.0 standard</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Added optional category objects for all default categories in XACML 3.0 instead of the 4 most common ones only.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Updated example in 4.2.4.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Fixed the Transport section to reference the REST profile.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Fixed broken samples</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Added references to IEEE 754-2008 rather than Javascript for the special numerical values</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Fixed the Content section to include the namespaces requirement</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Fixed the default value for XPathVersion to be in accordance with</td>
</tr>
<tr>
<td>WD17</td>
<td>14 April 2014</td>
<td>David Brossard</td>
<td>• Added the MissingAttributeValue object definition.</td>
</tr>
<tr>
<td>------</td>
<td>--------------</td>
<td>----------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Updated the profile title per conversation on the XACML TC list</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Updated section 3.2.1 on object names in JSON</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Fixed broken reference to 3.3.1 in 3.3.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Updated the inference rule for double and integers to remove any doubt as to the potential datatypes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Fixed wording in section 4.2.1 (much like vs. just like)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Simplified the wording of section 4.2.2.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Updated the example in section 4.2.2.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Changed the shorthand name subject to access-subject to be consistent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Added the Indeterminate behavior for invalid numerical values</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Fixed the base 64 encoding example in section 4.2.3.3.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Fixed the examples (wrong attribute names, missing parents, missing curly braces)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Changed the MS Word quotes into proper quotes</td>
</tr>
<tr>
<td>WD18</td>
<td>22 April 2014</td>
<td>David Brossard</td>
<td>• Changed the shorthand names to use Title Case instead. resource becomes Resource, access-subject becomes AccessSubject, and so on.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Updated the XPathCategory so that one can use the category shorthand notation as a valid value instead.</td>
</tr>
<tr>
<td>WD19</td>
<td>23 October 2014</td>
<td>David Brossard</td>
<td>• Introduced formatting changes based on feedback received on xacml-comment</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Fixed section 6 content-type and accept</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Fixed the wording on StatusCode</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Added captions to tables</td>
</tr>
<tr>
<td>WD20</td>
<td>26 April 2018</td>
<td>Hal Lockhart</td>
<td>• Merge in changes to references previously inserted in COS1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Correct Typos noted in Public Review</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Add Cyril Dangerville to Acknowledgements</td>
</tr>
</tbody>
</table>
| WD21 | 24 May 2018 | Steven Legg | • 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 | | • 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 | | • 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. |