OGC Geospatial eXensible Access Control Markup Language (GeoXACML) 3.0 Core

Copyright © 2013 Open Geospatial Consortium
To obtain additional rights of use, visit http://www.opengeospatial.org/legal/.

Attention
This document is written using the OGC template for Implementation Standards and contains normative language.

But it is important to notice that this OGC publication represents the opinion of the author and submitters regarding GeoXACML 3.0 Core.

Warning
This document is not an OGC Standard. This document is an OGC Discussion Paper and is therefore not an official position of the OGC membership. It is distributed for review and comment. It is subject to
OGC 13-100

change without notice and may not be referred to as an OGC Standard. Further, an OGC Discussion Paper should not be referenced as required or mandatory technology in procurements.

License Agreement

Permission is hereby granted by the Open Geospatial Consortium, (“Licensor”), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.

THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER’S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR’s sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications. This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.
Contents
1. Scope ............................................................................................................................ 7
2. Conformance ................................................................................................................ 7
   2.1 Overview ................................................................................................................ 7
   2.2 Specification identifier ........................................................................................... 8
   2.3 Conformance classes related to GeoXACML implementation .............................. 8
   2.4 Backward Compatibility with GeoXACML 1.0 .................................................... 8
3. References .................................................................................................................. 10
4. Terms and Definitions ............................................................................................... 10
5. Conventions ................................................................................................................ 13
   5.1 Abbreviated terms ............................................................................................... 13
   5.2 Document presentation of the specification .......................................................... 14
6. GeoXACML 3.0 Core Introduction (Informative) ..................................................... 14
   6.1 Introduction .......................................................................................................... 14
   6.2 Policy Language and Authorization Models ....................................................... 15
   6.3 Information Flow Model ...................................................................................... 17
   6.4 Extension capabilities of XACML ....................................................................... 19
      6.4.1 Defining a new Data Type Geometry ........................................................... 19
      6.4.2 Encoding of Data Type Geometry ............................................................... 19
      6.4.3 Defining a new Function ............................................................................... 20
      6.3.3 Defining new Identifiers / Attributes .......................................................... 21
   6.5 XACML as a programming language ................................................................... 22
6.6 GeoXACML Core – BASIC Conformance Class (Informative) ......................... 25
6.7 GeoXACML Core – ADVANCED Conformance Class (Informative) .......... 26
6.8 GeoXACML Core – ANALYSIS Conformance Class (Informative) .......... 27
6.9 Use Case – Restricting access to geospatial data ................................................. 29
6.10 Use Case – Restricting access based on user location ................................. 30

7. Requirements for GeoXACML 3.0 Core .............................................................. 30
7.1 Introduction (informative) ................................................................................... 30
7.2 Common Requirements Class ................................................................. 31
7.3 Requirements Class: Specification ................................................................. 32

Annex A: Conformance Class Abstract Test Suite (Normative) ......................... 46
A.1 Conformance testing XACML 3.0 ................................................................. 46
A.2 Conformance testing Data Type Geometry ................................................... 46
A.3 Conformance testing Functions ................................................................. 46
A.4 Conformance testing Condition Functions ................................................... 47
A.5 Conformance testing Data Type Geometry ................................................... 47
A.6 Conformance testing Functions ................................................................. 47
A.7 Conformance testing Data Type Geometry ................................................... 48
A.8 Conformance testing Functions ................................................................. 48
i. **Abstract**

This standard defines the version 3.0 of a geospatial extension to the OASIS eXtensible Access Control Markup Language (XACML) Version 3.0 standard. It thereby enables the interoperable definition of access rights / constraints using the XACML 3.0 language, processing model and policy schema but extends the ability to phrase conditions on geographic characteristics of subjects, resources and objects.

In that sense, a GeoXACML policy could restrict access to geospatial information, e.g. provided by OGC Web Services. However, a GeoXACML policy could also restrict access to non geospatial assets by stating restrictions for access based on the location of the user (or the mobile device used) trying to access the protected assets. Therefore, this standard applies to main stream IT.

For enabling processing of access control decisions based on geometry, Geospatial eXensible Access Control Markup Language (GeoXACML) 3.0 Core inherits by normative reference ISO 19125 which defines a geometry model and functions on geometry instances which enrich the XACML 3.0 specification.

ii. **Keywords**

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

ogcdoc, OGC document, standard, GeoXACML, XACML, access control, geometric XACML

iii. **Preface**

This document defines the version 3.0 of the geospatial extension to the OASIS eXtensible Access Control Markup Language (XACML) Version 3.0. It hereby supersedes the previous version GeoXACML 1.0 which defines the geospatial extension to OASIS extensible Access Control Markup Language 2.0. It is important to notice that version 2.0 of GeoXACML does not exist!

The geospatial extension defined by GeoXACML 3.0 Core uses the extension points from OASIS eXtensible Access Control Markup Language (XACML) Version 3.0 to introduce a new data type Geometry and functions that extend the set of available functions in XACML 3.0. The semantics of the data type Geometry and the functions introduced is defined in ISO 19125.

In a nutshell, GeoXACML 3.0 defines the appropriate XML wrapper to fit OGC Simple Features into XACML 3.0.
Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

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

University of the Bundeswehr
Oracle
Defense Information Systems Agency (DISA)
National Geospatial-Intelligence Agency (NGA)

v. Submitters
All questions regarding this submission should be directed to the editor:

<table>
<thead>
<tr>
<th>Name</th>
<th>Affiliation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andreas Matheus</td>
<td>andreas.matheus &lt;at&gt; unibw.de</td>
</tr>
</tbody>
</table>
1. Scope

This Standard defines Geospatial eXensible Access Control Markup Language (GeoXACML) 3.0 Core as a geospatial extension to OASIS eXtensible Access Control Markup Language (XACML) Version 3.0.

As such, this specification defines implementation rules for a geospatially enriched Policy Decision Point (GeoPDP) as defined by OASIS eXtensible Access Control Markup Language (XACML) Version 3.0.

2. Conformance

Conformance to this standard can be reached by an implementation of a geospatially enriched Policy Decision Point (GeoPDP). In order to fulfill this, this standard provides different conformance classes under the standardization target “implementation”.

In order to ensure that an implementation provides successful processing of a GeoXACML 3.0 policy, responds to authorization decision requests, and produces compliant authorization decisions, this standard defines conformance classes under the standardization target “instance”.

Finally, for the purpose of defining semantics when extending the XACML 3.0 processing language, this standard also defines requirements and conformance classes under the standardization target “specification”.

2.1 Overview

This Standard defines multiple standardization target types:

- Specification: Help to define processing semantics when extending the XACML 3.0 model. This standardization target type is mainly used for defining the GeoXACML 3.0 Adapter to ISO 19125.

- Instance: Ensure that GeoXACML 3.0 policies, ADRs (Authorization Decision Requests) and ADs (Authorization Decisions) are properly encoded
  - policy instances: i.e. XML documents that encode a GeoXACML compliant policy defining access rights / constraints;
OGC 13-100

- request instances: i.e. XML documents that encode a GeoXACML compliant ADR;
- response instances: i.e. XML documents that encode a GeoXACML compliant AD;

Implementation: Ensure that geometry encoding and functions on geometry instances are understood by a GeoPDP implementation when processing GeoXACML policies as well as requests and producing authorization decisions;

NOTE: GeoXACML 3.0 is XACML 3.0 schema compliant. Therefore, a GeoXACML 3.0 Policy, or a ADR or AD instance is using the XACML 3.0 namespace as defined in OASIS eXtensible Access Control Markup Language (XACML) Version 3.0.

2.2 Specification identifier

All requirements-classes and conformance-classes described in this document are owned by the specification identified as http://www.opengis.net/spec/GEOXACML/3.0/Core

2.3 Conformance classes related to GeoXACML implementation

The conformance rules are based on processing logic as defined by GeoXACML 3.0.

Table 2 — Conformance classes related GeoXACML implementation

<table>
<thead>
<tr>
<th>Conformance class</th>
<th>Description</th>
<th>Clause</th>
</tr>
</thead>
<tbody>
<tr>
<td>BASIC</td>
<td>Bag and Set as well as Condition functions, e.g. test functions for topological relations</td>
<td></td>
</tr>
<tr>
<td>ADVANCED</td>
<td>BASIC + functions for marshall / unmarshall + simple analysis functions including e.g. union, intersection</td>
<td></td>
</tr>
<tr>
<td>ANALYSIS</td>
<td>ADVANCED + all analysis functions e.g. convexHull, PolygonN</td>
<td></td>
</tr>
</tbody>
</table>

2.4 Backward Compatibility with GeoXACML 1.0

This version of GeoXACML reflects the new Policy structure mandated by XACML 3.0. Because the XACML 3.0 Schema is different from the Schema published with XACML 2.0, a GeoXACML 1.0 Policy document has a different structure than a GeoXACML 3.0 policy document. Even though a validation of a GeoXACML policy document of version 1.0 must fail if tested against the XACML 3.0 Schema, a transformation from the XACML 2.0 to the XACML 3.0 structure is possible. Therefore, a GeoXACML 1.0 policy can still be used in the GeoXACML 3.0 context, but a structure change is required.

In order to support the transformation from a GeoXACML 1.0 policy to a GeoXACML 3.0 policy, all GeoXACML 1.0 functions must be supported by GeoXACML 3.0. Even
though GeoXACML 3.0 is based on a new version of ISO 19125, additional capabilities are introduced which guarantee backwards compatibility for the functions. Implementations must take care that all new capabilities introduced in ISO 19125, e.g. the geometry types Empty and Circle, get new URNs indicating the Version 3.0. This requires that a policy document transformation from GeoXACML 1.0 to 3.0 includes the change of the function URNs as it is possible in 3.0 to process Empty and Circle which was not supported in 1.0.

Precaution must be taken with GeoXACML 1.0 policy instances, using GML 2 or GML 3 encoding extension, can actually be transformed to 3.0: A transformation is only possible, if all GML encoded geometries of the GeoXACML 1.0 policy instance are supported by a given Encoding Extension to this standard.
3. References

The following normative documents contain provisions that, through reference in this text, constitute provisions of this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

The following referenced documents are indispensable for the application 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.


ISO 19125\(^1\) http://portal.opengeospatial.org/files/?artifact_id=25355

**OGC Naming Authority (OGC-NA) Policies & Procedures OGC Document 09-046r2**
http://www.opengis.net/doc/POL/OGC-NA/1.1


4. Terms and Definitions

This document uses the terms defined in Sub-clause 5.3 of [OGC 06-121r8], which is based on the ISO/IEC Directives, Part 2, Rules for the structure and drafting of International Standards. In particular, the word “shall” (not “must”) is the verb form used to indicate a requirement to be strictly followed to conform to this standard.

For the purposes of this document, the following terms and definitions apply.

\(^1\) Currently under revision

\(^2\) \(A \lor B = \neg (\neg A \land \neg B)\)

\(^3\) Please see the normative section for the mandatory set of functions defined for the basic and the advanced
4.1 element <XML>

Basic information item of an XML document containing child elements, attributes and character data.

[ISO 19136:2007]

NOTE From the XML Information Set: —Each XML document contains one or more elements, the boundaries of which are either delimited by start-tags and end-tags, or, for empty elements, by an empty-element tag. Each element has a type, identified by name, sometimes called its „generic identifier“ (GI), and may have a set of attribute specifications. Each attribute specification has a name and a value.

4.2 schema <XML Schema>

Collection of schema components within the same target namespace.

EXAMPLE Schema components of W3C XML Schema are types, elements, attributes, groups, etc.

[ISO 19136:2007]

4.3 schema document <XML Schema>

XML document containing schema component definitions and declarations.

NOTE The W3C XML Schema provides an XML interchange format for schema information. A single schema document provides descriptions of components associated with a single XML namespace, but several documents may describe components in the same schema, i.e. the same target namespace.

[ISO 19136:2007]

4.4 GeoPDP

A Geospatial Policy Decision Point is an implementation of GeoXACML. It provides the capabilities to process the data type Geometry and the functions defined on Geometry. Because a GeoXACML compliant implementation must implement all mandatory capabilities of XACML, a GeoPDP is always capable to process “pure” XACML policies.

4.5 XACML definitions

The following definitions, as defined in OASIS eXtensible Access Control Markup Language (XACML) Version 3.0 are listed here for ease of reading.

Access - Performing an action

Advice - A supplementary piece of information in a policy or policy set which is provided to the PEP with 11 the decision of the PDP.

Access control - Controlling access in accordance with a policy
**Action** - An operation on a resource

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

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

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

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

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

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

**Conjunctive Normal Form** - xxxx

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

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

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

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

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

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

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

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

**Obligation** - An operation specified in a policy or policy set that should be performed by the PEP in conjunction with the enforcement of an authorization decision
**Policy** - A set of rules, an identifier for the rule-combining algorithm and (optionally) a set of obligations. May be a component of a policy set

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

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

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

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

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

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

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

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

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

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

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

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

5. **Conventions**

5.1 **Abbreviated terms**

1D One Dimensional

2D Two Dimensional
This document presents the GeoXACML 3.0 Core specification using a representation that follows the structures defined in the OGC Policy [The Specification Model — A Standard for Modular specifications OGC Document 08-131r3. http://www.opengis.net/doc/POL/SPEC]. All normative material is organized as requirements, conformance tests and conformance classes. Each is identified with a URI, and the content and dependencies are described in tables whose structure matches the specification model.

6. GeoXACML 3.0 Core Introduction (Informative)

6.1 Introduction

GeoXACML 3.0 Core defines a geospatial extension to the OASIS eXtensible Access Control Markup Language (XACML) Version 3.0 by introducing the data type Geometry and functions that work on the data type Geometry as defined in ISO 19125.
The XACML standard can be separated into two main sections, which are introduced in more detail in the following sections: (i) Policy Language and Authorization Model as well as (ii) Information Flow Model.

6.2 Policy Language and Authorization Models

The XACML Policy Language Model defines an XML encoding for expressing general purpose access restrictions and extension points to define your own Attribute Values, Functions, etc. The entire set of access rights/restrictions (rules) defines an XACML Policy. The Policy is structured, according to the following UML diagram (see Figure 1).

The top level element can either be a <Policy> or <PolicySet>. The <PolicySet> can host zero or more <PolicySet> elements, which can be included inline or by reference. This powerful feature allows the reuse of pre-defined policy segments as well as the integration of multiple policies.

Each <PolicySet> element can host one or more <Policy> elements, which is the container for a set of <Rule> elements. Inside the <Rule> element, conditions can be formed to express complex access restrictions, using the <Condition> element.

Each <PolicySet>, <Policy> and <Rule> element have a <Target> element, which can be used to define simple matching conditions for categories such as Subject, Action, Resource and Environment. This allows the effective structuring of a policy into subtrees, which eases the maintenance of rights defined in a policy. On the other hand, the simple matching in a <Target> element ensures fast decision making, when it comes to deriving an authorization decision.

In order to derive an authorization decision (i.e. XACML authorization decision) for a given request, the XACML policy is traversed from the top (i.e. <PolicySet> element) to the leaves (i.e. <Rule> elements). For all matching <Rule> elements, their Effect (i.e. Permit or Deny) is taken as the most basic driver for the authorization decision. By traversing up the policy the effects of all Rules – associated to a <Policy> element – are combined using the Rule Combining Algorithm. The resulting effects of all <Policy> elements are matched on the next highest level, until reaching the top <PolicySet> element; the Policy Combining Algorithm creates the final effect of the entire policy, which represents the authorization decision.

The XACML Policy Language defines four different results for the authorization decision: (i) Permit, (ii) Deny, (iii) Indeterminate and (iv) NotApplicable. Finally, the process of deriving an authorization decision can result in an error, which is documented as additional information in the <Decision> element.

OASIS eXtensible Access Control Markup Language (XACML) Version 3.0 also defines the concepts of Advice and Obligations. Whereas it is optional for a PEP to process an Advice, it is mandatory for Obligations. The PEP may only act according to the decision,
after the successful processing of all Obligations attached to the decision. Basically, two
options exist: The decision can be “Permit with Obligation” or “Deny with Obligation”,
which can be expressed in the <Obligation> element, attached to the <Rule>, <Policy> or
<PolicySet> element. It is also possible to attach an Advice to the decision. Care must be
taken when structuring a policy that comprises Obligations as the processing of included
Obligations vary with the different Combining Algorithms.

Regarding the differences of XACML 3.0 over 2.0 it is important to mention the change
in the new structure of the <Target>. For XACML 2.0 matching was separated into
Subject, Action, Resource and Environment, XACML 3.0 supports the Matching as a
Conjunctive Normal Form (CNF) using the elements <AnyOf> and <AllOf> where each
individual Matching must specify a category (e.g. Subject, Action, Resource or
Environment). This improves the conditional matching when attribute values from
different categories must be compared; this was not possible in XACML 2.0.

![XACML Policy Language Model](image)

**Figure 1 — XACML Policy Language Model**

As a consequence of this change to the <Target> matching, the XACML 3.0 matching is
more comprehensive and therefore enables automatic transformation from XACML 2.0 to
XACML 3.0. However, direct transformed into the other direction (from XACML 3.0 to
XACML 2.0 <Target> matching) may be impossible, depending on the concrete matching.

The following example illustrates the deficit in the XACML 2.0 matching by introducing a <Target> matching that is XACML 3.0 but not XACML 2.0 compliant. The breaking is caused by the OR matching of different categories; here environment and subject category: considering the IP address of the client to reside in the environment category and the subject name to reside in the subject category, the OR matching between these categories cannot be mapped to XACML 2.0 matching as the logic always is AND between different categories. Even though the logic could be transformed using the de Morgan transformation\(^2\) – which is only possible if many other constraints that are out of scope for this example have been certified – to adopt the AND matching between the different categories, the required “not equals” matching function is not available in XACML 2.0.

![Example <Target> matching supported by XACML 3.0 but not by XACML 2.0](image)

Another improved feature in the Language Model is the ability to define the delegation of rights. In that respect, GeoXACML introduces the concept of a Policy Issuer. The use of <PolicyIssuer> allows to distinguish between trusted and delegated policies. Any <PolicySet> or <Policy> that does not indicate a <PolicyIssuer> is trusted and participates in the process of deriving an authorization decision. Any <PolicySet> or <Policy> that includes a <PolicyIssuer> element requires the PDP to determine if a “trusted relationship” between the accessing subject in the ADR context and the <PolicyIssuer> exists. Only in the case of a successful trust verification, the <PolicySet> or <Policy> may participate in the process of deriving an authorization decision.

6.3 Information Flow Model

The XACML Information Flow Model defines the architecture of a modular and distributed access control system. In addition, it defines the exchange of messages between the components and the structure of the messages. The following figure illustrates the informative architecture and the sequence of messages, sent between the components of the access control system.

\[^2\text{ } A \lor B = \neg (\neg A \land \neg B)\]
The Policy Administration Point (PAP) is the component that allows one or multiple policy administrators to maintain access rights in a set of policies. In addition, the PAP might provide an interface for requesting policies.

The Policy Decision Point (PDP) is the component that derives an authorization decision based on a request, received from one or multiple Policy Enforcement Point(s) (PEP). A PDP may request policies from the PAP or use a policy repository on file or in a database.

The Policy Enforcement Point (PEP) can be characterized as a binary switch that either forwards the intercepted request from the client to the service (and the response from the service to the client respectively) or replies with an adequate error message. The decision if the request or response is to be forwarded or blocked depends on the authorization...
decisions, received from the PDP. Because the PEP must request authorization decisions in a particular XACML message format, it is the duty of the context handler to collect all relevant information and prepare the authorization decision request message. The information, collected by the context handler can include the identity information about the user, the action to be taken on the resource, information about the resource itself, the IP address of the client, the time of the request, certificate information, etc… In order to collect all relevant information, it can be required to request such information from the Policy Information Point (PIP).

The PIP provides interfaces to the context handler in a proxy fashion to simplify the information fetching. For example, the PIP could provide an interface for collecting user credentials which maps to LDAP, Kerberos, etc. Also, the PIP could provide a database or Web Service interface to enable the collection of resources via a “trusted custom” channel.

6.4 Extension capabilities of XACML

The XACML specification defines the non-normative extensibility points (section 8, OASIS eXtensible Access Control Markup Language (XACML) Version 3.0). For this specification, it is important to note that the 

6.4.1 Defining a new Data Type Geometry

Section 8.1 of the XACML specification states that “The following XML attributes have values that are URIs. These may be extended by the creation of new URIs associated with new semantics for these attributes. … “Category, AttributeId, DataType, FunctionId, MatchId, ObligationId, AdviceId, PolicyCombiningAlgId, RuleCombiningAlgId, StatusCode, SubjectCategory.”

This capability allows the definition of geometry data type, as defined by GeoXACML using the extension point DataTypes. The XACML compliant URN is defined to be

6.4.2 Encoding of Data Type Geometry

Section 8.2 of the XACML specification states that “<xacml:AttributeValue> and <xacml-context:AttributeValue> elements MAY contain an instance of a structured XML data-type.”.

This provides two options for encoding a geometry:

1. As a string value to the <AttributeValue> element
The GeoXACML 3.0 Core defines the mandatory encoding for using the string value to use Well Known Text

2. As XML

The GeoXACML 3.0 Core defines an extension point such that Encoding Extension can define different XML encodings.

```xml
<xs:element name="AttributeValue" type="xacml:AttributeValueType"
substitutionGroup="xacml:Expression"/>
<xs:complexType name="AttributeValueType" mixed="true">
  <xs:complexContent mixed="true">
    <xs:extension base="xacml:ExpressionType">
      <xs:sequence>
        <xs:any namespace="##any" processContents="lax" minOccurs="0"
                   maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
      <xs:anyAttribute namespace="##any" processContents="lax"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Figure 4 — XACML schema definition of the `<AttributeValue>` element

### 6.4.3 Defining a new Function

A `<Function>` element has an attribute named FunctionId, which is of type `xs:anyURI`. According to the extension capabilities of XACML, additional functions can be defined by associating a unique `FunctionId` to it.

This capability allows the definition of geo-specific functions, as defined by GeoXACML.

```xml
<xs:element name="Function" type="xacml:FunctionType"/>
<xs:complexType name="FunctionType">
  <xs:attribute name="FunctionId" type="xs:anyURI" use="required"/>
</xs:complexType>
```

Figure 5 — XACML schema definition of the `<Function>` element
6.3.3 Defining new Identifiers / Attributes

An `<AttributeDesignator>` element allows fetching information from an XACML AuthorizationDecisionRequest based on named attributes. In order to specify new names for attributes, the `<AttributeDesignatorType>` has an attribute named `AttributeId`. In order to use GeoXACML specific data types, the value of the attribute `DataType` SHALL be used according to the specified data types. According to the extension capabilities of XACML, additional identifier-names or attribute-names can be defined by associating a unique `AttributeId` to it.

An `<AttributeSelector>` element allows fetching information from an XACML AuthorizationDecisionRequest based on the XML encoded information as it can be inserted into the `<Content>` element. The value of the attribute named `DataType` SHALL be used according to the specified data type.

These capabilities allow the fetching of geo-specific information from the ADR using either the XACML AttributeDesignator or the AttributeSelector without modifying any XACML schema (policy, authorization decision request or authorization decision schema).

The following two figures highlight the extension points in the XACML schema that are used by GeoXACML.

**Figure 6 — XACML schema definition of the `<AttributeDesignatorType>` element**

```
<xs:complexType name="AttributeDesignatorType">
  <xs:complexContent>
    <xs:extension base="xacml:ExpressionType">
      <xs:attribute name="Category" type="xs:anyURI" use="required"/>
      <xs:attribute name="AttributeId" type="xs:anyURI" use="required"/>
      <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
      <xs:attribute name="Issuer" type="xs:string" use="optional"/>
      <xs:attribute name="MustBePresent" type="xs:boolean" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

**Figure 7 — XACML schema definition of the `<AttributeSelectorType>` element**

```
<xs:complexType name="AttributeSelectorType">
  <xs:complexContent>
    <xs:extension base="xacml:ExpressionType">
      <xs:attribute name="Category" type="xs:anyURI" use="required"/>
      <xs:attribute name="ContextSelectorId" type="xs:anyURI" use="optional"/>
      <xs:attribute name="Path" type="xs:string" use="required"/>
      <xs:attribute name="DataType" type="xs:anyURI" use="required"/>
      <xs:attribute name="MustBePresent" type="xs:boolean" use="required"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```
6.5 XACML as a programming language

Looking at the actual extension GeoXACML from a developers point of view, it can best be understood when XACML is seen as a programming language where the dialect is encoded in XML. In that respect, GeoXACML bridges the gap between the OO Design described in ISO 19125 and the Procedural Design taken in XACML. At the highest level, GeoXACML can be understood according to the Adapter Pattern [see Gamma, et.al.], as illustrated in the following example.

Let’s assume that we want to implement a condition function testing if the dimension of a given geometry instance is greater than zero and if so, return the integer value 1.

The following pseudo code will illustrate the approach following the OO Design Pattern:

```
1:   Integer Permit = 1;
2:   Integer Rule ()
3:   {
4:     Geometry g = new Point ("Point (0 0 )");
5:     if (g.Dimension() > 0)
6:       return Permit;
7:   }
```

Line 1: Defines the constant Permit as type Integer

Line 2: Define the condition function Rule

Lines 3-7: The Code block for the function

Line 4: Instantiate a geometry of concrete type Point

Line 5: The actual condition testing the dimension of the geometry instance and comparing it with the literal value 0

Line 6: Iff the condition from line 5 is true, return 1

The pseudo code below illustrates the above Rule following Procedural Programming:

```
1:   Integer Permit = 1;
2:   Integer Rule ()
3:   {
4:     Geometry g = new Point ("Point (0 0 )");
5:     if (GeoXACML.Dimension(g) > 0)
```
The only major difference to this pseudo code is in line 5, where the function Dimension is not called from the geometry object instance but statically. For that to work, the static function with the signature \( \text{Dimension}(\text{Geometry}): \text{Integer} \) must have been defined in a adapter class GeoXACML. To define different adapters by different conformance classes is what the GeoXACML 3.0 Core specification is all about.

In order to be compliant with the XACML environment, the functions defined by GeoXACML through the adapter pattern must also implement XACML compliant error handling. This is particularly important as GeoXACML does not differentiate the concrete sub-types to Geometry as defined in ISO 19125. And therefore, it is possible that a GeoXACML function gets invoked on a geometry type for which it is not defined. Therefore, a GeoXACML implementation must check the concrete types of parameters and issue a XACML conformant PROCESSING_ERROR in case the function is not defined for the data type of the parameter. Furthermore, functions defined by ISO 19125 often return NULL in the case of an error. This is not possible with XACML; instead a PROCESSING_ERROR must be returned. In that sense, the Adapter Pattern (as defined in Gamma et.al.) used in GeoXACML adapts the function signature and its behavior as defined in ISO 19125 to be conform with XACML 3.0.

The following pseudo code illustrates a more precise Adapter behavior for the example function.

```java
class GeoXACML <<extends>> ISO19125 {
    public static Integer Dimension (Geometry g) {
        if (g instanceof Empty) {
            return new ProcessingError("operation not defined on Empty");
        }
        try {
            return g.dimension();
        }
        catch (NullException) {
            return new ProcessingError("operation error: NULL");
        }
}
```

Line 5-8: Adapt the behavior to GeoXACML environment and check for the concrete type of the parameter.
Line 7: Return PROCESSING_ERROR if actual type is Empty.

Line 13-16: Adapt the error behavior of the function as defined in ISO 19125.

Because XACML 3.0 mandates an XML encoding of the programming logic, the above condition must be presented in XML. The following XML code illustrates the exact same logic but compliant with the XACML 3.0 Schema and using the ISO 19125 function defined in GeoXACML.

```xml
1: <Rule RuleId="dimension" Effect="Permit">
2:   <Condition>
3:     <Apply
4:       FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-greater-than">
5:       <Apply FunctionId="urn:ogc:def:function:geoxacml:3.0:geometry-dimension">
6:         <AttributeValue DataTypes="urn:ogc:def:datatype:geoxacml:3.0:geometry" Point(0 0)"/>
7:       </Apply>
8:       <AttributeValue DataTypes="http://www.w3.org/2001/XMLSchema#integer">0</AttributeValue>
9:     </Apply>
10:   </Condition>
11: </Rule>
```

Line 1: Use of the Rule element as defined in XACML 3.0. The “function” return value Permit is already in the definition.

Line 2: Declaring the if-statement using the Condition element as defined in XACML 3.0. Parameters are in line 4 and 7.

Line 3: Declaration of the comparison logic for the if-statement using predefined comparison operators from XACML 3.0.

Line 4: Call of the GeoXACML 3.0 function Dimension using the Apply element as defined in XACML 3.0. Parameter in line 5.

Line 5: Instantiating a concrete geometry of type Point (from ISO 19125) using WKT for the GeoXACML data type Geometry.

Line 7: Declaring the literal value 0 of type Integer to be part of the comparison function.
The final bit that GeoXACML must specify to reach compliance with XACML are URNs for each defined function. So for example the function `Dimension():Integer` as defined in ISO 19125 gets the GeoXACML signature `Dimension(Geometry):Integer` and the function URN is `urn:ogc:def:function:geoxacml:3.0:geometry-dimension`.

### 6.6 GeoXACML Core – BASIC Conformance Class (Informative)

For the purpose of better understanding what the normative section defines, this is the pseudo code that illustrates the implementation for the BASIC conformance class adapter functions. The functions defined in ISO 19125 are shaded grey.

class GeoXACML_BASIC <<extends>> ISO19125
{
    #define Equals "urn:ogc:def:function:geoxacml:3.0:geometry-equals";    
    #define Disjoint "urn:ogc:def:function:geoxacml:3.0:geometry-disjoint";  
    #define Intersects "urn:ogc:def:function:geoxacml:3.0:geometry-intersects";    
    #define Touches "urn:ogc:def:function:geoxacml:3.0:geometry-touches";    
    #define Crosses "urn:ogc:def:function:geoxacml:3.0:geometry-crosses";    
    #define Within "urn:ogc:def:function:geoxacml:3.0:geometry-within";    
    #define Contains "urn:ogc:def:function:geoxacml:3.0:geometry-contains";    
    #define Overlaps "urn:ogc:def:function:geoxacml:3.0:geometry-overlaps";    

    Boolean Equals (Geometry this, Geometry another)
    { return this.dimension(another); }
    Boolean Disjoint (Geometry this, Geometry another)
    { return this.disjoint(another); }
    Boolean Intersects (Geometry this, Geometry another)
    { return this.intersects(another); }
    Boolean Touches (Geometry this, Geometry another)
    { return this.touches(another); }
    Boolean Crosses (Geometry this, Geometry another)
    { return this.crosses(another); }
    Boolean Within (Geometry this, Geometry another)
    { return this.within(another); }
    Boolean Contains (Geometry this, Geometry another)
    { return this.contains(another); }
    Boolean Overlaps (Geometry this, Geometry another)
    { return this.overlaps(another); }
}

Copyright © 2013 Open Geospatial Consortium
6.7 GeoXACML Core – ADVANCED Conformance Class (Informative)

For the purpose of better understand what the normative section defines, this is the pseudo code that illustrates the implementation for the ADVANCED conformance class adapter functions. The functions defined in ISO 19125 are shaded grey.

class GeoXACML_ADVANCED <<extends>> GeoXACML_BASIC
{
    #define IsWithinDistance "urn:ogc:def:function:geoxacml:3.0:geometry-is-within-distance";
    #define Dimension "urn:ogc:def:function:geoxacml:3.0:geometry-dimension";
    #define GeometryType "urn:ogc:def:function:geoxacml:3.0:geometry-type";
    #define SRID "urn:ogc:def:function:geoxacml:3.0:geometry-srid";
    #define AsText "urn:ogc:def:function:geoxacml:3.0:string-from-geometry";
    #define IsEmpty "urn:ogc:def:function:geoxacml:3.0:geometry-is-empty";
    #define IsSimple "urn:ogc:def:function:geoxacml:3.0:geometry-is-simple";
    #define Is3D "urn:ogc:def:function:geoxacml:3.0:geometry-is-3d";
    #define IsClosed "urn:ogc:def:function:geoxacml:3.0:geometry-is-closed";
    #define IsValid "urn:ogc:def:function:geoxacml:3.0:geometry-is-valid";
    #define IsRing "urn:ogc:def:function:geoxacml:3.0:geometry-is-ring";
    #define IsMeasured "urn:ogc:def:function:geoxacml:3.0:geometry-is-measured";
    #define Relate "urn:ogc:def:function:geoxacml:3.0:geometry-relate";
    #define Length "urn:ogc:def:function:geoxacml:3.0:geometry-length";
    #define Area "urn:ogc:def:function:geoxacml:3.0:geometry-area";

    Boolean IsWithinDistance (Geometry this, Geometry another, distance:Double)
    { return this.isWithinDistance(another, distance); }
    Integer Dimension (Geometry this)
    { return this.dimension(); }
    String GeometryType (Geometry this)
    { return this.geometryType(); }
    Integer SRID (Geometry this)
    { return this.srid(); }
    String AsText (Geometry this)
    { return this.asText(); }
    Boolean IsEmpty (Geometry this)
    { return this.isEmpty(); }
    Boolean IsSimple (Geometry this)
    { return this.isSimple(); }
    Boolean Is3D (Geometry this)
    { return this.is3D(); }
    Boolean IsClosed (Geometry this)
    { return this.isClosed(); }
    Boolean IsRing (Geometry this)
    { return this.isRing(); }
    Boolean IsMeasured (Geometry this)
    { return this.isMeasured(); }
    Boolean IsValid (Geometry this)
    { return this.isValid(); }

{ if this instanceOf (Curve or MultiCurve) return this.isClosed(); else return PROCESSING_ERROR; }
Boolean IsValid (Geometry this)
{ return this.isValid(); }
Boolean IsRing (Geometry this)
{ if this instanceOf Curve return this.isRing(); else return PROCESSING_ERROR; }
Boolean IsMeasured (Geometry this)
{ return this.isMeasured(); }
Boolean Relate (Geometry this, Geometry another, String matrix)
{ return this.relate(another, matrix); }
Double Distance (Geometry this, Geometry another)
{ return this.distance(another); }
Double Length (Geometry this)
{ if this instanceOf (Curve or MultiCurve) return this.length(); else return PROCESSING_ERROR; }
Double Area (Geometry this)
{ if this instanceOf (Surface or MultiSurface) return this.area(); else return PROCESSING_ERROR; }
Double X (Geometry this)
{ if this instanceOf Point return this.X(); else return PROCESSING_ERROR; }
Double Y (Geometry this)
{ if this instanceOf Point return this.Y(); else return PROCESSING_ERROR; }
Double Z (Geometry this)
{ if this instanceOf Point return this.Z(); else return PROCESSING_ERROR; }
Double M (Geometry this)
{ if this instanceOf Point return this.M(); else return PROCESSING_ERROR; }

6.8 GeoXACML Core – ANALYSIS Conformance Class (Informative)

For the purpose of better understand what the normative section defines, this is the pseudo code that illustrates the implementation for the ANALYSIS conformance class adapter functions. The functions defined in ISO 19125 are shaded grey.

class GeoXACML_ANALYSIS <<extends>> GeoXACML_ AVANCED
{
  #define Envelope "urn:ogc:def:function:geoxacml:3.0:geometry-envelope"
  #define Boundary " urn:ogc:def:function:geoxacml:3.0:geometry-boundary";
  #define LocateAlong " urn:ogc:def:function:geoxacml:3.0:geometry-locate-along";
  #define LocateBetween " urn:ogc:def:function:geoxacml:3.0:geometry-locate-between";
  #define Buffer " urn:ogc:def:function:geoxacml:3.0:geometry-buffer";
  #define ConvexHull " urn:ogc:def:function:geoxacml:3.0:geometry-convex-hull";
  #define Intersection " urn:ogc:def:function:geoxacml:3.0:geometry-intersection";
  #define Union " urn:ogc:def:function:geoxacml:3.0:geometry-union";
  #define Difference " urn:ogc:def:function:geoxacml:3.0:geometry-difference";
}
Geometry Envelope (Geometry this)
{ return this.envelope(); }
Geometry Boundary (Geometry this)
{ if this instanceOf Surface return this.boundary(); else return PROCESSING_ERROR; }
Geometry LocateAlong (Geometry this, Geometry another, Double mValue)
{ return this.locateAlong(another, mValue); }
Geometry LocateBetween (Geometry this, Double mStart, Double mEnd)
{ return this.locateBetween(mStart, mEnd); }
Geometry Buffer (Geometry this)
{ return this.buffer(); }
Geometry ConvexHull (Geometry this)
{ return this.convexHull(); }
Geometry Intersection (Geometry this, Geometry another)
{ return this.intersection(another); }
Geometry Union (Geometry this, Geometry another)
{ return this.union(another); }
Geometry Difference (Geometry this, Geometry another)
{ return this.difference(another); }
Geometry SymDifference (Geometry this, Geometry another)
{ return this.symDifference(another); }
Integer NumGeometries (GeometryCollection this)
{ return this.numGeometries(); }

#define SymDifference " urn:ogc:def:function:geoxacml:3.0:geometry-sym-difference";
#define NumGeometries " urn:ogc:def:function:geoxacml:3.0:geometry-num-geometries";
#define GeometryN " urn:ogc:def:function:geoxacml:3.0:geometry-n";
#define StartPoint " urn:ogc:def:function:geoxacml:3.0:geometry-start-point";
#define EndPoint " urn:ogc:def:function:geoxacml:3.0:geometry-end-point";
#define NumPoints " urn:ogc:def:function:geoxacml:3.0:geometry-num-points";
#define PointN " urn:ogc:def:function:geoxacml:3.0:geometry-point-n";
#define ExteriorRing " urn:ogc:def:function:geoxacml:3.0:geometry-exterior-ring";
#define NumInteriorRing " urn:ogc:def:function:geoxacml:3.0:geometry-num-interior-ring";
#define InteriorRingN " urn:ogc:def:function:geoxacml:3.0:geometry-interior-ring-n";
#define Centroid " urn:ogc:def:function:geoxacml:3.0:geometry-centroid";
#define PointOnSurface " urn:ogc:def:function:geoxacml:3.0:geometry-point-on-surface";
#define NumPatches " urn:ogc:def:function:geoxacml:3.0:geometry-num-patches";
#define PatchN " urn:ogc:def:function:geoxacml:3.0:geometry-patch-n";
#define BoundaryPolygons " urn:ogc:def:function:geoxacml:3.0:geometry-boundary-polygons";
#define GCFromGeometryBag " urn:ogc:def:function:geoxacml:3.0:geometry-collection-from-geometry-bag";
#define GeometryBagFromGC " urn:ogc:def:function:geoxacml:3.0:geometry-bag-from-geometry-collection";
OGC 13-100

{ return this.numGeometries(); }
Geometry GeometryN (GeometryCollection this, Integer N)
{ if this instanceof GeometryCollection return this.geometryN(N); else return PROCESSING_ERROR; }
Geometry StartPoint (Geometry this)
{ if this instanceof Curve return this.startPoint(); else return PROCESSING_ERROR; }
Geometry EndPoint (Geometry this)
{ if this instanceof Curve return this.endPoint(); else return PROCESSING_ERROR; }
Integer NumPoints (Geometry this)
{ if this instanceof LineString return this.numPoints(); else return PROCESSING_ERROR; }
Geometry PointN (Geometry this, Integer N)
{ if this instanceof LineString return this.PointN(N); else return PROCESSING_ERROR; }
Geometry ExteriorRing (Geometry this)
{ if this instanceof Surface return this.exteriorRing(); else return PROCESSING_ERROR; }
Integer NumInteriorRing (Geometry this)
{ if this instanceof Surface return this.numInteriorRing(); else return PROCESSING_ERROR; }
Geometry InteriorRingN (Geometry this, Integer N)
{ if this instanceof Surface return this.interiorRing(N); else return PROCESSING_ERROR; }
Geometry Centroid (Geometry this)
{ if this instanceof (Surface or MultiSurface) return this.centroid(); else return PROCESSING_ERROR; }
Geometry PointOnSurface (Geometry this)
{ if this instanceof (Surface or MultiSurface) return this.pointOnSurface(); else return PROCESSING_ERROR; }
Integer NumPatches (Geometry this)
{ if this instanceof PolyhedralSurface return this.numPatches(); else return PROCESSING_ERROR; }
Geometry PatchN (Geometry this, Integer N)
{ if this instanceof PolyhedralSurface return this.patchN(N); else return PROCESSING_ERROR; }
Geometry BoundingPolygons (Geometry this, Geometry another)
{ if this instanceof PolyhedralSurface return this.boundingPolygons(another); else return PROCESSING_ERROR; }
GeometryCollection GCFromGeometryBag (GeometryBag this)
{ GeometryCollection gc = new GeometryCollection();
 for (int ix = 0; ix < this.bagSize(); ix++) {gc.add(this.get(ix));}
 return gc; }
Geometry GeometryBagFromGC (GeometryCollection this)
{ GeometryBag gb = new GeometryBag();
 for (int ix = 0; ix < this.numGeometries(); ix++) {gb.add(this.geometryN(ix));}
 return gb; }
}

6.9 Use Case – Restricting access to geospatial data
6.10 Use Case – Restricting access based on user location

7. Requirements for GeoXACML 3.0 Core

7.1 Introduction (informative)

The GeoXACML 3.0 Core is defined as an extension to OASIS eXtensible Access Control Markup Language (XACML) Version 3.0. In that regard, a GeoXACML 3.0 Core implementation must be fully compliant with the XACML 3.0 specification; in particular all data types and functions marked “mandatory” must be supported.

But because typical use cases for GeoXACML involve deriving authorization decisions based on XML encoded resources, e.g. OWS POST requests or OWS responses such as WFS feature collections, the GeoXACML 3.0 Core compliance will require implementing the one data type urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression marked “optional” in XACML and all functions related to that data type.

By introducing the data type urn:ogc:def:dataType:geoxacml:3.0:geometry, GeoXACML 3.0 Core must define functions on that data type which follow the procedural semantics of the XACML 3.0 standard. In particular the functions to marshall/unmarshall any data type to/from a string representation must be supported. Therefore, GeoXACML 3.0 Core must define the functions

urn:ogc:def:function:geoxacml:3.0:string-from-geometry and
urn:ogc:def:function:geoxacml:3.0:geometry-from-string. In addition, matching, test, bag and set functions defined by OASIS eXtensible Access Control Markup Language (XACML) Version 3.0 must be supported for the geometry data type.

In order for GeoXACML 3.0 Core to support spatial indexing of policies by target matching, the set of condition functions must be extended. According to OASIS eXtensible Access Control Markup Language (XACML) Version 3.0 the function signature must be two non-bag parameters with a boolean result. According to the geometry model and functions adopted from SF, functions to testing topological relations such as geometry-equals, geometry-within, etc. qualify. It is important for GeoXACML 3.0 Core to nominate all qualifying functions to be condition functions in the sense of OASIS eXtensible Access Control Markup Language (XACML) Version 3.0.

In order to accommodate the ease of implementation, GeoXACML 3.0 Core defines three implementation levels: basic, advanced and analysis. The basic implementation requires that the implementation supports the topological test functions. The advanced implementation basically\(^3\) supports all functions on geometry from ISO 19125 that are

\(^3\) Please see the normative section for the mandatory set of functions defined for the basic and the advanced conformance class.
labeled “query”. The analysis implementation conformance class requires that the implementation supports all functions on geometry from ISO 19125 and in addition the bag and set functions defined in XACML. Because the geometry data type marshalling / unmarshalling function is adopted from OASIS eXtensible Access Control Markup Language (XACML) Version 3.0, both conformance classes must support these functions.

Because GeoXACML 3.0 Core leverages the extension points as identified by OASIS eXtensible Access Control Markup Language (XACML) Version 3.0, all GeoXACML 3.0 policy instance documents are XACML 3.0 Schema compliant. Also, the XACML 3.0 Schema definitions for encoding the Authorization Decision Request and Response are sufficient for GeoXACML 3.0. Therefore, GeoXACML must not define compliance tests for policy, request and response instance documents.

In order to provide an overview, the following “quasi” UML diagram illustrates the relationships that GeoXACML is using / establishing.

7.2 Common Requirements Class

This section defines requirements that are common to other requirements.
7.3 Requirements Class: Specification

The standardization target for this requirements class is specification.

**Req 1** GeoXACML 3.0 urn prefix for function

GeoXACML 3.0 Core defines a non resolvable urn base identifier for functions

`urn:ogc:def:function:geoxacml:3.0`

**Req 2** GeoXACML 3.0 urn prefix for data type

GeoXACML 3.0 Core defines a non resolvable urn base identifier for functions

`urn:ogc:def:dataType:geoxacml:3.0`

**Req 3** OASIS eXtensible Access Control Markup Language (XACML) Version 3.0

GeoXACML 3.0 Core SHALL adopt all OASIS eXtensible Access Control Markup Language (XACML) Version 3.0 as defined in the section 10 “Conformance”.

GeoXACML 3.0 Core SHALL adopt OASIS eXtensible Access Control Markup Language (XACML) Version 3.0 data type

`urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression` marked “O” (optional) in section10.2.7.

GeoXACML 3.0 Core SHALL adopt OASIS eXtensible Access Control Markup Language (XACML) Version 3.0 functions for data type

`urn:oasis:names:tc:xacml:3.0:data-type:xpathExpression` marked “O” (optional) in section10.2.8.

**Req 4** XACML 3.0 Schema

GeoXACML 3.0 Core shall adopt the XACML 3.0 Schema.

**Req 5** Data type Geometry

GeoXACML 3.0 Core defines the following XACML compliant identifier for the data type Geometry as defined in ISO 19125:

`urn:ogc:def:dataType:geoxacml:3.0:geometry`

**Req 6** Geometry encoding using WKT

An instance of data type Geometry SHALL be represented as a string value of an XACML `<AttributeValue>` element where the DataType attribute is set to the value

`urn:ogc:def:dataType:geoxacml:3.0:geometry`

The encoding of the enclosed geometry SHALL be compliant to the definition of Well-Known-Text is ISO 19125.
Req 7  Geometry encoding using XML

An instance of data type Geometry SHALL be represented as a direct child of an XACML
<AttributeValue> element where the DataType attribute is set to the value
urn:ogc:def:dataType:geoxacml:3.0:geometry.

The encoding of the enclosed geometry SHALL be compliant to the definition of a
gеometry encoding extension to this standard.

Req 8  Data type GeometryBag

A GeometryBag shall be an XACML bag with the data type
urn:ogc:def:dataType:geoxacml:3.0:geometry

Req 9  Function GeometryType

This function SHALL have the signature GeometryType(this:Geometry):String
and the identifier as urn:ogc:def:function:geoxacml:3.0:geometry-type.

This function SHALL be compliant to GeometryType():String as defined in ISO
19125.

Note: This function returns the class name of the concrete geometry subtype according to
the UML diagram as defined in ISO 19125.

Req 10  Function Dimension

This function SHALL have the signature Dimension(this:Geometry):Integer
and the identifier as urn:ogc:def:function:geoxacml:3.0:geometry-dimension.

This function SHALL be compliant to Dimension():Integer as defined in ISO
19125.

Req 11  Function SRID

This function SHALL have the signature Dimension(this:Geometry):Integer
and the identifier as urn:ogc:def:function:geoxacml:3.0:geometry-srid.

This function SHALL be compliant to SRID():Integer\(^4\) as defined in ISO 19125.

Req 12  Function Envelope

\(^4\) This is most likely going to change to SRID():String
This function SHALL have the signature `Envelope(this:Geometry):Geometry` and the identifier as `urn:ogc:def:function:geoxacml:3.0:geometry-envelope`.

This function SHALL be compliant to `Envelope():Geometry` as defined in ISO 19125.

Req 13  Function AsText

This function SHALL have the signature `AsText(this:Geometry):String` and the identifier as `urn:ogc:def:function:geoxacml:3.0:string-from-geometry`.

This function SHALL be compliant to `AsText():String` as defined in ISO 19125.

Req 14  Function AsBinary

GeoXACML 3.0 Core DOES NOT define a XACML 3.0 corresponding representation and identifier for the function `AsBinary():Binary` as defined in ISO 19125.

Req 15  Function IsEmpty

This function SHALL have the signature `IsEmpty(this:Geometry):Boolean` and the identifier as `urn:ogc:def:function:geoxacml:3.0:geometry-is-empty`.

This function SHALL be compliant to `IsEmpty():Integer` as defined in ISO 19125.

Req 16  Function IsSimple

This function SHALL have the signature `IsSimple(this:Geometry):Boolean` and the identifier as `urn:ogc:def:function:geoxacml:3.0:geometry-is-simple`.

This function SHALL be compliant to `IsSimple():Integer` as defined in ISO 19125.

Req 17  Function Is3D

This function SHALL have the signature `Is3D(this:Geometry):Boolean` and the identifier as `urn:ogc:def:function:geoxacml:3.0:geometry-is-3d`.

This function SHALL be compliant to `Is3D():Boolean` as defined in ISO 19125.

Req 18  Function IsMeasured

This function SHALL have the signature `IsMeasured(this:Geometry):Boolean` and the identifier as `urn:ogc:def:function:geoxacml:3.0:geometry-is-measured`.
This function SHALL be compliant to \texttt{IsMeasured()}:Integer as defined in ISO 19125.

\textbf{Req 19}   Function Boundary

This function SHALL have the signature \texttt{Boundary(Geometry):Geometry} and the identifier as \texttt{urn:ogc:def:function:geoxacml:3.0:geometry-boundary}.

This function SHALL be compliant to \texttt{Boundary()}:Geometry as defined in ISO 19125.

\textbf{Req 20}   Function Equals

This function SHALL have the signature \texttt{Equals(this:Geometry,another:Geometry):Boolean} and the identifier as \texttt{urn:ogc:def:function:geoxacml:3.0:geometry-equals}.

This function SHALL be compliant to \texttt{equals(Geometry)}:Integer as defined in ISO 19125.

\textbf{Req 21}   Function Disjoint

This function SHALL have the signature \texttt{Disjoint(this:Geometry,another:Geometry):Boolean} and the identifier as \texttt{urn:ogc:def:function:geoxacml:3.0:geometry-disjoint}.

This function SHALL be compliant to \texttt{Disjoint(Geometry)}:Integer as defined in ISO 19125.

\textbf{Req 22}   Function Intersects

This function SHALL have the signature \texttt{Intersects(this:Geometry,another:Geometry):Boolean} and the identifier as \texttt{urn:ogc:def:function:geoxacml:3.0:geometry-intersects}.

This function SHALL be compliant to \texttt{Intersects(Geometry)}:Integer as defined in ISO 19125.

\textbf{Req 23}   Function Touches

This function SHALL have the signature \texttt{Touches(this:Geometry,another:Geometry):Boolean} and the identifier as \texttt{urn:ogc:def:function:geoxacml:3.0:geometry-touches}.

This function SHALL be compliant to \texttt{Touches(Geometry)}:Integer as defined in ISO 19125.
Req 24  Function Crosses

This function SHALL have the signature
Crosses(this:Geometry,another:Geometry):Boolean and the identifier as
urn:ogc:def:function:geoxacml:3.0:geometry-crosses.

This function SHALL be compliant to Crosses(Geometry):Integer as defined in
ISO 19125.

Req 25  Function Within

This function SHALL have the signature
Within(this:Geometry,another:Geometry):Boolean and the identifier as
urn:ogc:def:function:geoxacml:3.0:geometry-within.

This function SHALL be compliant to Within(Geometry):Integer as defined in ISO
19125.

Req 26  Function Contains

This function SHALL have the signature
Contains(this:Geometry,another:Geometry):Boolean and the identifier as
urn:ogc:def:function:geoxacml:3.0:geometry-contains.

This function SHALL be compliant to Contains(Geometry):Integer as defined in
ISO 19125.

Req 27  Function Overlaps

This function SHALL have the signature
Overlaps(this:Geometry,another:Geometry):Boolean and the identifier as
urn:ogc:def:function:geoxacml:3.0:geometry-overlaps.

This function SHALL be compliant to Overlaps(Geometry):Integer as defined in
ISO 19125.

Req 28  Function Relate

This function SHALL have the signature
Relate(this:Geometry,another:Geometry,String):Boolean and the
identifier as urn:ogc:def:function:geoxacml:3.0:geometry-relate.

This function SHALL be compliant to Relate(Geometry,String):Integer as
defined in ISO 19125.

Req 29  Function LocateAlong
This function SHALL have the signature

This function SHALL be compliant to LocateAlong(Geometry, Double):Geometry as defined in ISO 19125.

Req 30 Function LocateBetween

This function SHALL have the signature
LocateBetween(this:Geometry, mStart:Double, mEnd:Double):Geometry and the identifier as urn:ogc:def:function:geoxacml:3.0:geometry-locate-between.

This function SHALL be compliant to LocateBetween(Double, Double):Geometry as defined in ISO 19125.

Req 31 Function Distance

This function SHALL have the signature
Distance(this:Geometry, another:Geometry):Double and the identifier as urn:ogc:def:function:geoxacml:3.0:geometry-distance.

This function SHALL be compliant to Distance(another:Geometry):Double as defined in ISO 19125.

Req 32 Function IsWithinDistance

This function SHALL have the signature
IsWithinDistance(this:Geometry, another:Geometry, d:Double):Boolean and the identifier as urn:ogc:def:function:geoxacml:3.0:geometry-is-within-distance.

This function SHALL be compliant to IsWithinDistance():Double as defined in ISO 19125.

Req 33 Function Buffer

This function SHALL have the signature

This function SHALL be compliant to Buffer(Double):Geometry as defined in ISO 19125.

Req 34 Function ConvexHull
This function SHALL have the signature \( \text{ConvexHull}(\text{this:Geometry}) : \text{Geometry} \)
and the identifier as \( \text{urn:ogc:def:function:geoxacml:3.0:geometry-convex-hull} \).

This function SHALL be compliant to \( \text{ConvexHull}() : \text{Geometry} \) as defined in ISO 19125.

Req 35  Function Intersection

This function SHALL have the signature
\( \text{Intersection}(\text{this:Geometry,another:Geometry}) : \text{Geometry} \)
and the identifier as \( \text{urn:ogc:def:function:geoxacml:3.0:geometry-intersection} \).

This function SHALL be compliant to \( \text{Intersection(Geometry)} : \text{Geometry} \) as defined in ISO 19125.

Req 36  Function Union

This function SHALL have the signature
\( \text{Union}(\text{this:Geometry,another:Geometry}) : \text{Geometry} \)
and the identifier as \( \text{urn:ogc:def:function:geoxacml:3.0:geometry-union} \).

This function SHALL be compliant to \( \text{Union(Geometry)} : \text{Geometry} \) as defined in ISO 19125.

Req 37  Function Difference

This function SHALL have the signature
\( \text{Difference}(\text{this:Geometry,another:Geometry}) : \text{Geometry} \)
and the identifier as \( \text{urn:ogc:def:function:geoxacml:3.0:geometry-difference}. \)

This function SHALL be compliant to \( \text{Difference(Geometry)} : \text{Geometry} \) as defined in ISO 19125.

Req 38  Function SymDifference

This function SHALL have the signature
\( \text{SymDifference}(\text{this:Geometry,another:Geometry}) : \text{Geometry} \)
and the identifier as \( \text{urn:ogc:def:function:geoxacml:3.0:geometry-sym-difference}. \)

This function SHALL be compliant to \( \text{SymDifference(Geometry)} : \text{Geometry} \) as defined in ISO 19125.

Req 39  Function NumGeometries
This function SHALL have the signature
NumGeometries(this:GeometryCollection):Integer and the identifier as

This function SHALL be compliant to NumGeometries():Integer as defined in ISO
19125.

Req 40  Function GeometryN

This function SHALL have the signature
Geometry(this:GeometryBag,N:Integer):Geometry and the identifier as
urn:ogc:def:function:geoxacml:3.0:geometry-n.

This function SHALL be compliant to GeometryN(N:Integer):Geometry as
defined in ISO 19125.

Req 41  Function Length

This function SHALL have the signature Length(this:Geometry):Double and the
identifier as urn:ogc:def:function:geoxacml:3.0:geometry-length.

This function SHALL be compliant to Length():Double as defined in ISO 19125.

Req 42  Function X

This function SHALL have the signature X(this:Geometry):Double and the
identifier as urn:ogc:def:function:geoxacml:3.0:geometry-x.

This function SHALL be compliant to X():Double as defined in ISO 19125.

Req 43  Function Y

This function SHALL have the signature Y(this:Geometry):Double and the
identifier as urn:ogc:def:function:geoxacml:3.0:geometry-y.

This function SHALL be compliant to Y():Double as defined in ISO 19125.

Req 44  Function Z

This function SHALL have the signature Z(this:Geometry):Double and the
identifier as urn:ogc:def:function:geoxacml:3.0:geometry-z.

This function SHALL be compliant to Z():Double as defined in ISO 19125.

Req 45  Function M
This function SHALL have the signature \( M(\text{this:Geometry}):\text{Double} \) and the identifier as \( \text{urn:ogc:def:function:geoxacml:3.0:geometry-m} \).

This function SHALL be compliant to \( M():\text{Double} \) as defined in ISO 19125.

**Req 46** Function StartPoint

This function SHALL have the signature \( \text{StartPoint(Geometry):Geometry} \) and the identifier as \( \text{urn:ogc:def:function:geoxacml:3.0:geometry-start-point} \).

This function SHALL be compliant to \( \text{StartPoint()}:\text{Geometry} \) as defined in ISO 19125.

**Req 47** Function EndPoint

This function SHALL have the signature \( \text{EndPoint(this:Geometry):Geometry} \) and the identifier as \( \text{urn:ogc:def:function:geoxacml:3.0:geometry-end-point} \).

This function SHALL be compliant to \( \text{EndPoint()}:\text{Geometry} \) as defined in ISO 19125.

**Req 48** Function IsClosed

This function SHALL have the signature \( \text{IsClosed(this:Geometry):Boolean} \) and the identifier as \( \text{urn:ogc:def:function:geoxacml:3.0:geometry-is-closed} \).

This function SHALL be compliant to \( \text{IsClosed()}:\text{Integer} \) as defined in ISO 19125.

**Req 49** Function IsValid

This function SHALL have the signature \( \text{IsValid(this:Geometry):Boolean} \) and the identifier as \( \text{urn:ogc:def:function:geoxacml:3.0:geometry-is-valid} \).

This function SHALL be compliant to \( \text{IsValid()}:\text{Integer} \) as defined in ISO 19125.

**Req 50** Function IsRing

This function SHALL have the signature \( \text{IsRing(this:Geometry):Boolean} \) and the identifier as \( \text{urn:ogc:def:function:geoxacml:3.0:geometry-is-ring} \).

This function SHALL be compliant to \( \text{IsRing()}:\text{Integer} \) as defined in ISO 19125.

**Req 51** Function NumPoints
This function SHALL have the signature `NumPoints(this:Geometry):Integer` and the identifier as `urn:ogc:def:function:geoxacml:3.0:geometry-num-points`.

This function SHALL be compliant to `NumPoints():Integer` as defined in ISO 19125.

**Req 52** Function `PointN`

This function SHALL have the signature `PointN(this:Geometry,N:Integer):Geometry` and the identifier as `urn:ogc:def:function:geoxacml:3.0:geometry-point-n`.

This function SHALL be compliant to `PointN(Integer):Point` as defined in ISO 19125.

**Req 53** Function `Area`

This function SHALL have the signature `Area(this:Geometry):Double` and the identifier as `urn:ogc:def:function:geoxacml:3.0:geometry-area`.

This function SHALL be compliant to `Area():Double` as defined in ISO 19125.

**Req 54** Function `Centroid`

This function SHALL have the signature `Centroid(this:Geometry):Geometry` and the identifier as `urn:ogc:def:function:geoxacml:3.0:geometry-centroid`.

This function SHALL be compliant to `Centroid():Point` as defined in ISO 19125.

**Req 55** Function `PointOnSurface`

This function SHALL have the signature `PointOnSurface(this:Geometry):Geometry` and the identifier as `urn:ogc:def:function:geoxacml:3.0:geometry-point-on-surface`.

This function SHALL be compliant to `PointOnSurface():Point` as defined in ISO 19125.

**Req 56** Function `ExteriorRing`

This function SHALL have the signature `ExteriorRing(this:Geometry):Geometry` and the identifier as `urn:ogc:def:function:geoxacml:3.0:geometry-exterior-ring`.
This function SHALL be compliant to ExteriorRing():LineString as defined in ISO 19125.

**Req 57** Function NumInteriorRing

This function SHALL have the signature

This function SHALL be compliant to NumInteriorRing():Integer as defined in ISO 19125.

**Req 58** Function InteriorRingN

This function SHALL have the signature

This function SHALL be compliant to InteriorRingN(Integer):LineString as defined in ISO 19125.

**Req 59** Function NumPatches

This function SHALL have the signature NumPatches(this:Geometry):Integer and the identifier as urn:ogc:def:function:geoxacml:3.0:geometry-num-patches.

This function SHALL be compliant to NumPatches():Integer as defined in ISO 19125.

**Req 60** Function PatchN

This function SHALL have the signature PatchN(this:Geometry,N:Integer):Geometry and the identifier as urn:ogc:def:function:geoxacml:3.0:geometry-patch-n.

This function SHALL be compliant to PatchN(Integer):Geometry as defined in ISO 19125.

**Req 61** Function BoundingPolygons

This function SHALL have the signature BoundingPolygons(this:Geometry,p:Geometry):Geometry and the identifier as urn:ogc:def:function:geoxacml:3.0:geometry-bounding-polygons.
This function SHALL be compliant to
\texttt{BoundingPolygons(Polygon):MultiPolygon} as defined in ISO 19125.

\textbf{Req 62} Function \texttt{GeometryFromString}

This function SHALL have the signature
\texttt{GeometryFromString(wkt:String):Geometry} and the identifier as
\texttt{urn:ogc:def:function:geoxacml:3.0:geometry-from-string}.

This function SHALL accept a parameter of type \texttt{String} which value is compliant to the
\texttt{WKT} representation as defined in ISO 19125.

This function SHALL return a geometry instance according to the \texttt{WKT} representation of
the argument.

\textbf{Req 63} Function \texttt{GeometryOneAndOnly}

This function SHALL have the signature
\texttt{GeometryOneAndOnly(bag:GeometryBag):Geometry} and the identifier as
\texttt{urn:ogc:def:function:geoxacml:3.0:geometry-one-and-only}.

This function SHALL return the only value in the bag.

This function SHALL return “Immediate” if the bag does not have one and only one
value.

\textbf{Req 64} Function \texttt{GeometryBagSize}

This function SHALL have the signature
\texttt{GeometryBagSize(bag:GeometryBag):Integer} and the identifier as
\texttt{urn:ogc:def:function:geoxacml:3.0:geometry-bag-size}.

This function SHALL return the number of values of type \texttt{Geometry} in the bag.

\textbf{Req 65} Function \texttt{GeometryIsIn}

This function SHALL have the signature
\texttt{GeometryIsIn(g:Geometry,bag:GeometryBag):Boolean} and the identifier as
\texttt{urn:ogc:def:function:geoxacml:3.0:geometry-is-in}.

The function SHALL return “True” if and only if the first argument matches by the
\texttt{urn:ogc:def:function:geoxacml:3.0:geometry-equals} any value in the bag.

This function SHALL return “False” otherwise or if the argument is an empty bag.

\textbf{Req 66} Function \texttt{Bag}
This function SHALL have the signature \( \text{Bag}(\text{Geometry}^*) : \text{GeometryBag} \) and the identifier as urn:ogc:def:function:geoxacml:3.0:geometry-bag.

This function SHALL return a bag of values where each member is of type \( \text{Geometry} \).

This function SHALL return an empty bag (a bag with zero members) in case there is no argument value.

**Req 67  Function BagIntersection**

This function SHALL have the signature
\[
\text{BagIntersection}(\text{bag1:GeometryBag}, \text{bag2:GeometryBag}) : \text{GeometryBag}
\]
and the identifier as urn:ogc:def:function:geoxacml:3.0:geometry-bag-intersection.

This function SHALL return a bag of values of type \( \text{Geometry} \) such that it contains only elements that are common between the two bags. This is determined by using the function urn:ogc:def:function:geoxacml:3.0:geometry-equals.

No duplicates as determined by
urn:ogc:def:function:geoxacml:3.0:geometry-equals, SHALL exist in the result.

**Req 68  Function BagAtLeastOneMemberOf**

This function SHALL have the signature
\[
\text{BagAtLeastOneMemberOf}(\text{bag1:GeometryBag}, \text{bag2:GeometryBag}) : \text{Boolean}
\]
and the identifier as urn:ogc:def:function:geoxacml:3.0:geometry-bag-at-least-one-member-of.

This function SHALL return "True" if and only if at least one element of the first argument is contained in the second argument as determined by
urn:ogc:def:function:geoxacml:3.0:geometry-is-in.

**Req 69  Function BagUnion**

This function SHALL have the signature
\[
\text{BagUnion}(\text{bag1:GeometryBag}, \text{bag2:GeometryBag}) : \text{GeometryBag}
\]and the identifier as urn:ogc:def:function:geoxacml:3.0:geometry-bag-union.

This function SHALL return a bag of \( \text{Geometry} \) such that it contains all elements of all the argument bags. No duplicates, as determined by
urn:ogc:def:function:geoxacml:3.0:geometry-equals, SHALL exist in the result.

**Req 70  Function BagSubset**
This function SHALL have the signature
\texttt{BagSubset(bag1:GeometryBag,bag2:GeometryBag):Boolean} and the identifier
\texttt{urn:ogc:def:function:geoxacml:3.0:geometry-bag-subset}.

This function SHALL return "True" if and only if the first argument is a subset of the
second argument. Each argument SHALL be considered to have had its duplicates
removed, as determined by \texttt{urn:ogc:def:function:geoxacml:3.0:geometry-equals},
before the subset calculation.

\textbf{Req 71} Function \texttt{SetEquals}

This function SHALL have the signature
\texttt{SetEquals(bag1:GeometryBag,bag2:GeometryBag):Boolean} and the identifier
\texttt{urn:ogc:def:function:geoxacml:3.0:geometry-set-equals}.

This function SHALL return the result of applying
\texttt{urn:oasis:names:tc:xacml:1.0:function:and} to the application of
\texttt{urn:ogc:def:function:geoxacml:3.0:geometry-bag-subset} to the first and
second arguments and the application of
\texttt{urn:ogc:def:function:geoxacml:3.0:geometry-bag-subset} to the second
and first arguments.

\textbf{Req 72} Function \texttt{BagFromGeometryCollection}

This function SHALL have the signature
\texttt{BagFromGC(gc:GeometryCollection):GeometryBag} and the identifier as
\texttt{urn:ogc:def:function:geoxacml:3.0:geometry-bag-from-geometry-collection}.

This function SHALL return a bag of values of type \texttt{Geometry} by adding each geometry
of the \texttt{GeometryCollection} as a member of type \texttt{Geometry}.

\textbf{Req 73} Function \texttt{GeometryCollectionFromBag}

This function SHALL have the signature
\texttt{GCFromBag(GeometryBag):GeometryCollection} and the identifier as
\texttt{urn:ogc:def:function:geoxacml:3.0:geometry-collection-from-geometry-bag}.

This function SHALL return a \texttt{GeometryCollection} as defined in ISO 19125 by adding
each member of the bag as a geometry to the collection.

\textbf{Req 74} Function \texttt{GeometryFromBinary}

\textit{GeoXACML 3.0 Core \textbf{DOES NOT} define a XACML 3.0 corresponding representation
and identifier for this function.}
Annex A: Conformance Class Abstract Test Suite (Normative)

This Annex defines three implementation levels of GeoXACML enriched Policy Decision Point (GeoPDP). Each implementation level is represented by a conformance class that provides tests to ensure compliance of an implementation.

Conformance class: BASIC Implementation

A.1 Conformance testing XACML 3.0

Test that the implementation is compliant with XACML 3.0 according to Req 3.

Test that the implementation is able to process a GeoXACML 3.0 Policy, which is a valid XML document according to Req 4.

A.2 Conformance testing Data Type Geometry

Test that the implementation supports the creation of an <AttributeValue> with data type urn:ogc:def:dataType:geoxacml:3.0:geometry according to Req 5 and where the geometry is described as a value encoded using WKT according to Req 6.

A.3 Conformance testing Functions

Test that the following functions are supported by the implementation:

<table>
<thead>
<tr>
<th>Function URN</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-contains</td>
<td>Req 26</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-crosses</td>
<td>Req 24</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-disjoint</td>
<td>Req 21</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-equals</td>
<td>Req 20</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-intersects</td>
<td>Req 23</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-overlaps</td>
<td>Req 27</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-touches</td>
<td>Req 23</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-within</td>
<td>Req 25</td>
</tr>
</tbody>
</table>
A.4 Conformance testing Condition Functions

Test that the following functions are supported by the implementation are supported as XACML Condition Functions and can participate in matching inside the <Target> element:

<table>
<thead>
<tr>
<th>Function URN</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-contains</td>
<td>Req 26</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-crosses</td>
<td>Req 24</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-disjoint</td>
<td>Req 21</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-equals</td>
<td>Req 20</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-intersects</td>
<td>Req 23</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-overlaps</td>
<td>Req 27</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-touches</td>
<td>Req 23</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-within</td>
<td>Req 25</td>
</tr>
</tbody>
</table>

Conformance class: ADVANCED Implementation

This conformance class extends the conformance class “BASIC”.

Test that the implementation is supporting the conformance class “BASIC”.

A.5 Conformance testing Data Type Geometry

Test that the implementation supports the creation of a geometry from an <AttributeValue> with data http://www.w3.org/2001/XMLSchema#string according to Req 62.

A.6 Conformance testing Functions

Test that the following functions are supported by the implementation:

<table>
<thead>
<tr>
<th>Function URN</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-is-within-distance</td>
<td>Req 32</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-dimension</td>
<td>Req 10</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-type</td>
<td>Req 9</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-srid</td>
<td>Req 11</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-is-empty</td>
<td>Req 15</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-is-simple</td>
<td>Req 16</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-is-3d</td>
<td>Req 17</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-is-valid</td>
<td>Req 49</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-is-closed</td>
<td>Req 48</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-is-measured</td>
<td>Req 18</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-is-ring</td>
<td>Req 50</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-relate</td>
<td>Req 28</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-distance</td>
<td>Req 31</td>
</tr>
</tbody>
</table>
Conformance class: ANALYSIS Implementation

This conformance class extends the conformance class “ADVANCED”.

Test that the implementation is supporting the conformance class “ADVANCED”.

A.7 Conformance testing Data Type Geometry

Test that the implementation supports the creation of a bag with data type `urn:ogc:def:dataType:geoxacml:3.0:geometry` according to Req 8.

A.8 Conformance testing Functions

Test that the following functions are supported by the implementation:

<table>
<thead>
<tr>
<th>Function URN</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>urn:ogc:def:function:geoxacml:3.0:geometry-envelope</code></td>
<td>Req 12</td>
</tr>
<tr>
<td><code>urn:ogc:def:function:geoxacml:3.0:geometry-boundary</code></td>
<td>Req 19</td>
</tr>
<tr>
<td><code>urn:ogc:def:function:geoxacml:3.0:geometry-locate-along</code></td>
<td>Req 29</td>
</tr>
<tr>
<td><code>urn:ogc:def:function:geoxacml:3.0:geometry-locate-between</code></td>
<td>Req 30</td>
</tr>
<tr>
<td><code>urn:ogc:def:function:geoxacml:3.0:geometry-buffer</code></td>
<td>Req 33</td>
</tr>
<tr>
<td><code>urn:ogc:def:function:geoxacml:3.0:geometry-convex-hull</code></td>
<td>Req 34</td>
</tr>
<tr>
<td><code>urn:ogc:def:function:geoxacml:3.0:geometry-intersection</code></td>
<td>Req 35</td>
</tr>
<tr>
<td><code>urn:ogc:def:function:geoxacml:3.0:geometry-union</code></td>
<td>Req 36</td>
</tr>
<tr>
<td><code>urn:ogc:def:function:geoxacml:3.0:geometry-difference</code></td>
<td>Req 37</td>
</tr>
<tr>
<td><code>urn:ogc:def:function:geoxacml:3.0:geometry-sym-difference</code></td>
<td>Req 38</td>
</tr>
<tr>
<td><code>urn:ogc:def:function:geoxacml:3.0:geometry-num-geometries</code></td>
<td>Req 39</td>
</tr>
<tr>
<td><code>urn:ogc:def:function:geoxacml:3.0:geometry-n</code></td>
<td>Req 40</td>
</tr>
<tr>
<td><code>urn:ogc:def:function:geoxacml:3.0:geometry-start-point</code></td>
<td>Req 46</td>
</tr>
<tr>
<td><code>urn:ogc:def:function:geoxacml:3.0:geometry-num-points</code></td>
<td>Req 51</td>
</tr>
<tr>
<td><code>urn:ogc:def:function:geoxacml:3.0:geometry-end-point</code></td>
<td>Req 47</td>
</tr>
<tr>
<td><code>urn:ogc:def:function:geoxacml:3.0:geometry-point-n</code></td>
<td>Req 52</td>
</tr>
<tr>
<td><code>urn:ogc:def:function:geoxacml:3.0:geometry-exterior-ring</code></td>
<td>Req 56</td>
</tr>
<tr>
<td><code>urn:ogc:def:function:geoxacml:3.0:geometry-num-interior-ring</code></td>
<td>Req 57</td>
</tr>
<tr>
<td><code>urn:ogc:def:function:geoxacml:3.0:geometry-interior-ring-n</code></td>
<td>Req 58</td>
</tr>
<tr>
<td>Function Expression</td>
<td>Requirement</td>
</tr>
<tr>
<td>---------------------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-centroid</td>
<td>Req 54</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-point-on-surface</td>
<td>Req 55</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-patch-n</td>
<td>Req 60</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-num-patches</td>
<td>Req 59</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-bounding-polygons</td>
<td>Req 61</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-collection-from-geometry-bag</td>
<td>Req 73</td>
</tr>
<tr>
<td>urn:ogc:def:function:geoxacml:3.0:geometry-bag-from-geometry-collection</td>
<td>Req 72</td>
</tr>
</tbody>
</table>
Annex B: Conformance Test Files (informative)
Annex C: Bibliography

Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides: *Design Patterns: Elements of Reusable Object-Oriented Software*, ISBN 0-201-63361-2

Annex D: Document revision history

<table>
<thead>
<tr>
<th>Date</th>
<th>Release</th>
<th>Author</th>
<th>Paragraph modified</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2013-03-21</td>
<td>0.1</td>
<td>Andreas Matheus</td>
<td>All</td>
<td>Document created</td>
</tr>
<tr>
<td>2013-05-03</td>
<td>0.2</td>
<td>Andreas Matheus</td>
<td>All</td>
<td>Comments from Norman Brickman (MITRE) incorporated</td>
</tr>
<tr>
<td>2013-10-28</td>
<td>0.3</td>
<td>Andreas Matheus</td>
<td>All</td>
<td>Cleanup for Discussion Paper release</td>
</tr>
</tbody>
</table>