Open Geospatial Consortium

Submission Date: 2013-09-05

Approval Date: 2013-09-25

Publication Date: 2013-11-06

External identifier of this OGC® document: http://www.opengis.net/doc/DP/GEOXACML-CORE

Internal reference number of this OGC® document: OGC 13-100

Category: OGC® Publicly Available Discussion Paper

Editor: Andreas Matheus

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

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

Document type: OGC® Discussion Paper
Document stage: Draft OGC Standard
Document language: English

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

Contents

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:

Name	Affiliation
Andreas Matheus	andreas.matheus <at> unibw.de

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;

  - 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

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

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.

OASIS eXtensible Access Control Markup Language (XACML) Version 3.0, 22 January 2013, http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf

OASIS eXtensible Access Control Markup Language (XACML) Version 3.0 XML Schema, http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd

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

Policy Directives for Writing and Publishing OGC Standards: TC Decisions. OGC Document 06-135r7. http://www.opengis.net/doc/POL/STD

The Specification Model — A Standard for Modular specifications OGC Document 08-131r3. http://www.opengis.net/doc/POL/SPEC

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 v B = ¬(¬A ^ ¬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

3D	Three Dimensional
AD	Authorization Decision
ADR	Authorization Decision Request
GeoXACML	Geospatial eXtensible Access Control Markup Language
ISO	International Organization for Standardization
OASIS	Organization for the Advancement of Structured Information Standards
OGC	Open Geospatial Consortium
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
UML	Unified Modeling Language
XACML	eXtensible Access Control Markup Language
XML	Extensible Markup Language
XSD	W3C XML Schema Definition Language

5.2 Document presentation of the specification

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 sub-trees, 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.

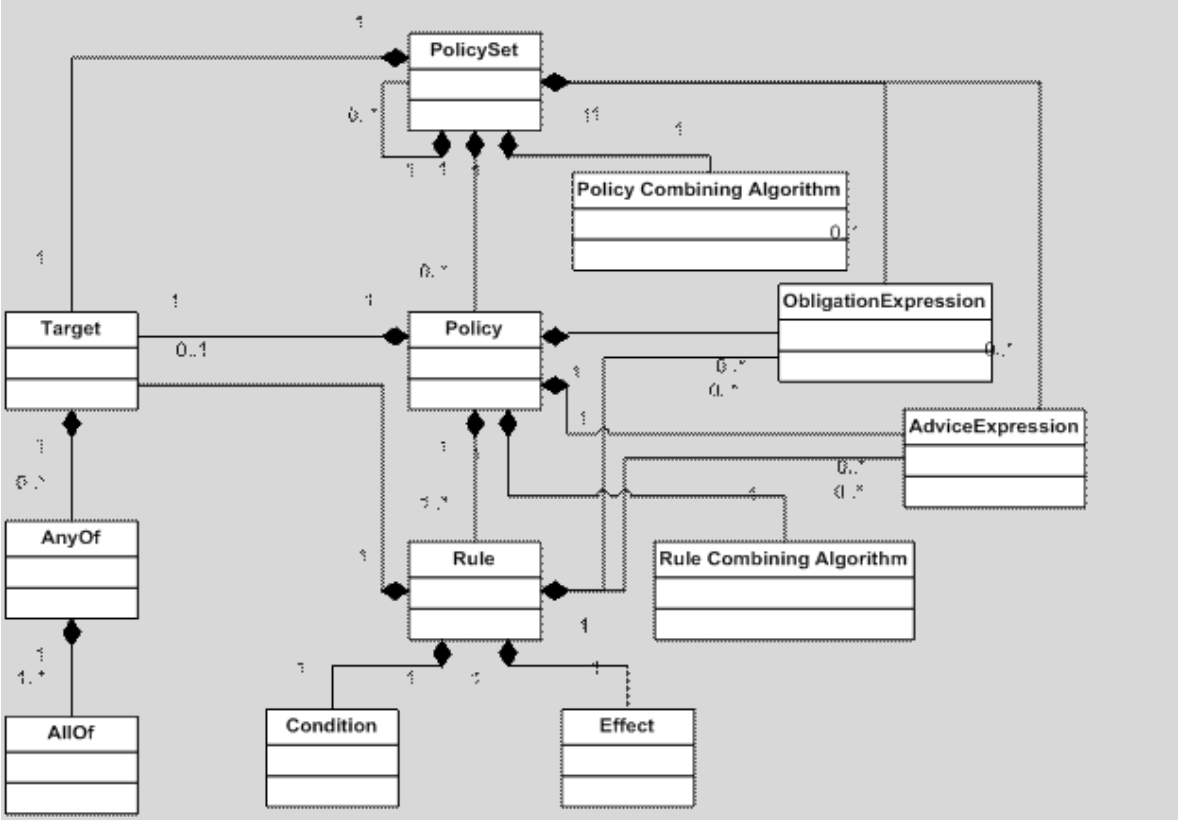

Figure 1 — XACML Policy Language Model

As a consequence of this change to the <Target> matching, the XACML 3.0 matching is more comprehensive and therefore enables automtic 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 machting 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 alsways 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.

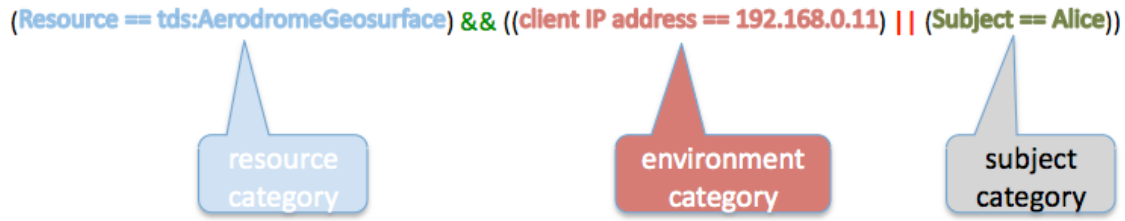

Figure 2 — Example <Target> matching supported by XACML 3.0 but not by XACML 2.0

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] A v B = ¬(¬A ^ ¬B)

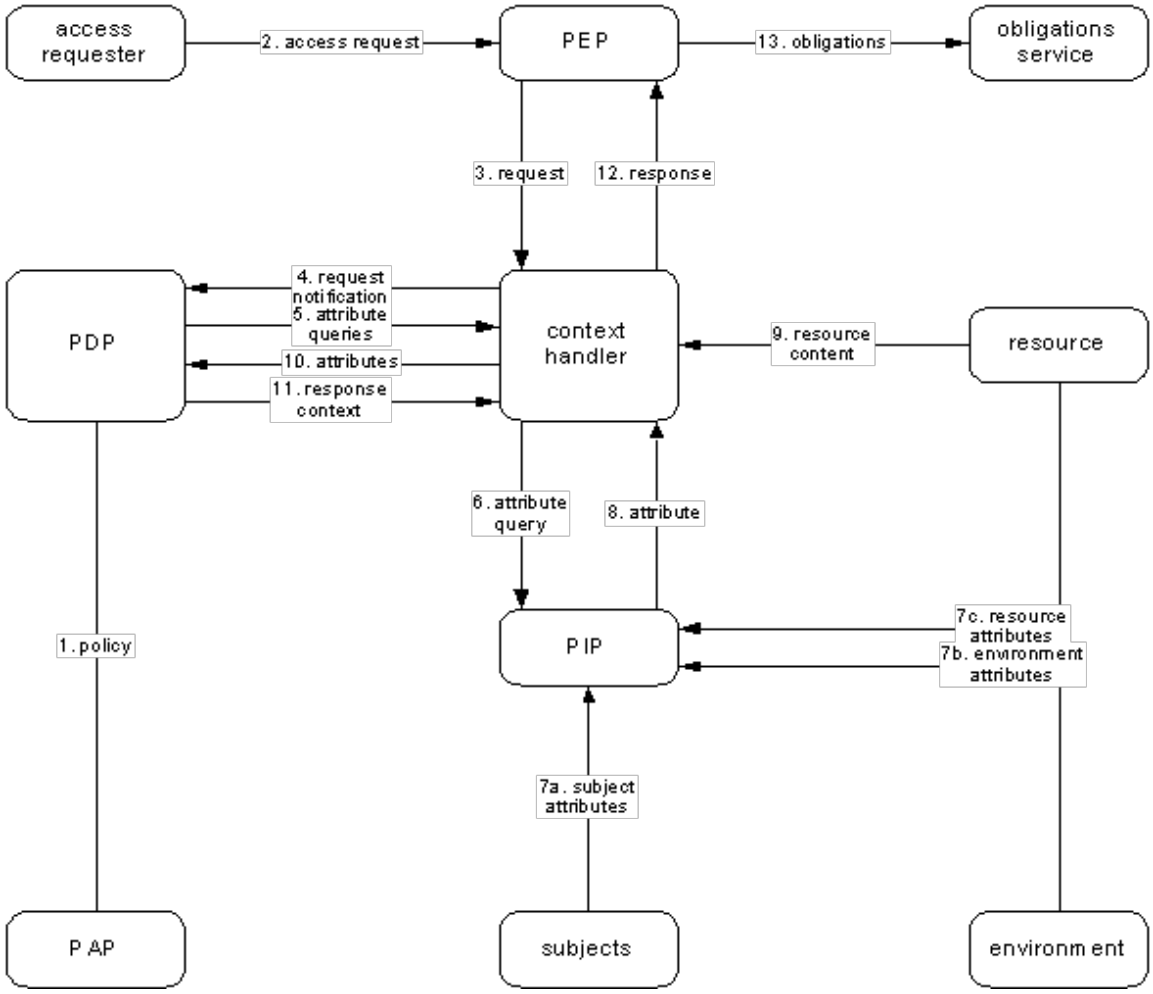

Figure 3 — XACML Information Flow Model [XACML 3.0, figure1]

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 `DataType, FunctionId` and `AttributeId` can be extended.

Please see the XACML schema definitions in http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd (OASIS eXtensible Access Control Markup Language (XACML) Version 3.0) for the XML format of the elements.

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 `DataType`. The XACML compliant URN is defined to be `urn:ogc:def:dataType:geoxacml:3.0:geometry`.

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.

```
<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.

```
<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.

```
<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 6 — XACML schema definition of the `<AttributeDesignatorType>` 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>>
```

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

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:  }
```

Line1: 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)
```

```
6:        return Permit;
7:  }
```

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 `Dimension(Geometry):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.

```
1:   class GeoXACML <<extends>> ISO19125
2:   {
3:     public static Integer Dimension (Geometry g)
4:     {
5:       if (g instanceOf Empty)
6:       {
7:         return new ProcessingError(„operation not defined on
     Empty“);
8:       }
9:       Try
10:      {
11:        return g.dimension();
12:      }
13:      catch (NullException)
14:      {
15:        return new ProcessingError(„operation error: NULL“);
16:      }
17:    }
18:  }
```

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.

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

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_AVANCED <<extends>> GeoXACML_BASIC

{

  #define IsWithinDifference ”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)
```

```
  { 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 matix)
  { 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”;
```

```
  #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-
bounding-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";

  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();}
  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.

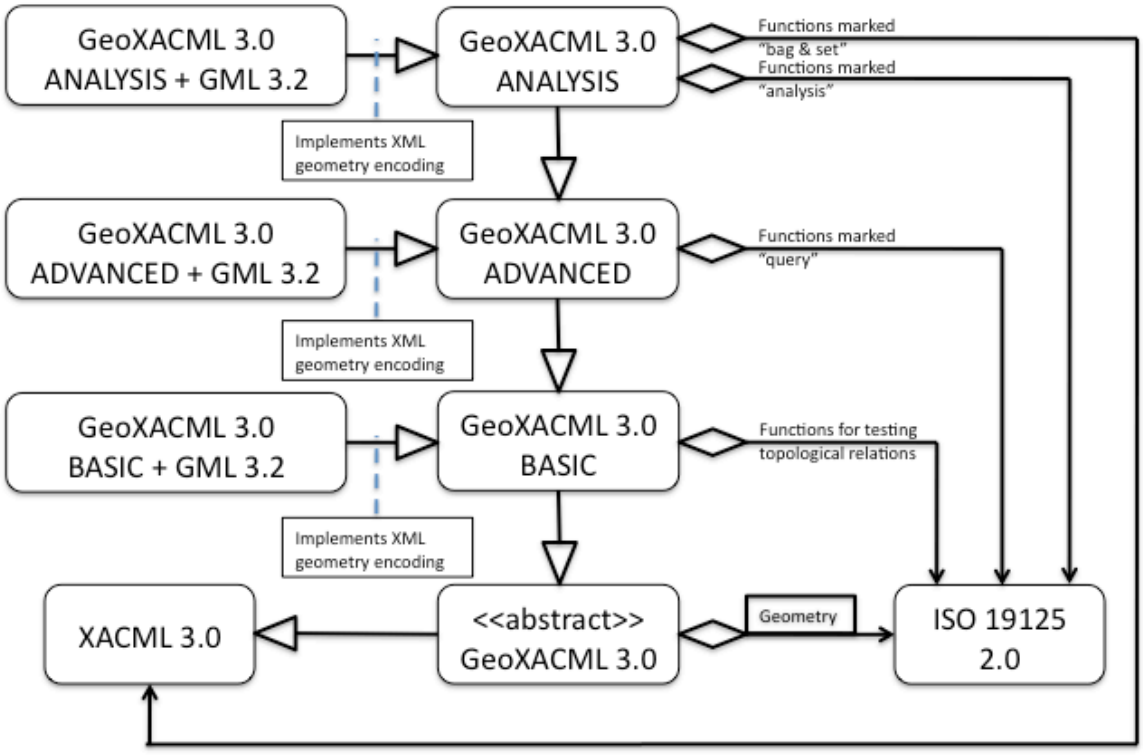

Figure 7 — GeoXACML relations illustrated in UML

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 geometry 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 `IsMeasured():Integer` as defined in ISO 19125.

Req 19 Function Boundary

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

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

Req 20 Function Equals

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

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

Req 21 Function Disjoint

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

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

Req 22 Function Intersects

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

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

Req 23 Function Touches

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

This function SHALL be compliant to `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 `LocateAlong(this:Geometry,another:Geometry,Double):Geometry` and the identifier as `urn:ogc:def:function:geoxacml:3.0:geometry-locate-along`.

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 `IsWitinDistance():Double` as defined in ISO 19125.

Req 33 Function Buffer

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

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

Req 34 Function ConvexHull

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

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

Req 35 Function Intersection

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

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

Req 36 Function Union

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

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

Req 37 Function Difference

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

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

Req 38 Function SymDifference

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

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

Req 39 Function NumGeometries

This function SHALL have the signature `NumGeometries(this:GeometryCollection):Integer` and the identifier as `urn:ogc:def:function:geoxacml:3.0:geometry-num-geometries`.

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(this:Geometry):Double` and the identifier as `urn:ogc:def:function:geoxacml:3.0:geometry-m`.

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

Req 46 Function StartPoint

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

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

Req 47 Function EndPoint

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

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

Req 48 Function IsClosed

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

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

Req 49 Function IsValid

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

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

Req 50 Function IsRing

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

This function SHALL be compliant to `IsRing():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 `NumInteriorRing(this:Geometry):Integer` and the identifier as `urn:ogc:def:function:geoxacml:3.0:geometry-num-interior-ring.`

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

Req 58 Function InteriorRingN

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

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
`BoundingPolygons(Polygon):MultiPolygon` as defined in ISO 19125.

Req 62 Function GeometryFromString

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

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

This function SHALL return a geometry instance according to the WKT representation of the argument.

Req 63 Function GeometryOneAndOnly

This function SHALL have the signature
`GeometryOneAndOnly(bag:GeometryBag):Geometry` and the identifier as
`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.

Req 64 Function GeometryBagSize

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

This function SHALL return the number of values of type `Geometry` in the bag.

Req 65 Function GeometryIsIn

This function SHALL have the signature
`GeometryIsIn(g:Geometry,bag:GeometryBag):Boolean` and the identifier as
`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 `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.

Req 66 Function Bag

This function SHALL have the signature `Bag(Geometry*):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 `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 `BagIntersection(bag1:GeometryBag,bag2:GeometryBag):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 `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 `BagAtLeastOneMemberOf(bag1:GeometryBag,bag2:GeometryBag):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 `BagUnion(bag1:GeometryBag,bag2:GeometryBag):GeometryBag` and the identifier as `urn:ogc:def:function:geoxacml:3.0:geometry-bag-union`.

This function SHALL return a bag of `Geoemtry` 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 `BagSubset(bag1:GeometryBag,bag2:GeometryBag):Boolean` and the identifier as `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 `urn:ogc:def:function:geoxacml:3.0:geometry-equals`, before the subset calculation.

Req 71 Function SetEquals

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

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

Req 72 Function BagFromGeometryCollection

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

This function SHALL return a bag of values of type `Geometry` by adding each geometry of the GeometryCollection as a member of type `Geometry`.

Req 73 Function GeometryCollectionFromBag

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

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

Req 74 Function GeometryFromBinary

GeoXACML 3.0 Core **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:

Function URN	Requirement
`urn:ogc:def:function:geoxacml:3.0:geometry-contains`	Req 26
`urn:ogc:def:function:geoxacml:3.0:geometry-crosses`	Req 24
`urn:ogc:def:function:geoxacml:3.0:geometry-disjoint`	Req 21
urn:ogc:def:function:geoxacml:3.0:geometry-equals	Req 20
`urn:ogc:def:function:geoxacml:3.0:geometry-intersects`	Req 23
urn:ogc:def:function:geoxacml:3.0:geometry-overlaps	Req 27
urn:ogc:def:function:geoxacml:3.0:geometry-touches	Req 23
urn:ogc:def:function:geoxacml:3.0:geometry-within	Req 25

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:

Function URN	Requirement
urn:ogc:def:function:geoxacml:3.0:geometry-contains	Req 26
urn:ogc:def:function:geoxacml:3.0:geometry-crosses	Req 24
urn:ogc:def:function:geoxacml:3.0:geometry-disjoint	Req 21
urn:ogc:def:function:geoxacml:3.0:geometry-equals	Req 20
urn:ogc:def:function:geoxacml:3.0:geometry-intersects	Req 23
urn:ogc:def:function:geoxacml:3.0:geometry-overlaps	Req 27
urn:ogc:def:function:geoxacml:3.0:geometry-touches	Req 23
urn:ogc:def:function:geoxacml:3.0:geometry-within	Req 25

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:

Function URN	Requirement
urn:ogc:def:function:geoxacml:3.0:geometry-is-within-distance	Req 32
urn:ogc:def:function:geoxacml:3.0:geometry-dimension	Req 10
urn:ogc:def:function:geoxacml:3.0:geometry-type	Req 9
urn:ogc:def:function:geoxacml:3.0:geometry-srid	Req 11
urn:ogc:def:function:geoxacml:3.0:geometry-is-empty	Req 15
urn:ogc:def:function:geoxacml:3.0:geometry-is-simple	Req 16
urn:ogc:def:function:geoxacml:3.0:geometry-is-3d	Req 17
urn:ogc:def:function:geoxacml:3.0:geometry-is-valid	Req 49
urn:ogc:def:function:geoxacml:3.0:geometry-is-closed	Req 48
urn:ogc:def:function:geoxacml:3.0:geometry-is-measured	Req 18
urn:ogc:def:function:geoxacml:3.0:geometry-is-ring	Req 50
urn:ogc:def:function:geoxacml:3.0:geometry-relate	Req 28
urn:ogc:def:function:geoxacml:3.0:geometry-distance	Req 31

urn:ogc:def:function:geoxacml:3.0:geometry-length	Req 41
urn:ogc:def:function:geoxacml:3.0:geometry-area	Req 53
urn:ogc:def:function:geoxacml:3.0:geometry-x	Req 42
urn:ogc:def:function:geoxacml:3.0:geometry-y	Req 43
urn:ogc:def:function:geoxacml:3.0:geometry-z	Req 44
urn:ogc:def:function:geoxacml:3.0:geometry-m	Req 45

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:

Function URN	Requirement
`urn:ogc:def:function:geoxacml:3.0:geometry-envelope`	Req 12
urn:ogc:def:function:geoxacml:3.0:geometry-boundary	Req 19
urn:ogc:def:function:geoxacml:3.0:geometry-locate-along	Req 29
urn:ogc:def:function:geoxacml:3.0:geometry-locate-between	Req 30
urn:ogc:def:function:geoxacml:3.0:geometry-buffer	Req 33
urn:ogc:def:function:geoxacml:3.0:geometry-convex-hull	Req 34
urn:ogc:def:function:geoxacml:3.0:geometry-intersection	Req 35
urn:ogc:def:function:geoxacml:3.0:geometry-union	Req 36
urn:ogc:def:function:geoxacml:3.0:geometry-difference	Req 37
urn:ogc:def:function:geoxacml:3.0:geometry-sym-difference	Req 38
urn:ogc:def:function:geoxacml:3.0:geometry-num-geometries	Req 39
urn:ogc:def:function:geoxacml:3.0:geometry-n	Req 40
urn:ogc:def:function:geoxacml:3.0:geometry-start-point	Req 46
urn:ogc:def:function:geoxacml:3.0:geometry-num-points	Req 51
urn:ogc:def:function:geoxacml:3.0:geometry-end-point	Req 47
urn:ogc:def:function:geoxacml:3.0:geometry-point-n	Req 52
urn:ogc:def:function:geoxacml:3.0:geometry-exterior-ring	Req 56
urn:ogc:def:function:geoxacml:3.0:geometry-num-interior-ring	Req 57
urn:ogc:def:function:geoxacml:3.0:geometry-interior-ring-n	Req 58

urn:ogc:def:function:geoxacml:3.0:geometry-centroid	Req 54
urn:ogc:def:function:geoxacml:3.0:geometry-point-on-surface	Req 55
urn:ogc:def:function:geoxacml:3.0:geometry-patch-n	Req 60
urn:ogc:def:function:geoxacml:3.0:geometry-num-patches	Req 59
urn:ogc:def:function:geoxacml:3.0:geometry-bounding-polygons	Req 61
urn:ogc:def:function:geoxacml:3.0:geometry-collection-from-geometry-bag	Req 73
urn:ogc:def:function:geoxacml:3.0:geometry-bag-from-geometry-collection	Req 72

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

Date	Release	Author	Paragraph modified	Description
2013-03-21	0.1	Andreas Matheus	All	Document created
2013-05-03	0.2	Andreas Matheus	All	Comments from Norman Brickman (MITRE) incorporated
2013-10-28	0.3	Andreas Matheus	All	Cleanup for Discussion Paper release