OGC® DOCUMENT: 22-050

External identifier of this OGC® document: http://www.opengis.net/doc/{doc-type}/{standard}/{m.n}

Open Geospatial Consortium

OGC GEOSPATIAL EXENSIBLE ACCESS CONTROL MARKUP LANGUAGE (GEOXACML) 3.0 JSON PROFILE V1.0

STANDARD
Implementation

DRAFT

Version: 0.6
Submission Date: 2023-02-06
Approval Date: 2029-03-30
Publication Date: 2029-03-30
Editor: Andreas Matheus

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF RECOMMENDATIONS

I ABSTRACT

This OGC Geospatial eXensible Access Control Markup Language (GeoXACML) 3.0 JSON Profile v1.0 (GeoXACML 3.0 JSON Profile) defines an extension to the JSON Profile of XACML 3.0 Version 1.1 for supporting GeoXACML Authorization Decision Requests and Authorization Decision encoded in JSON. This ensures an easy uptake in environments where JSON is the preferred encoding.

For supporting `Geometry` as defined by GeoXACML 3.0 Core conformance class, this profile extends the `Attribute` DataType definition from JSON Profile of XACML 3.0 Version 1.1 with the geometry data-type `urn:ogc:def:dataType:geoxacml:3.0:geometry`

The GeoXACML 3.0 JSON Profile Standard supports the `Attribute` value to use Well-Known-Text, Well-Known-Binary (hex-encoding) or GeoJSON as an encoding alternative for the geometry data-type defined in GeoXACML 3.0.

To support the use of the GeoXACML 3.0 specific attributes `srid`, `precision` and `allowTransformation`, this profile extends the default JSON schema definition from JSON Profile of XACML 3.0 Version 1.1 accordingly.

II KEYWORDS

The following are keywords to be used by search engines and document catalogues.

ogcdoc, OGC document, XACML, GeoXACML, JSON, Profile

III SECURITY CONSIDERATIONS

The GeoXACML 3.0 JSON Profile does not introduce specific attack vectors to those typically considered when parsing and validating JSON encoded service request / response. When the origin of the data is unknown, great care should be applied when parsing the data. The EdwardHuang post lists five common denial of service attack scenarios when parsing JSON data.

Mitigating attacks for the GeoXACML 3.0 JSON Profile are not different from those that can be found on the Internet. To mitigate parsing attacks, the use of a tight JSON schema in combination with meaningful limits for payload size, object size, nesting depth, etc. should be considered.

The use of JWS (JSON Web Signature) may be used in addition to this profile to establish trust in the origin of the request / response.

IV SUBMITTING ORGANIZATIONS

The following organizations submitted this Document to the Open Geospatial Consortium (OGC):

- Secure Dimensions GmbH
- Natural Resources Canada (NRCAN)
- Defense Information Systems Agency (DISA)

1

SCOPE

1 SCOPE

This Standard defines an extension to JSON Profile of XACML 3.0 Version 1.1 for supporting the encoding of a GeoXACML Authorization Decision Request and Authorization Decision in JSON.

This profile defines the encoding options for a `Geometry` instance as defined in GeoXACML 3.0 based on `Well-Known-Text`, `Well-Known-Binary` and `GeoJSON`.

2

CONFORMANCE

2 CONFORMANCE

This Standard defines two Conformance Classes.

Conformance with this Standard shall be checked using all the relevant tests specified in Annex A (normative) of this document. The framework, concepts, and methodology for testing, and the criteria to be achieved to claim conformance are specified in the OGC Compliance Testing Policies and Procedures and the OGC Compliance Testing web site.

In order to conform to this OGC® Standard, a software implementation shall pass all tests defined in Annex A.

All requirements-classes and conformance-classes described in this document are owned by the standard(s) identified.

2.1. Conformance Class Data Model

CONFORMANCE CLASS 1	
IDENTIFIER	`/conf/data-model`
REQUIREMENTS CLASS	Requirements class 1: `/req-class/data-model`
TARGET TYPE	Implementation Specification
CONFORMANCE TESTS	Conformance test A.1: `/conf/data-model/definition` Conformance test A.2: `/conf/data-model/crs-axis-order` Conformance test A.3: `/conf/data-model/media-type` Conformance test A.4: `/conf/data-model/wkt` Conformance test A.5: `/conf/data-model/wkb` Conformance test A.6: `/conf/data-model/geojson`

2.2. Conformance Class Core

CONFORMANCE CLASS 2	
IDENTIFIER	`/conf/core`

CONFORMANCE CLASS 2	
REQUIREMENTS CLASS	Requirements class 2: `/req-class/core`
PREREQUISITE	Requirements class 1: `/req-class/data-model`
TARGET TYPE	Implementation
CONFORMANCE TESTS	Conformance test A.7: `/conf/core/media-type-impl` Conformance test A.8: `/conf/core/json-schema-impl` Conformance test A.9: `/conf/core/wkt-impl` Conformance test A.10: `/conf/core/wkb-impl` Conformance test A.11: `/conf/core/geojson-impl` Conformance test A.12: `/conf/core/allow-transformation-impl` Conformance test A.13: `/conf/core/crs-impl` Conformance test A.14: `/conf/core/axis-order-impl-1` Conformance test A.15: `/conf/core/axis-order-impl-2` Conformance test A.16: `/conf/core/error-reporting-impl`

3

NORMATIVE REFERENCES

3 NORMATIVE REFERENCES

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

OGC Geospatial eXtensible Markup Language (GeoXACML) 3.0, Draft OGC 22-049, OGC, 2023

eXtensible Access Control Markup Language (XACML) Version 3.0, OASIS, 2013, http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html

JSON Profile of XACML 3.0 Version 1.1, OASIS, 2019, http://docs.oasis-open.org/xacml/xacml-json-http/v1.1/xacml-json-http-v1.1.html

The GeoJSON Format, IETF, 2016, https://www.rfc-editor.org/rfc/rfc7946

Geographic information — Simple features access — Part 1: Common architecture, ISO, 2004, https://portal.opengeospatial.org/files/?artifact_id=25355

4

TERMS AND DEFINITIONS

4 TERMS AND DEFINITIONS

No terms and definitions are listed in this document.

All terms and definition can be found in GeoXACML 3.0.

The following JSON property names are defined for the `Attribute` element according to the name convention of JSON Profile of XACML 3.0 Version 1.1:

- SRID: Element property as defined in GeoXACML 3.0
- Precision: Element property as defined in GeoXACML 3.0
- AllowTransformation: Element property as defined in GeoXACML 3.0

5

CONVENTIONS

5 CONVENTIONS

This section provides details and examples for any conventions used in the document. Examples of conventions are symbols, abbreviations, use of XML schema, or special notes regarding how to read the document.

5.1. Identifiers

The normative provisions in this standard are denoted by the URI

`http://www.opengis.net/spec/geoxacml/3.0/json-profile/1.0`

All requirements and conformance tests that appear in this document are denoted by partial URIs which are relative to this base.

6

INTRODUCTION TO GEOXACML 3.0 JSON PROFILE V1.0

6 INTRODUCTION TO GEOXACML 3.0 JSON PROFILE V1.0

The XACML Version 3.0 Standard defines an XML-based policy language and the structure for the encoding of Authorization Decision Request (ADR) and Authorization Decision (AD) in XML. A more lightweight and compact encoding in JSON is defined in JSON Profile of XACML 3.0 Version 1.1.

The GeoXACML 3.0 Standard supports the XML encoding of ADR and AD. This is because that feature is inherited from XACML Version 3.0. The direct use of JSON Profile of XACML 3.0 Version 1.1 is not possible, because it does not support the `Geometry` data-type as defined in OGC GeoXACML 3.0.

Also, declaring additional properties for the `Attribute` element is not possible which downgrades the geometry encodings to use the GeoXACML 3.0 default CRS `urn:ogc:def:crs:OGC::CRS84`.

To support the same expressiveness for encoding the ADR and AD in JSON, as it is possible for XML, this Profile defines the new media-type `application/geoxacml+json` which supports the GeoXACML 3.0 data-type `urn:ogc:def:dataType:geoxacml:3.0:geometry` and additional properties for the `Attribute` element.

7

GEOXACML 3.0 JSON PROFILE V1.0 REQUIREMENTS

7 GEOXACML 3.0 JSON PROFILE V1.0 REQUIREMENTS

This section defines the requirements to extend the JSON Profile of XACML 3.0 Version 1.1 for encoding ADR and AD in JSON. This profile is typically used with a GeoXACML 3.0 implementation as a service, hence supports the `API` conformance class.

7.1. Requirement Class Data Model (abstract)

The standardization target for this requirements class is Implementation Specification.

The **Data Model** Requirements Class defines additional properties for the `<Attribute>` element as defined in JSON Profile of XACML 3.0 Version 1.1.

REQUIREMENTS CLASS 1: GEOXACML 3.0 JSON PROFILE V1.0 DATA MODEL	
IDENTIFIER	`/req-class/data-model`
OBLIGATION	requirement
TARGET TYPE	Implementation Specification
CONFORMANCE CLASS	Conformance class 1: `/conf/data-model`
PREREQUISITES	GeoXACML 3.0 JSON Profile of XACML 3.0 Version 1.1
NORMATIVE STATEMENTS	Requirement 1: `/req/data-type` Requirement 2: `/req/crs` Requirement 3: `/req/axis-order` Requirement 4: `/req/media-type` Requirement 5: `/req/geojson` Requirement 6: `/req/wkt` Requirement 7: `/req/wkb` Requirement 8: `/req/srid` Requirement 9: `/req/precision` Requirement 10: `/req/allow-transformation`

REQUIREMENT 1	
IDENTIFIER	`/req/data-type`
INCLUDED IN	Requirements class 1: `/req-class/data-model`
STATEMENT	This profile extends the `DataType` definition from JSON Profile of XACML 3.0 Version 1.1, §3.3.1 with the GeoXACML 3 data-type `urn:ogc:def:dataType:geoxacml:3.0:geometry`.

REQUIREMENT 2	
IDENTIFIER	`/req/crs`
INCLUDED IN	Requirements class 1: `/req-class/data-model`
STATEMENT	This profile inherits the default Coordinate Reference System (CRS) identifier (`urn:ogc:def:crs:OGC::CRS84`) as specified in [geoxacml3].

REQUIREMENT 3	
IDENTIFIER	`/req/axis-order`
INCLUDED IN	Requirements class 1: `/req-class/data-model`
STATEMENT	This profile inherits the axis order (longitude/latitude) as specified in [geoxacml3].

REQUIREMENT 4	
IDENTIFIER	`/req/media-type`
INCLUDED IN	Requirements class 1: `/req-class/data-model`
STATEMENT	This profile defines the media-type `application/geoxacml+json`.

REQUIREMENT 5	
IDENTIFIER	`/req/geojson`

REQUIREMENT 5	
INCLUDED IN	Requirements class 1: `/req-class/data-model`
STATEMENT	This profile defines the value for the `Attribute` element defined in JSON Profile of XACML 3.0 Version 1.1, §3.3.1 may use the GeoJSON `Geometry` object as defined in The GeoSON Format, §3.1 to represent a geometry value.

REQUIREMENT 6	
IDENTIFIER	`/req/wkt`
INCLUDED IN	Requirements class 1: `/req-class/data-model`
STATEMENT	This profile defines the value for the `Attribute` element defined in JSON Profile of XACML 3.0 Version 1.1, §3.3.1 may use the WKT `Geometry` encoding as defined in OGC Simple Features to represent a geometry value.

REQUIREMENT 7	
IDENTIFIER	`/req/wkb`
INCLUDED IN	Requirements class 1: `/req-class/data-model`
STATEMENT	This profile defines the value for the `Attribute` element defined in JSON Profile of XACML 3.0 Version 1.1, §3.3.1 may use the WKB `Geometry` encoding as hex-string as defined in OGC Simple Features to represent a geometry value.

REQUIREMENT 8	
IDENTIFIER	`/req/srid`
INCLUDED IN	Requirements class 1: `/req-class/data-model`
STATEMENT	This profile defines the additional element `SRID` for the `Attribute` element defined in JSON Profile of XACML 3.0 Version 1.1, §3.3.1.

REQUIREMENT 9	
IDENTIFIER	`/req/precision`

REQUIREMENT 9	
INCLUDED IN	Requirements class 1: `/req-class/data-model`
STATEMENT	This profile defines the additional element `Precision` for the `Attribute` element defined in JSON Profile of XACML 3.0 Version 1.1, §3.3.1.

REQUIREMENT 10	
IDENTIFIER	`/req/allow-transformation`
INCLUDED IN	Requirements class 1: `/req-class/data-model`
STATEMENT	This profile defines the additional element `AllowTransformation` for the `Attribute` element defined in JSON Profile of XACML 3.0 Version 1.1, §3.3.1.

7.2. Requirements Class GeoXACML 3.0 JSON Profile v1.0 Core

The standardization target for this requirements class is Implementation.

REQUIREMENTS CLASS 2: GEOXACML 3.0 JSON PROFILE V1.0 CORE	
IDENTIFIER	`/req-class/core`
OBLIGATION	requirement
TARGET TYPE	Implementation
CONFORMANCE CLASS	Conformance class 2: `/conf/core`
PREREQUISITE	Requirements class 1: `/req-class/data-model`
NORMATIVE STATEMENTS	Requirement 11: `/req/wkt-impl` Requirement 12: `/req/wkb-impl` Requirement 13: `/req/geojson-impl` Requirement 14: `/req/crs-impl` Requirement 15: `/req/axis-order-impl` Requirement 16: `/req/media-type-impl` Requirement 17: `/req/srid-impl`

REQUIREMENTS CLASS 2: GEOXACML 3.0 JSON PROFILE V1.0 CORE	
	Requirement 18: `/req/precision-impl` Requirement 19: `/req/allow-transformation-impl` Requirement 20: `/req/error-reporting-impl` Requirement 21: `/req/json-schema-impl`

REQUIREMENT 11	
IDENTIFIER	`/req/wkt-impl`
INCLUDED IN	Requirements class 2: `/req-class/core`
STATEMENT	An implementation SHALL support the Well Known Text (WKT) encoding as defined in OGC Simple Features for expressing geometry value in the `Attribute` value.

REQUIREMENT 12	
IDENTIFIER	`/req/wkb-impl`
INCLUDED IN	Requirements class 2: `/req-class/core`
STATEMENT	An implementation SHALL support the Well Known Binary (WKB) hexstring encoding as defined in OGC Simple Features for expressing geometry value in the `Attribute` value.

REQUIREMENT 13	
IDENTIFIER	`/req/geojson-impl`
INCLUDED IN	Requirements class 2: `/req-class/core`
STATEMENT	An implementation SHALL support the GeoJSON encoding as defined in The GeoSON Format for expressing geometry value in the `Attribute` value.

REQUIREMENT 14	
IDENTIFIER	`/req/crs-impl`
INCLUDED IN	Requirements class 2: `/req-class/core`

REQUIREMENT 14	
STATEMENT	An implementation SHALL use the default CRS for calculating the geometry coordinate values unless specified otherwise using the element `SRID` for the `Attribute` element defined in Clause 7.1, Requirement 8: `/req/srid`.

REQUIREMENT 15	
IDENTIFIER	`/req/axis-order-impl`
INCLUDED IN	Requirements class 2: `/req-class/core`
STATEMENT	An implementation SHALL use the default axis order for serialization of the geometry coordinate values unless specified otherwise, indicated by using the element `SRID` for the `Attribute` element defined in Clause 7.1, Requirement 8: `/req/srid`.

REQUIREMENT 16	
IDENTIFIER	`/req/media-type-impl`
INCLUDED IN	Requirements class 2: `/req-class/core`
STATEMENT	An implementation SHALL support the media-type `application/geoxacml+json`. The media type `application/geoxacml+json` SHALL be used in association with the HTTP `Content-Type` and `Accept` headers when sending Authorization Decision Request and asking to receive a GeoXACML 3.0 compliant Authorization Decision via HTTP transport.

REQUIREMENT 17	
IDENTIFIER	`/req/srid-impl`
INCLUDED IN	Requirements class 2: `/req-class/core`
STATEMENT	An implementation SHALL support the additional element `SRID` for the `Attribute` element defined in Clause 7.1, Requirement 8: `/req/srid`. An implementation SHALL set the `SRID` value to the integer that identifies the CRS which was used to calculate the geometry coordinate values.

REQUIREMENT 18	
IDENTIFIER	`/req/precision-impl`

REQUIREMENT 18	
INCLUDED IN	Requirements class 2: `/req-class/core`
STATEMENT	This profile defines the additional element `Precision` for the `Attribute` element defined in Clause 7.1, Requirement 9: `/req/precision`.

REQUIREMENT 19	
IDENTIFIER	`/req/allow-transformation-impl`
INCLUDED IN	Requirements class 2: `/req-class/core`
STATEMENT	This profile defines the additional element `AllowTransformation` for the `Attribute` element defined in Clause 7.1, Requirement 10: `/req/allow-transformation`.

REQUIREMENT 20	
IDENTIFIER	`/req/error-reporting-impl`
INCLUDED IN	Requirements class 2: `/req-class/core`
STATEMENT	An implementation SHALL support the `MissingAttributeDetail` element as part of the `StatusCode` element when reporting an `Indeterminate` decision with value code `urn:ogc:def:identifier:geoxacml:3.0:crs-error` as defined in GeoXACML 3.0.

REQUIREMENT 21	
IDENTIFIER	`/req/json-schema-impl`
INCLUDED IN	Requirements class 2: `/req-class/core`
STATEMENT	An implementation SHALL support the JSON request schema as defined below for the ADR structure when using the `application/geoxacml+json` media type.

```
{
  "$schema": "http://json-schema.org/draft-06/schema",
  "$id": "Request-with-geometry.schema.json",
  "title": "JSON schema of Request object defined in GeoXACML 3.0 JSON Profile
v1.0",
  "definitions": {
    "RequestReferenceType": {
      "type": "object",
      "properties": {
        "ReferenceId": {
          "type": "array",
          "items": {
            "description": "Each item is a Category/Id",
            "type": "string"
          },
          "minItems": 1
        }
      },
      "required": [
        "ReferenceId"
      ],
      "additionalProperties": false
    },
    "MultiRequestsType": {
      "type": "object",
      "properties": {
        "RequestReference": {
          "type": "array",
          "items": {
            "$ref": "#/definitions/RequestReferenceType"
          },
          "minItems": 1
        }
      },
      "required": [
        "RequestReference"
      ],
      "additionalProperties": false
    },
    "RequestType": {
      "type": "object",
      "properties": {
        "ReturnPolicyIdList": {
          "type": "boolean"
        },
        "CombinedDecision": {
          "type": "boolean"
        },
        "XPathVersion": {
          "type": "string"
        },
        "Category": {
          "type": "array",
          "items": {
            "$ref": "common-std-with-geometry.schema.json#/definitions/Attribut
eCategoryType"
          },
          "minItems": 1
        },
        "MultiRequests": {
          "$ref": "#/definitions/MultiRequestsType"
        }
      },
      "required": [
        "Category"
      ],
      "additionalProperties": false
    }
  },
  "type": "object",
  "properties": {
    "Request": {
      "$ref": "#/definitions/RequestType"
    }
  },
  "required": [
    "Request"
  ],
  "additionalProperties": false
}
```

Figure 1 — GeoXACML 3.0 JSON Profile Request schema[1]

```
{
  "$schema": "http://json-schema.org/draft-06/schema",
  "$id": "common-std-with-geometry.schema.json",
  "title": "Common JSON schema to Request and Response objects defined in JSON
profile of XACML 3.0 v1.0",
  "definitions": {
    "AttributeValueType": {
      "anyOf": [
        {
          "type": "boolean"
        },
        {
          "type": "number"
        },
        {
          "type": "string"
        },
        {
          "$ref": "Geometry.schema.json"
        },
        {
          "type": "array",
          "items": {
            "type": "boolean"
          },
          "minItems": 0
        },
        {
          "type": "array",
          "items": {
            "type": [
              "string",
              "number"
            ]
          },
          "minItems": 0
        },
        {
          "type": "array",
          "items": {
            "$ref": "Geometry.schema.json"
          },
          "minItems": 0
```

[1]source: https://github.com/authzforce/xacml-json-model/blob/develop/src/main/resources/org/ow2/authzforce/xacml/json/model/Request.schema.json

```
      }
    ]
  },
  "AttributeType": {
    "type": "object",
    "properties": {
      "AttributeId": {
        "type": "string",
        "format": "uri-reference"
      },
      "Issuer": {
        "type": "string"
      },
      "IncludeInResult": {
        "type": "boolean"
      },
      "DataType": {
        "type": "string",
        "format": "uri-reference"
      },
      "Value": {
        "$ref": "#/definitions/AttributeValueType"
      },
      "SRID": {
        "type": "number"
      },
      "AllowTransformation": {
        "type": "boolean"
      },
      "Precision": {
        "type": "number"
      }
    },
    "required": [
      "AttributeId",
      "Value"
    ],
    "additionalProperties": false
  },
  "AttributeCategoryType": {
    "type": "object",
    "properties": {
      "CategoryId": {
        "type": "string",
        "format": "uri-reference"
      },
      "Id": {
        "type": "string"
      },
      "Content": {
        "type": "string"
      },
      "Attribute": {
        "type": "array",
        "items": {
          "$ref": "#/definitions/AttributeType"
        },
        "minItems": 0
      }
    },
    "required": [
      "CategoryId"
    ],
      "additionalProperties": false
    },
    "IdReferenceType": {
      "type": "object",
      "properties": {
        "Id": {
          "type": "string",
          "format": "uri-reference"
        },
        "Version": {
          "type": "string"
        }
      },
      "required": [
        "Id"
      ],
      "additionalProperties": false
    }
  }
}
```

Figure 2 — common-std-with-geometry.schema.json[2]

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "$id": "https://geojson.org/schema/Geometry.json",
  "title": "GeoJSON Geometry",
  "oneOf": [
    {
      "title": "GeoJSON Point",
      "type": "object",
      "required": [
        "type",
        "coordinates"
      ],
      "properties": {
        "type": {
          "type": "string",
          "enum": [
            "Point"
          ]
        },
        "coordinates": {
          "type": "array",
          "minItems": 2,
          "items": {
            "type": "number"
          }
        },
        "bbox": {
          "type": "array",
          "minItems": 4,
          "items": {
            "type": "number"
          }
        }
```

[2]source: https://github.com/authzforce/xacml-json-model/blob/develop/src/main/resources/org/ow2/authzforce/xacml/json/model/common-std.schema.json

```
  }
},
{
  "title": "GeoJSON LineString",
  "type": "object",
  "required": [
    "type",
    "coordinates"
  ],
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "LineString"
      ]
    },
    "coordinates": {
      "type": "array",
      "minItems": 2,
      "items": {
        "type": "array",
        "minItems": 2,
        "items": {
          "type": "number"
        }
      }
    },
    "bbox": {
      "type": "array",
      "minItems": 4,
      "items": {
        "type": "number"
      }
    }
  }
},
{
  "title": "GeoJSON Polygon",
  "type": "object",
  "required": [
    "type",
    "coordinates"
  ],
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "Polygon"
      ]
    },
    "coordinates": {
      "type": "array",
      "items": {
        "type": "array",
        "minItems": 4,
        "items": {
          "type": "array",
          "minItems": 2,
          "items": {
            "type": "number"
          }
        }
      }
    },
    "bbox": {
      "type": "array",
      "minItems": 4,
      "items": {
        "type": "number"
      }
    }
  }
},
{
  "title": "GeoJSON MultiPoint",
  "type": "object",
  "required": [
    "type",
    "coordinates"
  ],
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "MultiPoint"
      ]
    },
    "coordinates": {
      "type": "array",
      "items": {
        "type": "array",
        "minItems": 2,
        "items": {
          "type": "number"
        }
      }
    },
    "bbox": {
      "type": "array",
      "minItems": 4,
      "items": {
        "type": "number"
      }
    }
  }
},
{
  "title": "GeoJSON MultiLineString",
  "type": "object",
  "required": [
    "type",
    "coordinates"
  ],
  "properties": {
    "type": {
      "type": "string",
      "enum": [
        "MultiLineString"
      ]
    },
    "coordinates": {
      "type": "array",
      "items": {
        "type": "array",
        "minItems": 2,
        "items": {
              "type": "array",
              "minItems": 2,
              "items": {
                "type": "number"
              }
            }
          }
        },
        "bbox": {
          "type": "array",
          "minItems": 4,
          "items": {
            "type": "number"
          }
        }
      }
    },
    {
      "title": "GeoJSON MultiPolygon",
      "type": "object",
      "required": [
        "type",
        "coordinates"
      ],
      "properties": {
        "type": {
          "type": "string",
          "enum": [
            "MultiPolygon"
          ]
        },
        "coordinates": {
          "type": "array",
          "items": {
            "type": "array",
            "items": {
              "type": "array",
              "minItems": 4,
              "items": {
                "type": "array",
                "minItems": 2,
                "items": {
                  "type": "number"
                }
              }
            }
          }
        },
        "bbox": {
          "type": "array",
          "minItems": 4,
          "items": {
            "type": "number"
          }
        }
      }
    }
  ]
```

```
}
```

Figure 3 — GeoXACML 3.0 JSON Profile Geometry schema[3]

[3]source https://geojson.org/schema/Geometry.json

8

MEDIA TYPES FOR ANY DATA ENCODING(S)

8 MEDIA TYPES FOR ANY DATA ENCODING(S)

This Standard defines the following Media Type to be used for an Authorization Decision Request and Authorization Decision encoded according to this profile:

- `application/geoxacml+json`

A

ANNEX A (NORMATIVE) ABSTRACT TEST SUITE

A ANNEX A (NORMATIVE) ABSTRACT TEST SUITE

A.1. Conformance Class Data Model

The purpose of the tests from this conformance class is to construct different ADRs that are send to an implementation compliant with the API Conformance Class.

CONFORMANCE TEST A.1	
IDENTIFIER	`/conf/data-model/definition`
REQUIREMENTS	Requirement 1: `/req/data-type` Requirement 8: `/req/srid` Requirement 9: `/req/precision` Requirement 10: `/req/allow-transformation`
INCLUDED IN	Conformance class 1: `/conf/data-model`
INDIRECT PREREQUISITE	Conformance class 1: `/conf/data-model`
INDIRECT	/conf/data-model
TEST PURPOSE	To validate that the JSON schema for validating ADR contains the Attribute properties `SRID`, `Precision`, `AllowTransformation`.
TEST-METHOD-TYPE	Manual Inspection
TEST METHOD	Construct a JSON encoded ADR and validate that the `Attribute` element contains a valid geometry and the following elements:
A	`SRID` and verify that its value is of type Integer.
B	`Precision` and verify that its value is of type Integer.
C	`AllowTransformation` and verify that its value is of type Boolean.

CONFORMANCE TEST A.2	
IDENTIFIER	`/conf/data-model/crs-axis-order`
REQUIREMENTS	Requirement 2: `/req/crs` Requirement 3: `/req/axis-order`
INCLUDED IN	Conformance class 1: `/conf/data-model`
INDIRECT PREREQUISITE	Conformance class 1: `/conf/data-model`
INDIRECT	/conf/data-model
TEST PURPOSE	To validate that a JSON encoded ADR uses the default CRS and axis-order when no `SRID` element is present.
TEST-METHOD-TYPE	Manual Inspection
TEST METHOD	Construct a JSON encoded ADR and have the `Attribute` value contain a geometry serialized in the default CRS (`urn:ogc:def:crs: OGC::CRS84`) and default axis order (`longitude/latitude`).
A	Verify that the coordinates of the geometry are calculated using `urn:ogc:def:crs: OGC::CRS84`.
B	Verify that the coordinate order uses `longitude/latitude`.

CONFORMANCE TEST A.3	
IDENTIFIER	`/conf/data-model/media-type`
REQUIREMENT	Requirement 4: `/req/media-type`
INCLUDED IN	Conformance class 1: `/conf/data-model`
INDIRECT PREREQUISITE	Conformance class 1: `/conf/data-model`
INDIRECT	/conf/data-model
TEST PURPOSE	To validate that a HTTP POST request for sending a JSON encoded ADR uses the media type `application/geoxacml+json`.
TEST-METHOD-TYPE	Manual Inspection

CONFORMANCE TEST A.3	
TEST METHOD	Construct a HTTP POST request which body is a JSON encoded ADR compliant [_conf_data-model_conf_data-model] and [_conf_data-model_conf_crs-axis-order] and set the `Content-Type` header to value `application/geoxacml+json`.
A	Verify that the HTTP POST request uses media type `application/geoxacml+json`.

CONFORMANCE TEST A.4	
IDENTIFIER	`/conf/data-model/wkt`
REQUIREMENT	Requirement 6: `/req/wkt`
INCLUDED IN	Conformance class 1: `/conf/data-model`
INDIRECT PREREQUISITE	Conformance class 1: `/conf/data-model`
INDIRECT	/conf/data-model
TEST PURPOSE	To validate that a JSON encoded ADR contains an `Attribute` value that is a WKT compliant geometry.
TEST-METHOD-TYPE	Manual Inspection
TEST METHOD	Construct a JSON encoded ADR compliant [_conf_data-model_conf_data-model] and [_conf_data-model_conf_crs-axis-order] and set the `Attribute` value to WKT encoded geometry.
A	Verify that the `Attribute` value is valid WKT.

CONFORMANCE TEST A.5	
IDENTIFIER	`/conf/data-model/wkb`
REQUIREMENT	Requirement 7: `/req/wkb`
INCLUDED IN	Conformance class 1: `/conf/data-model`
INDIRECT PREREQUISITE	Conformance class 1: `/conf/data-model`
INDIRECT	/conf/data-model

CONFORMANCE TEST A.5	
TEST PURPOSE	To validate that a JSON encoded ADR contains an `Attribute` value that is a WKB hexstring compliant geometry.
TEST-METHOD-TYPE	Manual Inspection
TEST METHOD	Construct a JSON encoded ADR compliant [_conf_data-model_conf_data-model] and [_conf_data-model_conf_crs-axis-order] and set the `Attribute` value to WKB hexstring encoded geometry.
A	Verify that the `Attribute` value is valid WKB hexstring.

CONFORMANCE TEST A.6	
IDENTIFIER	`/conf/data-model/geojson`
REQUIREMENT	Requirement 5: `/req/geojson`
INCLUDED IN	Conformance class 1: `/conf/data-model`
INDIRECT PREREQUISITE	Conformance class 1: `/conf/data-model`
INDIRECT	/conf/data-model
TEST PURPOSE	To validate that a JSON encoded ADR contains an `Attribute` value that is a GeoJSON compliant geometry.
TEST-METHOD-TYPE	Manual Inspection
TEST METHOD	Construct a JSON encoded ADR compliant [_conf_data-model_conf_data-model] and [_conf_data-model_conf_crs-axis-order] and set the `Attribute` value to WKB hexstring encoded geometry.
A	Verify that the `Attribute` value is valid GeoJSON geometry.

A.2. Conformance Class Core

CONFORMANCE TEST A.7	
IDENTIFIER	`/conf/core/media-type-impl`

CONFORMANCE TEST A.7	
REQUIREMENTS	Conformance class 2: `/conf/core` Requirement 16: `/req/media-type-impl`
INCLUDED IN	Conformance class 2: `/conf/core`
TEST PURPOSE	To validate that the implementation accepts the media-type `GeoXACML_JSON` for HTTP headers `Content-Type` and `Accept`.
TEST-METHOD-TYPE	Postman
TEST METHOD	Send the ADR constructed and verified in `/conf/data-model/data-model` via HTTP POST to the implementation's `/decision` endpoint (as defined in `API` Conformance Class of Geo XACML 3.0) and verify that the request was not rejected, e.g. with HTTP status code 415.
A	Send the ADR with HTTP POST and `Content-Type` set to `GeoXACML_JSON` and verify that the response status code is not 415.
B	Send the ADR with HTTP POST and `Content-Type` and `Accept` set to `GeoXACML_JSON` and verify that the response `Content-Type` is set to `GeoXACML_JSON`.

CONFORMANCE TEST A.8	
IDENTIFIER	`/conf/core/json-schema-impl`
REQUIREMENTS	Conformance class 2: `/conf/core` Requirement 21: `/req/json-schema-impl` Requirement 16: `/req/media-type-impl` Requirement 17: `/req/srid-impl` Requirement 18: `/req/precision-impl` Requirement 19: `/req/allow-transformation-impl`
INCLUDED IN	Conformance class 2: `/conf/core`
TEST PURPOSE	To validate that the implementation processes an ADR compliant to this GeoXACML 3.0 JSON Profile v1.0 with no error.
TEST-METHOD-TYPE	Postman or OpenAPI
TEST METHOD	Send the ADR constructed and verified in `/conf/data-model/data-model` via HTTP POST to the implementation's `/decision` endpoint (as defined in `API` Conformance Class of Geo XACML 3.0) and verify that the received response (the AD) does not indicate a processing error.
A	Send the ADR constructed and verified in `/conf/data-model/data-model` with HTTP POST and Content-Type `GeoXACML_JSON` to the `/decision` endpoint and verify that the response does not contain an error.

CONFORMANCE TEST A.9	
IDENTIFIER	`/conf/core/wkt-impl`
REQUIREMENTS	Conformance class 2: `/conf/core` Requirement 11: `/req/wkt-impl`
INCLUDED IN	Conformance class 2: `/conf/core`
TEST PURPOSE	To validate that the implementation processes an ADR compliant to this GeoXACML 3.0 JSON Profile v1.0 with no error, when the `Attribute` value contains a WKT encoded geometry.
TEST-METHOD-TYPE	Postman or OpenAPI
TEST METHOD	Send the ADR constructed and verified in `/conf/data-model/wkt` via HTTP POST to the implementation's `/decision` endpoint (as defined in `API` Conformance Class of GeoXACML 3.0) and verify that the received response (the AD) does not indicate a processing error.
A	Send the ADR constructed and verified in `/conf/data-model/wkt` with HTTP POST and Content-Type `GeoXACML_JSON` to the `/decision` endpoint and verify that the response does not contain an error.

CONFORMANCE TEST A.10	
IDENTIFIER	`/conf/core/wkb-impl`
REQUIREMENTS	Conformance class 2: `/conf/core` Requirement 12: `/req/wkb-impl`
INCLUDED IN	Conformance class 2: `/conf/core`
TEST PURPOSE	To validate that the implementation processes an ADR compliant to this GeoXACML 3.0 JSON Profile v1.0 with no error, when the `Attribute` value contains a WKT encoded geometry.
TEST-METHOD-TYPE	Postman or OpenAPI
TEST METHOD	Send the ADR constructed and verified in `/conf/data-model/wkb` via HTTP POST to the implementation's `/decision` endpoint (as defined in `API` Conformance Class of GeoXACML 3.0) and verify that the received response (the AD) does not indicate a processing error.
A	Send the ADR constructed and verified in `/conf/data-model/wkb` with HTTP POST and Content-Type `GeoXACML_JSON` to the `/decision` endpoint and verify that the response does not contain an error.

CONFORMANCE TEST A.11	
IDENTIFIER	`/conf/core/geojson-impl`

CONFORMANCE TEST A.11	
REQUIREMENTS	Conformance class 2: `/conf/core` Requirement 13: `/req/geojson-impl`
INCLUDED IN	Conformance class 2: `/conf/core`
TEST PURPOSE	To validate that the implementation processes an ADR compliant to this GeoXACML 3.0 JSON Profile v1.0 with no error, when the `Attribute` value contains a WKT encoded geometry.
TEST-METHOD-TYPE	Postman or OpenAPI
TEST METHOD	Send the ADR constructed and verified in `/conf/data-model/geojson` via HTTP POST to the implementation's `/decision` endpoint (as defined in `API` Conformance Class of Geo XACML 3.0) and verify that the received response (the AD) does not indicate a processing error.
A	Send the ADR constructed and verified in `/conf/data-model/geojson` with HTTP POST and Content-Type `GeoXACML_JSON` to the `/decision` endpoint and verify that the response does not contain an error.

CONFORMANCE TEST A.12	
IDENTIFIER	`/conf/core/allow-transformation-impl`
REQUIREMENTS	Conformance class 2: `/conf/core` Requirement 19: `/req/allow-transformation-impl`
INCLUDED IN	Conformance class 2: `/conf/core`
TEST PURPOSE	To validate that the implementation honors the `allowTransformation` value.
TEST-METHOD-TYPE	Postman or OpenAPI
TEST METHOD	Use a GeoXACML policy that compares two geometries (e.g. using `geometry-equls`) and send an ADR with a geometry in different CRS from the geometry used in the policy. An implementation that is compliant to the GeoXACML 3.0 Conformance Class `CRS Transformation` will process the request without error. A `Core` compliant implementation must return an error, as it is not capable to execute the required CRS transformation
A	Verify that the implementation is compliant to conformance class `CRS Transformation`
B	Construct a test geometry
C	Construct a simple GeoXACML policy that compares two geometries (e.g. using `geometry-equals`): The first geometry is obtained from the ADR and the second geometry is obtained from the policy. Use the test geometry for the policy
D	Construct an ADR containing the test geometry and send the ADR to the implementation

CONFORMANCE TEST A.12	
E	Verify that the AD contains the desired decision and not an error

CONFORMANCE TEST A.13	
IDENTIFIER	`/conf/core/crs-impl`
REQUIREMENTS	Conformance class 2: `/conf/core` Requirement 14: `/req/crs-impl`
INCLUDED IN	Conformance class 2: `/conf/core`
TEST PURPOSE	To validate that the implementation honors the default CRS `CRS84`.
TEST-METHOD-TYPE	Postman or OpenAPI
TEST METHOD	Use a GeoXACML policy that compares two geometries (e.g. using `geometry-equls`) where the policy geometry is using the default CRS and send an ADR with a geometry using the default CRS. An implementation that honors the default CRS should process the ARD with no errors.
A	Construct a test geometry using the default CRS
B	Construct a simple GeoXACML policy that compares two geometries (e.g. using `geometry-equals`): The first geometry is obtained from the ADR and the second geometry is obtained from the policy. Use the test geometry for the policy
C	Construct an ADR containing the test geometry and send the ADR to the implementation
D	Verify that the AD contains the desired decision and not an error

CONFORMANCE TEST A.14	
IDENTIFIER	`/conf/core/axis-order-impl-1`
REQUIREMENTS	Conformance class 2: `/conf/core` Requirement 15: `/req/axis-order-impl`
INCLUDED IN	Conformance class 2: `/conf/core`
TEST PURPOSE	To validate that the implementation honors the default axis-order `Axis_Order`.
TEST-METHOD-TYPE	Postman or OpenAPI

CONFORMANCE TEST A.14	
TEST METHOD	Use a GeoXACML policy that compares two geometries (e.g. using `geometry-equls`) where the policy geometry is using the default CRSand default axis-order and send an ADR with a geometry using the default CRS and default axis-order. An implementation that honors the default axis-order should process the ARD with no errors.
A	Construct a test geometry using the default CRS and axis-order
B	Construct a simple GeoXACML policy that compares two geometries (e.g. using `geometry-equals`): The first geometry is obtained from the ADR and the second geometry is obtained from the policy. Use the test geometry for the policy
C	Construct an ADR containing the test geometry and send the ADR to the implementation
D	Verify that the AD contains the desired decision and not an error

CONFORMANCE TEST A.15	
IDENTIFIER	`/conf/core/axis-order-impl-2`
REQUIREMENTS	Conformance class 2: `/conf/core` Requirement 15: `/req/axis-order-impl`
INCLUDED IN	Conformance class 2: `/conf/core`
TEST PURPOSE	To validate that the implementation honors the default axis-order `Axis_Order`.
TEST-METHOD-TYPE	Postman or OpenAPI
TEST METHOD	Repeat test `/conf/core/axis-order-impl-1` where the geometry in the ADR has swapped axis.
A	Construct a test geometry using the default CRS and axis-order
B	Construct a simple GeoXACML policy that compares two geometries (e.g. using `geometry-equals`): The first geometry is obtained from the ADR and the second geometry is obtained from the policy. Use the test geometry for the policy with swapped coordinates
C	Construct an ADR containing the test geometry and send the ADR to the implementation
D	Verify that the AD contains the desired decision and not an error

CONFORMANCE TEST A.16	
IDENTIFIER	`/conf/core/error-reporting-impl`
REQUIREMENTS	Conformance class 2: `/conf/core` Requirement 20: `/req/error-reporting-impl`
INCLUDED IN	Conformance class 2: `/conf/core`
TEST PURPOSE	To validate that the implementation provides GeoXACML geometry specific error information.
TEST-METHOD-TYPE	Postman or OpenAPI
TEST METHOD	Verify that the implementation supports the encoding of the `MissingAttributeDetail` as defined in GeoXACML 3.0.
A	Instantiate the implementation with a GeoXACML policy that returns an `Indeterminate` decision response for any request where the geometry value is not encoded using the default CRS. Send such a JSON encoded ADR with HTTP `Content-Type` and `Accept` headers set to `application/geoxacml+json` to the implementation. Evaluate the response and in particular verify that the response has status `Indeterminate` and that there is a `MissingAttributeElement` encoded in JSON.
DESCRIPTION	**NOTE**In principle, this test in JSON should result in the same level of expressiveness as the equivalent test conducted in XML.

B

ANNEX B (INFORMATIVE) EXAMPLES FOR THE GEOXACML 3.0 JSON PROFILE V1.0

ANNEX B (INFORMATIVE) EXAMPLES FOR THE GEOXACML 3.0 JSON PROFILE V1.0

The following sections illustrate the use of the GeoXACML 3.0 JSON Profile v1.0.

B.1. Examples how to encode Geometry in ADR

```
{
    "Attribute": {
        "AttributeId": "subject-location",
        "DataType": "urn:ogc:def:dataType:geoxacml:3.0:geometry",
        "Value"     : "POINT(-77.035278 38.889444)"
    }
}
```

Figure B.1 — Geometry encoding in WKT with default CRS

```
{
    "Attribute": {
        "AttributeId": "subject-location",
        "DataType": "urn:ogc:def:dataType:geoxacml:3.0:geometry",
        "Value"     : "0101000000000000000000000040000000000000001040"
    }
}
```

Figure B.2 — Geometry encoding in WKB with default CRS

```
{
    "Attribute": {
        "AttributeId": "subject-location",
        "DataType": "urn:ogc:def:dataType:geoxacml:3.0:geometry",
        "Value"     : ["POINT(-77.035278 38.889444)", "Point (-122.4538755 37.
8106729)"]
    }
}
```

Figure B.3 — Geometry bag encoding in WKT with default CRS

```
{
    "Attribute": {
        "AttributeId": "subject-location",
        "DataType": "urn:ogc:def:dataType:geoxacml:3.0:geometry",
        "Value"     : "GEOMETRYCOLLECTION(POINT(-77.035278 38.889444), Point (-
122.4538755 37.8106729))"
    }
}
```

Figure B.4 — Homogeneous Geometry Collection encoding in WKT with default CRS

```
{
    "Attribute": {
        "AttributeId": "subject-location",
        "DataType": "urn:ogc:def:dataType:geoxacml:3.0:geometry",
        "Value": {
            "type": "Point",
            "coordinates": [-77.035278, 38.889444]
        }
    }
}
```

Figure B.5 — Geometry encoding in GeoJSON with default CRS

```
{
    "Attribute": {
        "AttributeId": "subject-location",
        "DataType": "urn:ogc:def:dataType:geoxacml:3.0:geometry",
        "Value": [
            {
                "type": "Point",
                "coordinates": [-77.035278, 38.889444]
            },
            {
                "type": "Point",
                "coordinates": [-77.035278, 38.889444]
            }
        ]
    }
}
```

Figure B.6 — Geometry bag encoding in GeoJSON with default CRS

```
{
    "Attribute": {
        "AttributeId": "subject-location",
        "DataType": "urn:ogc:def:dataType:geoxacml:3.0:geometry",
        "SRID":     3857,
        "Value"     : "POINT(-8571600.791082066 4579425.812870098)"
    }
}
```

Figure B.7 — Geometry encoding in WKT with CRS EPSG:3857

```
{
    "Attribute": {
        "AttributeId": "subject-location",
        "DataType": "urn:ogc:def:dataType:geoxacml:3.0:geometry",
        "Precision":  4,
        "Value"     : "POINT(-77.035278 38.889444)"
    }
}
```

Figure B.8 — Geometry encoding in WKT with precision of 4 decimal places

```
{
    "Attribute": {
        "AttributeId": "subject-location",
        "DataType": "urn:ogc:def:dataType:geoxacml:3.0:geometry",
        "AllowTransformation":  true,
        "Value"     : "POINT(-77.035278 38.889444)"
    }
}
```

Figure B.9 — Geometry encoding in WKT with `allowTransformation=true`

B.2. Example GeoXACML 3.0 policy, request and response

```
{
  "Request": {
    "Category": [
      {
        "CategoryId": "urn:oasis:names:tc:xacml:1.0:subject-category:access-
subject",
        "Attribute": [
          {
            "AttributeId": "subject-location",
            "DataType": "urn:ogc:def:dataType:geoxacml:3.0:geometry",
            "SRID": 4326
            "Value": "POINT(38.889444 -77.035278)"
          }
        ]
      }
    ]
  }
}
```

Figure B.10 — Request example using GeoXACML 3.0 JSON schema

```
{
  "Response": [
    {
      "Status": {
        "StatusCode": {
          "Value": "urn:ogc:def:function:geoxacml:3.0:crs-error"
        },
        "StatusMessage": "Geometry must be encoded using specified CRS",
        "StatusDetail": {
          "MissingAttributeDetail": {
            "DataType": "urn:ogc:def:dataType:geoxacml:3.0:geometry",
            "Category": "urn:oasis:names:tc:xacml:1.0:subject-category:access-
subject",
            "SRID": 3857,
            "AttributeId": "subject-location"
          }
        }
      },
      "Decision": "Indeterminate"
    }
  ]
}
```

Figure B.11 — Response example using GeoXACML 3.0 JSON schema including `MissingAttributeDetail`

NOTEThe Response above can be received using the GeoXACML 3.0 policy below.

```
<xacml3:PolicySet xmlns:xacml3="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
  xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 http://
docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:ns6="http://www.
w3.org/2005/Atom"
  xmlns:ns5="http://authzforce.github.io/core/xmlns/pdp/8"
  xmlns:ns4="http://authzforce.github.io/pap-dao-flat-file/xmlns/properties/3.
6"
  xmlns:ns3="http://authzforce.github.io/rest-api-model/xmlns/authz/5"
PolicySetId="root"
  Version="1"
  PolicyCombiningAlgId="urn:oasis:names:tc:xacml:3.0:policy-combining-
algorithm:deny-overrides">
  <xacml3:Target />
  <xacml3:Policy PolicyId="urn:ogc:geoxacml:3.0:conformance-test:core:policy:
geometry-encoding"
    Version="1"
    RuleCombiningAlgId="urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:
permit-overrides">
    <xacml3:Description>http://www.opengis.net/spec/GEOXACML/3.0/Core/conf/
function-equals/support-valid</xacml3:Description>
    <xacml3:Target />
    <xacml3:Rule RuleId="precision6" Effect="Permit">
      <xacml3:Target>
        <xacml3:AnyOf>
          <xacml3:AllOf>
            <xacml3:Match MatchId="urn:ogc:def:function:geoxacml:3.0:geometry-
has-precision">
              <xacml3:AttributeValue DataType="http://www.w3.org/2001/
XMLSchema#integer">4</xacml3:AttributeValue>
              <xacml3:AttributeDesignator
                Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
subject"
                AttributeId="subject-location" DataType="urn:ogc:def:dataType:
geoxacml:3.0:geometry"
                MustBePresent="true" />
            </xacml3:Match>
          </xacml3:AllOf>
        </xacml3:AnyOf>
      </xacml3:Target>
      <xacml3:Condition>
        <xacml3:Apply FunctionId="urn:ogc:def:function:geoxacml:3.0:geometry-
equals">
          <xacml3:AttributeValue DataType="urn:ogc:def:dataType:geoxacml:3.0:
geometry"
            xmlns:geoxacml="http://www.opengis.net/spec/geoxacml/3.0"
            geoxacml:srid="3857"
          >POINT(-8571600.791082066 4579425.812870098)</xacml3:AttributeValue>
          <xacml3:Apply FunctionId="urn:ogc:def:function:geoxacml:3.0:geometry-
one-and-only">
            <xacml3:AttributeDesignator
              Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
subject"
              AttributeId="subject-location" DataType="urn:ogc:def:dataType:
geoxacml:3.0:geometry"
              MustBePresent="true" />
          </xacml3:Apply>
        </xacml3:Apply>
      </xacml3:Condition>
    </xacml3:Rule>
    <xacml3:Rule RuleId="DenyAll" Effect="Deny"></xacml3:Rule>
  </xacml3:Policy>
</xacml3:PolicySet>
```

Figure B.12 — GeoXACML 3.0 policy example causing a `MissingAttributeDetail` response when the request geometry is not in CRS84 and `AllowTransformation=false`

C

ANNEX C (INFORMATIVE) REVISION HISTORY

ANNEX C (INFORMATIVE) REVISION HISTORY

Table C.1

DATE	RELEASE	EDITOR	PRIMARY CLAUSES MODIFIED	DESCRIPTION
2022-11-08	0.1	Andreas Matheus	all	initial version
2022-12-19	0.2	Andreas Matheus	all	support for additional properties in `Attribute` element; JSON schema added
2022-12-22	0.3	Andreas Matheus	all	simplification of requirements classes and conformance classes
2023-01-12	0.4	Andreas Matheus	all	align requirements, requirements classes, conformance classes, conformance tests using the new Metanorma annotations
2023-01-13	0.5	Andreas Matheus	all	applied OGC NA-Policy to Metanorma annotations
2023-02-06	0.6	Andreas Matheus	all	incorporated Carl Reed’s comments

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] *Edward Huang: 5 JSON Denial Attack that Every Hacker Take Advantage Of* (2021), https://edward-huang.com/programming/2021/03/09/5-json-denial-attack-that-every-hacker-take-advantage-of/