Open Geospatial Consortium Inc.

Date: 2009-07-24

Reference number of this document: OGC 09-036r2

Version: 0.3.0

Category: OGC Public Engineering Report

Editor(s): Jan Herrmann
Andreas Matheus

OGC® OWS-6 GeoXACML Engineering Report

Warning

This document is not an OGC Standard. This document is an OGC Public Engineering Report created as a deliverable in an OGC Interoperability Initiative and is not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Engineering Report should not be referenced as required or mandatory technology in procurements.

Document type:	OGC Engineering Report
Document stage:	Approved for Public release
Document language:	English

Table of contents

Figures

Page

Tables

Page

i. Preface

This Engineering Report was prepared as a deliverable for the OGC Web Services, Phase 6 (OWS-6) initiative of the OGC Interoperability Program. This document presents the work completed with respect to the Geoprocessing Workflow (GPW) activities within OWS-6.

This Engineering Report describes and evaluates the use of XACML and GeoXACML in OGC Web Service based IT infrastructures.

Suggested additions, changes, and comments on this report are welcome and encouraged. Such suggestions may be submitted by email message or by making suggested changes in an edited copy of this document.

ii. Document Terms and Definitions

This document uses the standard terms defined in sub-clause 5.3 of [OGC 05-008], 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.

iii. Submission and Contribution Contact Points

All questions regarding this document should be directed to the editor or the contributors:

Name	Organization
Jan Herrmann	Technische Universität München
Andreas Matheus	Universität der Bundeswehr München

iv. Revision History

Date	Release	Editor	Primary clauses modified	Description
2009/04/17	0.1.1	JH, AM	All	initial writing
2009/04/20	0.1.2	JH, AM	throughout	corrections after review
2009/05/04	0.1.3	JH, AM	throughout	minor editorial and formatting corrections
2009/7/11	0.3.0	Carl Reed	Various	Prepare for publication as public ER

v. Changes to the OGC Abstract Specification

The OpenGIS® Abstract Specification does not require changes to accommodate the technical contents of this document. Even though there are potential change requests to other OGC specification and to some XACML related OASIS specifications (c.p. 10.2). Change Requests to the corresponding standards shall be developed based on this document in future.

vi. Foreword

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 Inc. 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.

OWS-6 Testbed

OWS testbeds are part of OGC's Interoperability Program, a global, hands-on and collaborative prototyping program designed to rapidly develop, test and deliver Engineering Reports into OGC's Specification Program, where they are formalized for public release. In OGC's Interoperability Initiatives, international teams of technology providers work together to solve specific geoprocessing interoperability problems posed by the Initiative's sponsoring organizations. OGC Interoperability Initiatives include test beds, pilot projects, interoperability experiments and interoperability support services - all designed to encourage rapid development, testing, validation and adoption of OGC standards.

In April 2008, the OGC issued a call for sponsors for an OGC Web Services, Phase 6 (OWS-6) Testbed activity. The activity completed in June 2009. There is a series of on-line demonstrations available here:

http://www.opengeospatial.org/pub/www/ows6/index.html

The OWS-6 sponsors are organizations seeking open standards to address their urgent interoperability requirements. After analyzing their requirements, the OGC Interoperability Team recommended to the sponsors that the content of the OWS-6 initiative be organized around the following threads:

1. Sensor Web Enablement (SWE)
2. Geo Processing Workflow (GPW)
3. Aeronautical Information Management (AIM)
4. Decision Support Services (DSS)
5. Compliance Testing (CITE)

Additional background on these threads and the Request for Quotation / Call For Participation (RFQ/CFP) issued by OGC can be found at:

http://www.opengeospatial.org/projects/initiatives/ows-6.

The OWS-6 sponsoring organizations were:

- U.S. National Geospatial-Intelligence Agency (NGA)
- Joint Program Executive Office for Chemical and Biological Defense (JPEO-CBD)
- GeoConnections - Natural Resources Canada

- U.S. Federal Aviation Agency (FAA)
- EUROCONTROL
- EADS Defence and Communications Systems
- US Geological Survey
- Lockheed Martin
- BAE Systems
- ERDAS, Inc.

The OWS-6 participating organizations were:

52North, AM Consult, Carbon Project, Charles Roswell, Compusult, con terra, CubeWerx, ESRI, FedEx, Galdos, Geomatys, GIS.FCU, Taiwan, GMU CSISS, Hitachi Ltd., Hitachi Advanced Systems Corp, Hitachi Software Engineering Co., Ltd., iGSI, GmbH, interactive instruments, lat/lon, GmbH, LISAsoft, Luciad, Lufthansa, NOAA MDL, Northrop Grumman TASC, OSS Nokalva, PCAvionics, Snowflake, Spot Image/ESA/Spacebel, STFC, UK, UAB CREAF, Univ Bonn Karto, Univ Bonn IGG, Univ Bunderswehr, Univ Muenster IfGI, Vightel, Yumetech

OWS-6 GeoXACML Engineering Report

1 Overview

The aim of this OGC Engineering Report is to show how to provide access control for OGC Web Services (OWS). In the first part of this document we will briefly introduce the relevant details of XACML 2.0, OGC GeoXACML 1.0 and some related profiles. After the introduction of existing access control techniques appropriate to control access to OWS, we will analyse in detail the different ways how to apply the standardised techniques in the OWS context using the example of the OWS-6 GPW airport emergency response scenario. Pros and cons for different approaches will be discussed and possible solutions are deduced. Additionally we will provide detailed recommendations how to improve the used specifications in order that they are able to cope the complexity that one encounters when trying to control access to OWS.

2 Bibliography

[1] eXtensible Access Control Markup Language (XACML) Version 2.0, OASIS Standard. 01 February 2005. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

[2] Geospatial eXtensible Access Control Markup Language (GeoXACML), OGC Implementation Standard. 20 February 2008. http://www.opengeospatial.org/standards/geoxacml.

[3] Hierarchical resource profile of XACML v2.0, OASIS Standard. 01 February 2005. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-hier-profile-spec-os.pdf.

[4] Multiple resource profile of XACML v2.0, OASIS Standard, 01 February 2005, http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-mult-profile-spec-os.pdf.

[5] Core and hierarchical role based access control (RBAC) profile of XACML v2.0. RBAC profile. OASIS Standard. 01 February 2005. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-rbac-profile1-spec-os.pdf

[6] OpenGIS Web Service Common Implementation Standard, Version 1.1.0, OGC Implementation Standard. 09 February 2009. http://www.opengeospatial.org/standards/common

[7] OGC, Open Geospatial Consortium Inc.: OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture, Version: 1.2.0, Date: 2006-10-05, http://portal.opengeospatial.org/files/?artifact_id=18241c

[8] OASIS, SAML 2.0 profile of XACML v2.0, 2005-02-01, http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-profile-spec-os.pdf

[9] OGC, OpenGIS® Web Map Service Implementation Standard, Version 1.3.0, 2004-01-20, http://portal.opengeospatial.org/files/?artifact_id=4756

[10] Brief Introduction to XACML, http://www.oasis-open.org/committees/download.php/2713/Brief_Introduction_to_XACML.html

[11] Service Oriented Security Architecture applied to Spatial Data Infrastructures. Cristian OPINCARU, Munich 2008. http://deposit.ddb.de/cgi-bin/dokserv?idn=988029642&dok_var=d1&dok_ext=pdf&filename=988029642.pdf.

3 Terms and Definitions

For the purposes of this document, the following terms and definitions apply. Please note that terms and definitions taken from other specifications are informative and shaded 5% grey. They are included here for easy reading. For the normative definition, please follow the reference.

3.1 Terms and Definitions from eXtensible Access Control Markup Language (XACML) Version 2.0

For the normative definitions, please see [1]

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

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 (XACML 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

For the normative definitions of XACML related terms please see ([1], p. 8).

3.2 Miscellaneous Terms

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

Combined global A.C.D.R. – a global A.C.D.R. having multiple <Resource> elements

Global access control decision request (global A.C.D.R.) – an access control decision request referring to multiple resources

Global access control decision response – an aggregation of individual access control decision responses

High-level resource – a XML document (e.g. the WS request or response)

(Individual) resource – a XML node

Individual access control decision request (individual A.C.D.R.) – a decision request referring to exactly one node

Individual access control decision response (individual A.C.D.Resp.) – a decision response referring to exactly one node

OWS use case – access control for dynamically generated XML documents that represent the OGC Web Service request and/or OGC Web Service response.

4 Conventions

4.1 Abbreviated Terms

ACDR	access control decision request
ER	engineering report
GeoXACML	Geospatial eXtensible Access Control Markup Language
GML	Geography Markup Language
OASIS	Organization for the Advancement of Structured Information Standards
OGC	Open Geospatial Consortium
OWS	OGC Web Service
OWS-6	OGC Web Services Initiative, Phase 6
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
SAML	Security Assertion Markup Language
SDI	Spatial Data Infrastructure
SOA	Service Oriented Architecture
SRS	Spatial Reference System
URL	Uniform Resource Locator
URN	Uniform Resource Names
WFS	Web Feature Service
WMS	Web Map Service
WS	Web Service

XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language

5 Introduction

OGC Web Services have no built-in access control mechanism and are therefore a priori unprotected. So far no specifications exist, that describe how to establish access control for OGC Web Services. Nevertheless policy languages such as XACML and GeoXACML are available, that support the declaration and enforcement of fine-grained content and context dependant access control. In a nutshell, GeoXACML describes a policy language that uses XML encoding to express complex access rights (e.g. fine-grained spatial access rights). This standardized policy language allows the interoperable processing, exchange and collaborative creation of policies independent from the underlying service based architecture. Besides the policy language, GeoXACML and XACML respectively describe a general architecture and information flow model. This architecture enables a clean separation of the access control system from the Web Service it is protecting.

In order to facilitate the implementation of access control systems for OWS environments this report analyses in detail how to use GeoXACML[1] to protect OGC Web Services. After a theoretical discussion how to utilize GeoXACML for OWS, it will be shown by means of a concrete use case (the OWS-6 GPW airport emergency response scenario) how GeoXACML can be used to secure an OWS based infrastructure. Additional examples will further demonstrate the expressiveness of the policy language when used to protect OWS.

Another focus of this ER results from the following situation. In reality, different problems arise from the fact that GeoXACML, or rather its base specification XACML, gives much freedom when expressing access rights and access control decision requests, containing the information upon which an access control decision is based on. The flexibility that (Geo)XACML provides and the different supported request/response formats of OGC Web Services (e.g. http/get, http/post or SOAP) are potential causes of losing the interoperability between OWS and GeoXACML-based access control systems and between the cooperating components of these access control systems. It is obvious that the flexibility (Geo)XACML provides is inevitable in a general-use standard. As we are analysing in this report how (Geo)XACML can be used in service oriented architectures based on OGC Web Services, our initial situation has some very specific characteristics, that allow to limit the freedom that (Geo)XACML provides without reducing the expressiveness and the capabilities of (Geo)XACML. If the freedom when using the language can be reduced without entailing any functional limits, than the interoperability in GeoXACML based access control systems can be enhanced

[1] Note that GeoXACML is XACML augmented by spatial access control rules. If in a concrete use case no spatial access control rules are needed and it is therefore sufficient to use XACML only to protect an OWS, than the following explanations still apply; apart from sections referring to spatial access control rules.

significantly. Following this reasoning we will analyse in this ER if, and if so, how it is possible to improve the interoperability in GeoXACML based access control systems through clear and unique guidelines when using GeoXACML to protect OGC Web services.

A third focus of this ER is to analyze XACML and related specifications (e.g. the multiple or hierarchical resource profile of XACML) in the context of access control requirements in OWS environments. As we will show later, there are some sections in the corresponding standards that can be refined and extended in order to optimally fulfill the requirements when protecting OWS.

It is important to mention that this report focuses on access control only. Hence it neither describes a comprehensive OGC security framework nor the interplay of the access control system with other security services like e.g. an authentication service.

6 Background Information

In this section we will give some background information and insights in our general understanding of the area of interest.

6.1 General Security Architecture

Our general approach for security is to separate the security aspects technically from the Geo Web Service functionality as much as possible. The advantage of this approach is that existing security concepts and implementations from IT industry (WS-Security standard and extensions, GeoXACML, SAML) can be leveraged for implementing security aspects for Geo Web Services. The second advantage is the possibility of using Geo Web Service implementations (e.g., WMS, WFS) without or little modifications by placing security functionalities as proxy components between the Geo Web Service Client and the Geo Web Service.

In our security framework we assume that the security functionality is divided into separate services that address different security concerns such as authentication, authorization, audit, etc. These services can be flexibly combined and can be used in different configurations for several geo-processing services together. Each of these services can itself be composed of further services (for more information see [11]).

Some of the advantages of this modular security architecture approach are enumerated below:

1. Rather than creating a “fat” security component, our architecture splits the security into separated functional components, therefore reducing the complexity associated with development and maintenance;

2. The solution is fully scalable and easy to upgrade. New services can be easily inserted and existing services can be upgraded without affecting the other components;

3. Since the security services are essentially Web Services, they benefit from all the advantages that characterize Web Services: They can be developed in any programming language, be situated anywhere across the network and they are platform independent as they can be deployed on different application servers.

6.2 Architecture of the Access Control System

The figure below shows the general architecture of a modular and potentially distributed rule based access control system, as outlined in [1]. Even though this architecture proposal comes from the XACML standard, it is non-normative! In addition, it defines the corresponding information flow model between the components of the access control system, the service and context environment. Rule based access control systems offer great flexibility and expressiveness and are therefore an appropriate model to implement access control for OGC Web Services.

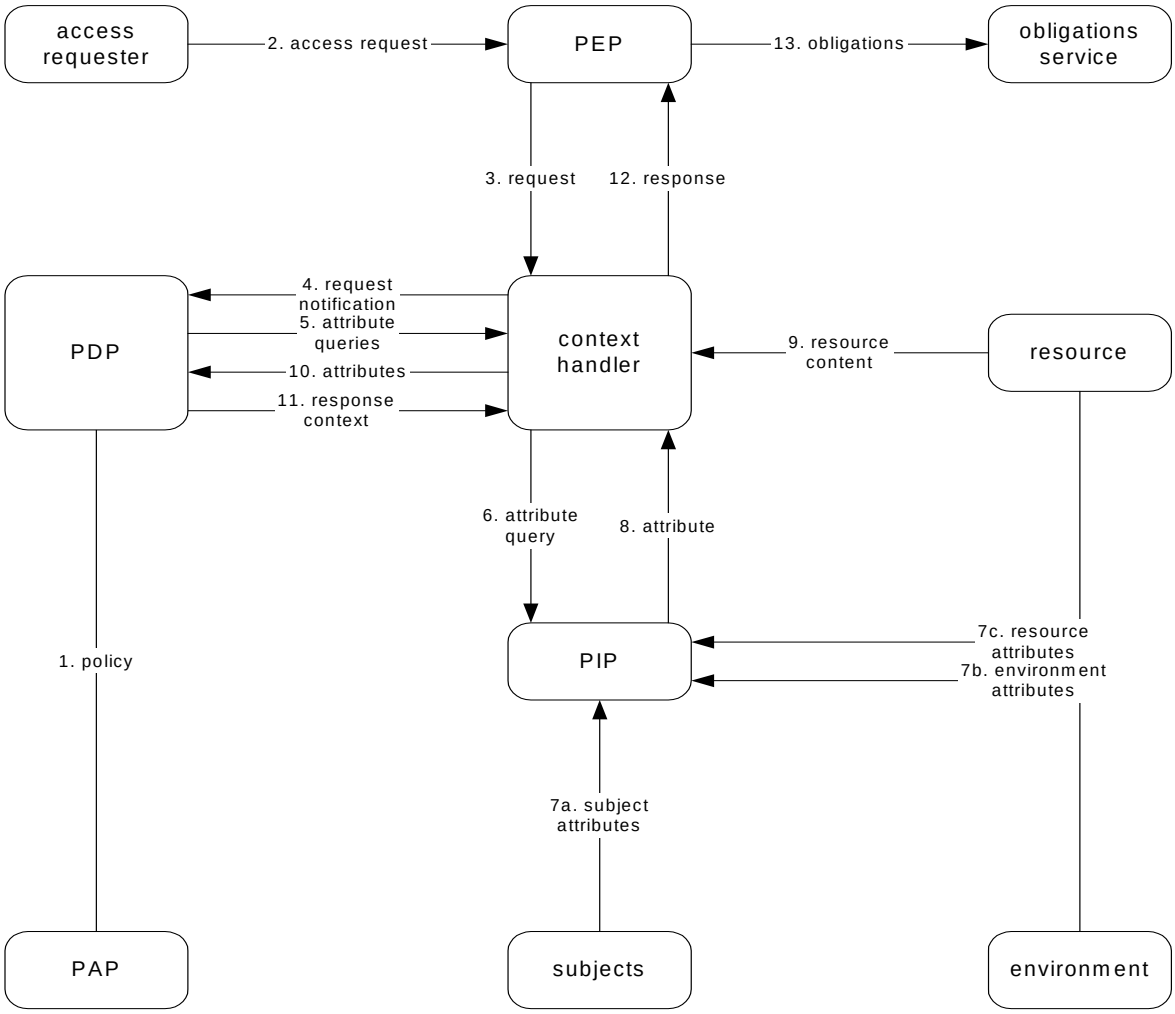

Figure 1: Architecture and Information Flow in a rule based access control system [1]

In the following sections the main components of the access control system and their functionality will be described briefly.

6.2.1 The Policy Administration Point

The Policy Administration Point (PAP) is the component that allows one or multiple policy administrators to edit, maintain and analyse access rights encoded as rules in a well defined access control language.

6.2.2 The Policy Decision Point

The Policy Decision Point (PDP) is the component that derives an authorization decision based on an authorisation decision request, received from a Policy Enforcement Point (PEP). A PDP accesses the policies through a policy repository stored in either a file or a data base or requests the policies from a PAP.

6.2.3 The Policy Enforcement Point

The Policy Enforcement Point (PEP) can be characterized as a switch or even filter that either forwards the intercepted (O)WS-request from the client to the service (and the (O)WS-response from the service to the client respectively) or replies with an adequate error message or modifies the intercepted (O)WS-request or (O)WS-response. The decision whether the (O)WS-request or (O)WS-response is to be forwarded, modified or blocked depends on the authorization decisions, received from the PDP.

The PEP must send authorization decision requests in a particular message format - the access control decision request message (ACDR). Information usually included in such an ACDR message is security assertion information like authentication information, transport protocol information and the Web Service request or response.

6.3 Brief introduction to XACML 2.0

This section provides a brief introduction to XACML (based on [10] and [1]). It is highly recommended to read more detailed information about XACML to become familiar with the characteristics of the language. OASIS XACML technical committee's web site offers comprehensive material.

The XACML describes a XML encoded policy language and a corresponding XML encoded language to express access control decision requests/responses. The policy language is used to describe access control requirements, and has standard extension points for defining e.g. new functions or data types. The request/response language allows to form a query to ask whether or not a given action should be allowed, and interpret the result. The response always includes an answer about whether the request should be allowed using one of four values: Permit, Deny, Indeterminate or Not Applicable. By using the standardised policy language and the corresponding standardised access control decision request/response language, it is possible to find the rules in a policy that apply to a given request and evaluate the request against these rules.

As XACML is a standardized language, developed by a large community, it is more mature than many existing proprietary or application-specific languages. The fact that XACML is standardised additionally facilitates the development of interoperable access control systems and thus allows an easy interoperation of applications that are secured by XACML. Additionally when using a standardized policy language like XACML, users will benefit from the usage of existing sophisticated and highly reliable off-the-shelf-tools for access control systems and from APIs or frameworks for the single components of XACML based access control systems.

At the root of all XACML policies is a Policy or a PolicySet element. A PolicySet is a container that can hold other Policies or PolicySets, as well as references to related policies found in remote locations. A Policy element represents a single access control policy, expressed through a set of Rules.

A XACML Rule is basically a set of conditions on the subject, resource, action and context information that must be met in order to apply to a given decision request. XACML provides a lot of functions to compare values found in a request with those included in the rule. If all the conditions of a rule are met, then the rule applies and its effect is incorporated when deriving the authorization decision.

The figure below shows XACML's policy language model.

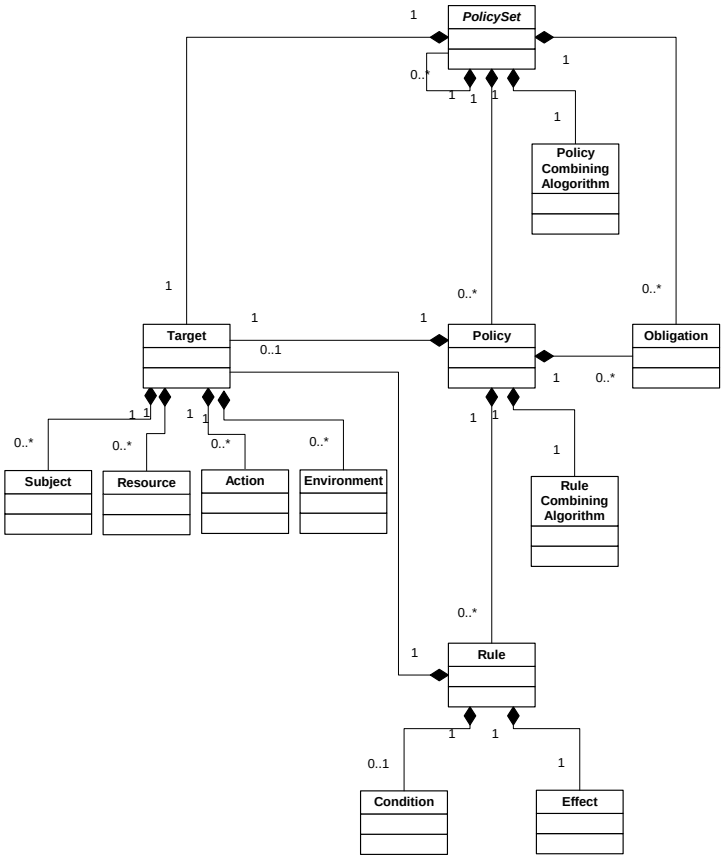

Figure 2: XACML's Policy Language Model [1]

An access control decision request, sent from a PEP to a PDP, is essentially formed by XACML attributes and the ResourceContent (e.g. the WS-request or -response). This data will than be compared to attribute values in a policy to make the access decisions.

A XACML rule or policy resolves attribute values or ResourceContent information in an access control decision request (ACDR) through two mechanisms: the AttributeDesignator and the AttributeSelector. An AttributeDesignator lets the rule or policy specify an attribute with a given name and type, and optionally an issuer as well, and then the PDP will look for that value in the request, or elsewhere if no matching values can be found in the request. AttributeSelectors allow a rule to look for attribute values through an XPath query. A data type and an XPath expression are provided, and these can be used to resolve some set of values either in the decision request document (more precisely in its <ResourceContent> element) or elsewhere.

Both the AttributeDesignator and the AttributeSelector can return multiple values (since there might be multiple matches in a request or elsewhere), so XACML provides a special attribute type called a Bag. Bags are unordered collections that allow duplicates, and are always what designators and selectors return, even if only one value was matched. In the case that no matches were made, an empty bag is returned.

Once some Bag of attribute values has been retrieved, they need to be compared in some way to expected values to make access decisions. This is done though a powerful system of functions. Functions can work on any combination of attribute values, and can return any kind of attribute value supported in the system. Functions can also be nested, so you can have functions that operate on the output of other functions, and this hierarchy can be arbitrarily complex. Custom functions can be written to provide an ever richer language for expressing access conditions.

One thing to note when building these hierarchies of functions is that most functions are defined as working on specific types (like strings or integers) while designators and selectors always return Bags of values. To handle this, XACML defines a collection of standard functions of the form [type]-one-and-only, which accept a bag of values of the specified type and return the single value if there is exactly one item in the bag, or an error if there are zero or multiple values in the bag [10].

Another feature XACML supports are obligations. An XACML obligation is an operation specified in a XACML Policy or PolicySet that should be performed by the PEP in conjunction with the enforcement of an authorization decision [1].

Thanks to the described capabilities of XACML, the language can be used to express access control rules that enforce authorization semantics when someone tries to access XML documents. In a SOA environment these XML documents generally represent Web Service requests and/or Web Service responses respectively. Therefore XACML can be used to state access control rules for OGC Web Services, which ensure that the users' requests and responses conform to the access control policy in place.

Note that XACML rules can refer to any node or node-set in a XML encoded Web Service requests or Web Service responses. This central characteristic allows for very fine-grained access control. If needed, the policy writer can write rules referring to individual nodes of arbitrary type (e.g. element nodes, attribute nodes, text nodes, etc.). Of course it is also possible to define rules referring to sets of nodes too.

The derivation of an access control decision for a specific node in a XML document can depend on the values of other nodes within this document. The policy writer has maximal flexibility in specifying which node or even node-sets will be evaluated through the supported functions, to derive an authorization decision for another node in the document. This property makes it possible to define content dependant access control rules (e.g. permit access to building nodes if the owner's name of this building equals Alice and the price of this building is less than 1,000,000 US $).

Another class of access rules that can be implemented with XACML is the so called context dependant rule type. Predicates in context dependant rules select the information represented in the context of an access control system and match this data against literals in rules thereby providing context depended authorizations. A context dependant rule is for example: permit access between 8 am and 7 pm.

The table below summarize the types of rules that can be defined using XACML.

Rule Type	**Example**
fine-grained rule	(deny, Alice, read, /Building/Price)
content dependant rule	(permit, Alice, read, /Building/Price, if /Building/owner = Alice)
context dependant rule	(permit, Alice , read, /Building, if accessTime is between 8am-8pm

Table 1: Types of rules supported by XACML

From the access control system perspective, the nodes of a XML document (representing the Web Service request or Web Service response) are atomic information entities. As it is possible to generate access control decisions for each single node (more accumulated decisions are of course possible too) a filtering mechanism after the access control process is trivial to implement. All the PEP needs to do, is to filter out each node for which the access control decision was deny. This enables, that a XACML based access control system can give back the intersection of requested and accessible (according to the access control policy) data entities. In this context it is worth mentioning, that XACML allows pre- and/or post-processing. This means that access control can be enforced on the Web Service request and/or Web Service response through appropriate pre- and/or post-processing rules. Dependant on the type of service interface and the needed rules either both approaches or only one of the two approaches might be used.

6.4 Introduction to GeoXACML 1.0

The sections below give a short introduction to GeoXACML. Readers of this ER are highly encouraged to read the GeoXACML standard [2] for more detailed information.

6.4.1 Features of GeoXACML at a Glance

The Geospatial eXtensible Access Control Markup Language (GeoXACML) defines a geo-specific extension to the OASIS standard "eXtensible Access Control Markup Language (XACML), Version 2.0".

A special type of content dependant access control rules needed in spatial data infrastructures (SDI) are the so called spatial access control rules (e.g. permit read access to building data if the building **is within the state boundary of California**). The spatial authorization semantics can be expressed through spatial predicates like within or disjoint in the condition parts of spatial rules. The predicates refer to the spatial attributes of features in the (O)WS-request or –response represented in the decision request and the spatial constants in the rule itself. A spatial attribute is e.g. the base geometry of a building that is uniquely defined through a GML polygon definition in a specific spatial reference system. Through these spatial rules stable geo-specific authorization semantics can be expressed directly. Spatial rules augment the capabilities of an access control system as they provide a powerful, native and completely new type of authorization semantics.

GeoXACML defines a spatial extension of XACML. Spatial data types and spatial authorization decision functions are added, that are needed to define expressive spatial access control rules. In short, GeoXACML specifies:

- How to use a geometry model based on Simple Features on which the geometric data types in spatial access rules have to be based on,
- different encoding languages for geometric data types,
- testing functions for topological relationships between geometries,
- geometric functions and
- a set of common XACML Bag functions for the new geometry data type.

6.4.2 GeoXACML's Geometry Model und Spatial Functions

In order to support a flexible and straight forward solution allowing geometric data types in rules or ACDR that easily comply with the base XACML specification, GeoXACML extends XACML by only one new data type, that is named "urn:ogc:def: dataType:geoxacml:1.0:geometry". This implies that the data type of every geometric attribute value, Bag of geometric values or pointer to geometric data (i.e.

AttributeSelector or AttributeDesignator) in a GeoXACML policy SHALL always be "urn:ogc:def:dataType:geoxacml:1.0:geometry". Specific values for a geometry <AttributeValue> or referenced spatial data of data type urn:ogc:def:dataType: geoxacml:1.0:geometry" must be represented by of one of the following basic types: Point, LineString, Polygon, MultiPoint, MultiLineString or MultiPolygon (c.p. [2], p.14 f).

By using GeoXACML the policy writer can use various topological functions (e.g. within, crosses, intersects), metric functions (e.g. area, length, distance) or geometric calculations (e.g. union, buffer, difference) within the spatial rule conditions. The table below shows spatial operators supported by GeoXACML that can be used to express powerful spatial access control rules:

Topological Functions	Constr. Geometric Functions	Miscellaneous Functions
Equals	Buffer	Distance
Disjoint	Boundary	IsWithinDistance
Touches	Union	Length
Crosses	Intersection	Area
Within	Difference	IsSimple
Contains	SymDifference	IsClosed
Overlaps	Centroid	IsValid
Intersects	ConvexHull	

Table 2: Spatial functions provided by GeoXACML

For examples showing the expressive capabilities of GeoXACML see 8.3 and 8.2.

6.4.3 Summary of GeoXACML's Capabilities

The following list summarizes some of the main characteristics of GeoXACML. Note that, as GeoXACML is an extension of XACML, it directly inherits all the capabilities provided by XACML.

GeoXACML allows…:

- the definition of fine grained, positive and negative access control rules (i.e. the atomic access control object is an arbitrary XML node).
- to base the authorization decisions on the evaluation of other node values.
- pre- and/or post-processing (i.e.: access control on the Web Service request or Web Service response).
- easy filtering/modification of interactions with insufficient rights.
- the definition of content dependant access control rules.
- the definition of spatial access control rules.
- the definition of context dependant access control rules.
- the flexible combination of all kind of rule types.
- the easy implementation of a spatial access control system based on standardized and expressive policies

6.5 The Hierarchical and Multiple Resource Profile of XACML 2.0

The next two sections introduce the Hierarchical and Multiple Resource Profile of XACML 2.0 ([3] and [4]). It should be highlighted, that both profiles can be used together with XACML or GeoXACML.

6.5.1 The Hierarchical and Multiple Resource Profile of XACML 2.0 in the Context of Access Control for OGC Web Services

As mentioned in the introduction, the motivation of this ER is to describe how to provide access control (in service oriented spatial data infrastructures) based on OGC Web Services (OWS). Therefore the assets we are trying to protect are Web Service requests and responses (except OWS that return images) are usually encoded in XML. From the access control perspective this implies that a PEP intercepts XML structured OGC Web Service requests and responses (if reasonable – i.e. if XML structured) and initiates the access control process. Hence the objects for which access rights have to be verified are dynamically generated XML documents - the Web Service request and/or Web Service response respectively. In the following we call the OWS request or response the high-level resources. Obviously a high-level resource as an atomic object is a pretty coarse-grained view of the data that we are trying to protect. In order to support fine-grained and content dependant access control we have to regard the nodes of these high-level resources as the actual entities for which the access control process evaluates the access rights. Therefore the nodes of the Web Service request or Web Service response are the "real" resources. We will refer to these XML nodes simply as resources. It should be clear that by defining rules that refer to the single nodes it is possible to define more coarse-grained authorizations too (see 8.3).

OASIS provides two profiles for the use of XACML with resources that are XML encoded – the Hierarchical resource profile of XACML 2.0 and the Multiple resource profile of XACML 2.0. As the resources in the OGC use case are hierarchically organized nodes encoded in XML, both profiles are of relevance and will be introduced in the next sections. In section 7.2 of this ER we will evaluate these two profiles in detail with respect to the OWS use case and derive suggestions that improve the quality and usefulness of these profiles.

6.5.2 A Brief Summary of the Hierarchical Resource Profile of XACML 2.0

The intent of the Hierarchical Rresource profile of XACML 2.0 is to provide a standardized profile for the use of XACML with resources that are structured as hierarchies (e.g. a tree of nodes encoded in XML). The profile addresses resources represented as nodes in XML documents or represented in some non-XML way ([3], S.1). As the resources subject to access control in OWS are XML nodes of OWS requests and responses, we will in the following only summarize the sections of the Hierarchical resource profile of XACML 2.0 that refer to XML encoded high-level resources.

Before we start with the summary of the Hierarchical resource profile of XACML 2.0, that will show how OASIS intends to use XACML in order to provide access control for a resource that is organized as a hierarchy, we will list two facts that make hierarchically organized resources special. First, in order to support the flexible definition of authorization semantics and easy administration, it is necessary to be able to express access rules that refer to an entire sub-tree or a single node (or leave) of the hierarchy. Second, in the context of content-dependant access rights, it is essential to be able to define rules that control access to one node depend on the value(s) of other nodes in the hierarchy.

As indicated before, the profile treats the nodes in a hierarchical high-level resource as individual resources. Every individual authorization decision requests refers to exactly one node (c.p. the resource-id attribute). An authorization decision that permits access to an interior node does not necessarily imply that access to its descendent nodes is permitted (see 7.2.1.1 item 1 for more details on this issue).

To ensure that XACML rules/policies apply to the resources -- the XML nodes -- as intended, it is very important to have a consistent representation of a hierarchical resource in an XACML access control decision request and of the identity of the nodes in the decision request (c.p. the resource-id values) and in the policies. We will come back to this important issue in section 7.1 of this document.

6.5.2.1 Representation of a Hierarchical Resource in an XACML Access Control Decision Request

According to the profile there are two potential ways of representing a hierarchical resource:

1. The hierarchy, of which the node (referred to in the ACDR by resource-id) is a part, is represented as an XML document that is included in the XACML ACDR.

2. The requested resource is not represented as a node in an XML document in the ACDR and there is no representation of the hierarchy (of which it is a part), included in the decision request.

It is pointed out in the profile that there is no assumed correlation between the structure of the resource as represented in the decision request and the actual structure of the high-level resource as intercepted by the PEP. This means that theoretically, option 1. and 2. might be possible to represent the high-level resource (the XML encoded OWS request or response). Nevertheless in the OGC context option 1. is the natural choice as we usually have XML encoded requests and responses. A more detailed discussion on this statement can be found in 7.3.

Following option one and some further remarks in the profile and the XACML specification itself, it follows, that a high-level resource (a XML encoded OWS request or response) is included in the XACML decision request (more precisely as child of the <ResourceContent> element).

6.5.2.2 Representation of the Identity of a Node in a XACML Decision Request

In order for XACML policies to apply consistently to nodes in a hierarchical resource, it is necessary for the nodes in that resource to be represented in the decision request and in the policies in a consistent way. If e.g. a XACML rule refers to a node using one representation, but the ACDR refers to the node using a different representation, then the rule will not apply, and security may be compromised ([3], S.6).

For this reason the profile recommends, that in a decision request the identity of a node in a resource, that is represented as an XML document, shall be a XPath expression that evaluates to exactly that one node in the copy of the resource, that is contained in the <ResourceContent> element of the <Resource> element of the <Request> element ([3], S.6).

Therefore an ACDR's <Resource> element shall contain amongst other nodes the following elements and XACML attributes:

- A <ResourceContent> element that contains the entire XML document instance of which the requested node is a part of. Remember that each individual ACDR refers to exactly one node (c.p. the resource-id attribute)

- An XACML <Attribute> element with an AttributeId of "urn:oasis::names:tc:xacml:1.0:resource:resource-id" and a DataType of "urn:oasis:names:tc:xacml:2.0:data-type:xpath-expression". The <Attribute-Value> of this <Attribute> SHALL be an XPath expression whose context node SHALL be the one and only child of the <ResourceContent> element. This XPath expression SHALL evaluate to a node set containing the single node in the <ResourceContent> element that is the node to which the access control decision request refers to.

In this summary we did not introduce the <Attribute> elements resource-parent, resource-ancestor and resource-ancestor-or-self that shall also be part of a decision request's <Resource> element according to the profile. We omitted their introduction as we don't see any use of these elements in the OWS use case and additionally they only represent redundant information in relation to the information contained in the <ResourceContent> element. It seems that these elements are only relevant if one follows option 2. as introduced above. A more detailed discussion on this topic and the following conclusions will be outlined in the analysis section (c.p. 7.2.1.1).

6.5.2.3 Stating Policies that Apply to Nodes

Conformant to the representation of a hierarchical resource in the access control decision request and the representation of the identity of the node, the ACDR refers to the profile describing some alternatives how to define matching policies, as of relevance for the OWS use case is section 4.2 of the profile. In this section the use of the xpath-node-match function is recommended to state rule predicates that apply to one or more nodes in that

resource. Additionally the standard XACML bag and higher-order bag functions MAY be used to state policies that apply to one or more nodes in a resource represented as an XML document. The nodes used as arguments to these functions MAY be specified using an <AttributeSelector> that selects a portion of the <ResourceContent> element of the <Resource> element ([3], S. 11). Section 7.2.1.1 item number 8 provides a detailed analysis of this recommendation and will reveal its weaknesses. Additionally a more sophisticated supplementary way of how to specify policy predicates for hierarchical resources will be introduced in the same section.

Before we close the summary of the Hierarchical resource profile of XACML 2.0, it is important to mention, that when using XACML with hierarchical resources like XML documents (and you are therefore using the Hierarchical resource profile of XACML 2.0), it is generally useful to generate ACDRs for multiple nodes (i.e. resources) in a single ACDR. In the following we will call such a request a global XACML access control decision request (global ACDR). How such a global ACDR is expressed and processed in a standardized way is described in the Multiple resource profile of XACML 2.0 (c.p. 6.5.3). This profile also describes how get back only a single authorization decision as a result to a global ACDR.

It is important to highlight, that all global XACML access control decision requests that refer to multiple nodes in a hierarchical resource will be resolved to individual access control decision requests that refer to exactly one single node in the hierarchy. For more details see 6.5.3.

The Hierarchical resource profile may be considered to be layered on top of the Multiple resource profile which in turn is layered on top of the core XACML (or GeoXACML) specification ([3], S.4).

6.5.3 A Brief Summary of the Multiple Resource Profile of XACML 2.0

This section provides a brief summary of OASIS's Multiple resource profile of XACML v2.0. Analogous to the summary of the Hierarchical resource profile of XACML 2.0 in section 6.5.2 we will in the following only summarize the sections of the profile that refer to nodes that are part of XML encoded high-level resources as this reflects the situation when XACML is used to control access to OWS.

6.5.3.1 Options how to Create an Access Control Decision Request for Multiple Resources in a Single Global XACML Access Control Decision Request

The general idea of the profile is to allow the definition of a global ACDR for multiple resources. The profile than specifies how to generate individual access control decision requests one for each of the individual resources as addressed through the global ACDR for multiple resources.

In order to support different use cases, the profile offers different options for defining a global access control decision request for multiple resources. For each option the transformation mechanism that defines how to derive the individual ACDR from the global ACDR is described.

The central elements that are necessary to specify a global ACDR for multiple resources is the scope and resource-id attribute that are child elements of an XACML <Request> element.

In the following sections we will briefly explain the different variants to express a global ACDR as described by the profile. As mentioned before only sections referring to XML encoded high-level resources are summarized.

Option 1:

Baseline:

- the <ResourceContent> element contains the whole XML document which is subject to the access control process.
- possible resources are all XML nodes under the <ResourceContent> element
- the resource-id attribute is of data type XPath expression and is used to identify exactly one node under the <ResourceContent> element.
- possible scope:xml values are Children, Descendants, Immediate and are used to indicate in combination with the resource-id attribute the node set that the global ACDR refers to.

A global ACDR for multiple resources is defined by the values for the scope:xml and resource-id XACML Attribute.

In case the value of the scope attribute is **Children**, the set of individual resources that are addressed through this global ACDR for multiple resources are, the node specified by the resource-id value (Remember: an XPath expression identifying exactly one node) and all child nodes of this node.

In case the value of the scope attribute is **Descendants**, the set of individual resources that are addressed through this global ACDR for multiple resources are, the node indicated by the resource-id value (Remember: an XPath expression identifying exactly one node) and all descendant nodes of this node.

In case the value of the scope attribute is **Immediate**, we have in fact an ACDR to a single node (the one indicated by the resource-id value) in a hierarchical resource. This is therefore no proper ACDR for multiple resources.

After having explained how to identify the individual resources specified by a global access control decision request for multiple resources, the next step is to explain how to derive the individual access control decision requests for each individual resource.

The profile defines that each individual ACDR shall be identical to the original global ACDR for multiple resources with two exceptions:

1. the “scope” attribute SHALL NOT be present and

2. the resource-id attribute shall uniquely identify the **individual resource**

Exception 2. directly implies that the XPath expression (i.e. the value of the resource-id attribute of the individual ACDR) identifies exactly one node under the <ResourceContent> element. Note that the values of the resource-id attributes of individual ACDR are therefore (with one exception) different from the values of the resource-id attribute of the global ACDR The mentioned exception occurs when the individual ACDR refers to the node specified by the resource-id value of the global ACDR.

Option 2:

In order to indicate to the Context Handler that this is another option of a global ACDR for multiple resources the **scope:xml** attribute value has to be

urn:oasis:names:tc:xacml:2.0:profile:multiple:**xpath-expression**

This option is a slightly generalised (but also limited) alternative of option 1 (see discussion under 7.2.1.2 for more details). Besides the new and one and only possible value for the scope attribute (urn:…:xpath-expression), the essential difference is, that in this option the resource-id attribute is an XPath expression evaluating to a **node set** that represents multiple nodes, the individual resources in the <ResourceContent> element. This means that in this case, the specification of the individual resources is done directly through the XPath expression in resource-id and not through the combined use of the resource-id value and scope values like children and descendant like in option 1.

The next step, the generation of the individual ACDRs for each individual resource is identical to Option 1.

Option 3:

This option starts from the pre-condition that the global ACDR for multiple resources has more than one <Request> element.

A Context Handler receiving such a combined global ACDR for multiple resources has to generate separate **global** ACDRs for each <Request> element. Each such global access control decision request SHALL be identical to the original request context with one exception: **only one <Resource> element SHALL be present**. If such a global ACDR contains a "scope" attribute, then it SHALL be further processed according to the processing introduced above (see option 1 and 2). This processing will hence involve the transformation of the actual ACDR into new individual ACDRs before they will finally get evaluated by the PDP.

6.5.3.2 Options how to Obtain a Single Authorization Decision in Response to an Access Control Decision Request for Multiple Resources

After having explained the different options how to generate an ACDR for multiple resources through a single request, another directly related topic has to be discussed. In certain use cases it might be desired that the response to a global ACDR for multiple resources is not a set of individual access control decision responses (one response for each corresponding individual access control decision request). A PEP may instead wish to submit a global ACDR for multiple resources (e.g. for all the nodes in a hierarchy), and may wish to obtain **only a single** authorization decision (<Result> element) that indicates whether access is permitted to **all** of the requested nodes. Such a request context might be used for example when the requester wants access to an entire XML document and if the PEP is not interested in the access control decisions per node.

We will briefly resume here how this has to be done conformant to the OASIS Multiple resource profile of XACML v2.0 specification.

To indicate to the Context Handler that the PEP wishes to receive only one global, single authorization decision response in reply to an access control decision request for multiple XML nodes, the scope:xml attribute value has to be set to

urn:oasis:names:tc:xacml:2.0:profile:multiple:**entire-hierarchy:xml**

At this point it is important to highlight that the Multiple resource profile of XACML v2.0 instructs also to use the scope attribute to indicate the behavior as just described. Thus, the direct consequence of this "overloaded" use of the scope attribute is that you can't freely combine the different options to generate an ACDR for multiple resources and the options for choosing between a single global access control decision response or multiple individual access control decision responses (corresponding to the responses to

the individual ACDRs). This unsatisfying situation will be explained in more detail in the analysis section (c.p. 7.2.1.2 item 7).

The new semantics implied through the scope:xml value of multiple:entire-hierarchy:xml manifests when it comes to the submission of the response(s) corresponding to the ACDR for multiple XML resources. The Context Handler submits each individual ACDR to the PDP for evaluation and SHALL keep track of the <Decision> in the corresponding <Result> elements. If and only if all the individual ACDRs evaluate to “Permit”, then a single <Result> containing a <Decision> of “Permit” SHALL be placed into the response context returned to the PEP. If any of the request contexts evaluates to “Deny”, “Indeterminate”, or “NotApplicable”, then a single <Result> containing a <Decision> of “Deny” SHALL be placed into the response context returned to the PEP (see 7.2.1.2 item 10 for a discussion of the aggregation in the indeterminate case).

7 Analysis of (Geo)XACML and Related Profiles in the Context of Access Control for OGC Web Services

So far, we introduced the general capabilities of XACML 2.0, GeoXACML 1.0 and the Multiple and Hierarchical resource profiles of XACML 2.0. During OWS-6 we used these standards to support access control in the airport scenario of the GPW thread.

The following sections are a summary of our findings that resulted from a detailed analysis of the applicability of the mentioned standards in OWS-6 and in OWS based infrastructures in general.

Upfront it is worth mentioning, that although the original motivation for the following discussion and analysis comes from the geospatial problem domain (i.e. access control for OWS), most of the findings will nevertheless be valid for Web Service based architectures in general. Therefore the findings in this ER will also be valuable for a broader community using XACML and should therefore be introduced to OASIS XACML WG (c.p. 10.2).

7.1 Possible Interoperability Flaws when Using XACML

While defining the needed access control rules for the GPW airport scenario and analyzing the different ways how this can be done, we identified some sources of unnecessary freedom that can result in loosing interoperability in the access control system even though you are using a standardized rule language like XACML or GeoXACML.

In this section we will describe where and why it is possible to loose interoperability in an XACML based access control system and how this loss of interoperability within the access control system can be avoided.

As explained before the PEP generates an ACDR, when intercepting the communication between the client and the service. This ACDR has to be expressed in a standardised syntax in order to guarantee that the rules will match and enforce the authorization semantics as intended. Therefore we have a strong dependency between the syntax and semantics of

- the ACDR and
- the access control rules.

Only if the rules and the ACDR are exactly associated with each other, the access control system will enforce the intended authorisation semantics. In other words: The way how the PEP includes the Web Service request or response into an ACDR has direct effect on how to define and structure an (Geo)XACML policy. From the opposite, the way how the

policy writer declares the access rights in a policy has direct influence on the way the PEP has to structure the ACDR.

It might be a little bit surprising that interoperability problems can occur when using a standard like XACML that already defines a language to express the ACDR and a corresponding language to define the rules and policies respectively.

The core of the problem is the philosophy behind the XACML core specification. XACML defines a general-purpose language and therefore XACML comes with some flexibility how to use the standardized ACDR language and the standardized rule language. It is exactly this flexibility that can result in interoperability flaws. When using XACML it is assumed that for a concrete application domain, the involved parties agree on unique guidelines that make it more definite how to use both languages. These guidelines ensure that in the end the rule instances will match on the ACDR instance. Note that at the moment there are no such guidelines how to use XAXCML or GeoXACML in the OWS use case even though they are actually urgently needed to ensure interoperability (c.p. 10.2).

In the OWS use case, i.e. in case of access control for OGC Web Service the mentioned flexibility shows up in the different alternatives, how to represent the information included in the OWS request or response in the ACDR.

In the following we will describe the different ways, how to represent the information included in an OWS request or response in an ACDR and derive a solution how this can be done best. In addition it will be explained, how to define the rules corresponding to the deduced solution. It is important to note that the solution proposed in the ER is based on OASIS' multiple and hierarchical resource profile of XACML.

7.1.1 Options how to Include a OWS Request or Response in a XACML ACDR

Section 6.3 introduced the two mechanisms XACML provides for representing resource specific information in an ACDR and how to reference to this information in the rules. In order to represent OWS specific information in an ACDR, the implementer of a PEP (responsible for the generation of an ACDR), has the choice between the two mechanisms (namely the Attribute/AttributeDesignator approach and the ResourceContent/AttributeSelector approach). As explained before this freedom of choice can result in the loss of interoperability and it is therefore important to understand the advantages and disadvantages of each of the mechanisms and to know which of the approaches is more appropriate in the OWS use case. Therefore, in the following two sections we will analyze the pros and cons of the two approaches.

7.1.1.1 Using the Attribute/AttributeDesignator Approach to Represent OWS Specific Information in a XACML ACDR

When using the Attribute/AttributeDesignator approach the PEP has to instantiate XACML Attribute name-value pairs based on the information available in the OWS request or response. This implies that the PEP has to parse the incoming OWS request or

response and check if certain information is available and if so instantiate the corresponding XACML Attribute name-value-pair and add it to the ACDR.

In the two examples below you can see the schema behind such an instantiation.

Example 1: Pre-Processing

Intercepted OWS Information	XACML Attributes in an ACDR	
<?xml version="1.0" ?> <wfs:GetFeature service="WFS" version="1.1.0" …> <wfs:Query typeName="myns:Building"> <wfs:PropertyName>myns:owner</…> <wfs:PropertyName>myns:price</…> <wfs:PropertyName>myns:location</…> <ogc:Filter> ... </ogc:Filter> </wfs:Query> <wfs:Query typeName="myns:Street"> <ogc:Filter> ... </ogc:Filter> </wfs:Query> … </wfs:GetFeature>	AttributeName	AttributeValue
	urn:???:service	urn:???:WFS
	urn:…:action-id	wfs:GetFeature
	urn:???:featureType	myns:Building
	urn:???:featureType	myns:Street
	urn:???:PropertyName	myns:owner
	urn:???:PropertyName	myns:price
	urn:???:PropertyName	myns:location
	…	…

Table 3: Attribute/AttributeDesignator approach when performing access control on the WS-request

Example 2: Post-Processing

Intercepted OWS Information	XACML Attributes in an ACDR	
<wfs:FeatureCollection ...> <gml:boundedBy>…</…> <gml:featureMember> <Building> <Owner>Alice</Owner> <Price>1000000</Price> <Location> <Polygon @srs=„...“>...</Polygon> </Location> </gml:featureMember> <gml:featureMember> <Building> <Owner>Bob</Owner> <Price>500000</Price> <Location> <Polygon @srs=„...“>...</Polygon> </Location> </gml:featureMember> <!--... more features .....--> </wfs:FeatureCollection>	AttributeName	AttributeValue
	urn:???:service	urn:???:WFS
	urn:…:action-id	urn:???:resp2GetFeature
	urn:???:featureType	urn:???:Building
	urn:???:owner1	urn:???:Alice
	urn:???:owner2	urn:???:Bob
	urn:???:price1	1000000
	urn:???:price2	500000
	urn:???:polygon	...
	urn:???:srs	urn:...:wgs84
	…	…

Table 4: Attribute/AttributeDesignator approach when performing access control on the response

During our analysis we discovered that, when using the Attribute/AttributeDesignator approach in the OWS context, one encounters a lot of difficulties. For example lots of URNs for attribute-names and -values have to be defined for unique expression and identification of the information in an ACDR (c.p. the tables above) and for unique identification inside the rules. Even worth is the fact that the XACML Attributes destroy the hierarchical structure and the semantic relationships of the OWS data and imply more generalized i.e. coarse-grained atomic information entities. XACML Attributes seems to be only appropriate to use if the information represented by them is atomic without structural relation, which is not the case in the OWS context. When transforming information contained in a OWS request or response that is hierarchically structured into XACML Attributes you loose the structural connections between the nodes and you generate more coarse-grained atomic information entities. This loss of referenceable semantic information can only be avoided through the extensive generation of attributes one for each possible subset (2^n) which is not feasible given the number of information entities in a OWS request or response.

It should however be noted that the argument that XACML Attributes provide better evaluation performance as the ResourceContent/AttributeSelector is for most cases not valid in the OWS context. XACML Attributes can be implemented as HashMaps ($\rightarrow$O(n) access time) but in order to instantiate them an XPath evaluation over the OWS data (in case of XML encoded (O)WS Request) is nevertheless necessary. Therefore there is no real performance advantage when using the Attribute/AttributeDesignator approach compared to the ResourceContent/AttributeSelector approach when caching of XPath evaluation results during the processing of an ACDR. The careful reader might have noticed that the performance advantage of XACML Attributes might apply in case of KVP encoded OWS requests. Nevertheless given the other serious disadvantages of the Attribute/AttributeDesignator approach this mechanism is nevertheless not more feasible (see 7.3 for another related problem).

Therefore we come to the conclusion that information about OWS requests or OWS responses should not exclusively or even partially be represented as XACML Attributes in the ACDR. In short, the reason is that the Attribute/AttributeDesignator approach is not powerful enough as arbitrary OWS requests and responses can not be easily nor completely transformed into appropriate XACML Attributes **without significantly reducing the possible authorization semantics**. Further more many URNs need to be defined in order to achieve interoperability.

7.1.1.2 Using the ResourceContent/AttributeSelector Approach to Represent OWS Specific Information in a XACML ACDR

After having shown that XACML Attributes are not a suitable generic mechanism for representing OWS specific information in an ACDR, we will analyse if OWS information can be adequately represented and referenced through the use of the ResourceContent/AttributeSelector mechanism.

One advantage of the ResourceContent/AttributeSelector approach is that it provides a quite simple solution and allows building PEPs that need very limited service unspecific

functionality. The PEP doesn't need to instantiate any explicit XACML Attribute for information contained in OWS requests or OWS responses. All information about OWS requests or OWS responses is represented inside the XACML <ResourceContent> element only. The PEP simply has to copy the OWS request or response as child element under the <ResourceContent> element **as is**.

Next advantage is that no attribute instantiation is necessary inside the PEP that implies that no URN definitions are necessary. Instead, the needed uniqueness of OWS information in ACDR is provided automatically thanks to the already standardized OGC XML schemas in addition to the specific application dependant OWS data schemas. The OGC schemas and the application dependant OWS data schemas together describe the data that can be contained in an OWS request or response and thereby provide a well defined frame for policy definition.

The most important advantage of the ResourceContent/AttributeSelector approach is its flexibility and expressiveness. Arbitrary information (i.e. node sets) under the <ResourceContent> element can be selected and serve as input for functions in XACML rules. It is obvious that this flexibility is not given when using the XACML attributes.

Using the ResourceContent/AttributeSelector approach implies that XPath processing will be necessary in case of content dependant rules, but as explained under 7.1.1.1 the performance of this approach is nevertheless not slower than the Attribute/Attribute-Designator approach. For this reason "performance penalty" is no disadvantage of this approach.

Another observation that should be mentioned here is that some little extra guidelines might be helpful to allow unique referencing of "indirect" information about OWS requests or OWS responses. The flexibility occurs if you like for example to retrieve uniquely the actual service name or operation name from an OWS response to a getFeature request through the ResourceContent/AttributeSelector approach. As this information is only indirectly available under the <ResourceContent> element the policy writer has to choose some indicator nodes to recognize the service type or operation name. He could e.g. decide to use the <FeatureCollection> Element as indicator for the action type in case of a WFS getFeature response. Having unique, standardized guidelines how to find out the service name and operation type for each OWS might contribute towards a more uniform definition of access control rules in the OWS use case. Note that these guidelines are not necessary to achieve interoperability between the PEP and PDP but are nevertheless helpful to facilitate more sophisticated policy administration.

7.1.2 Conclusion

Following the discussion in the last two sections we conclude that XACML Attributes should not be used at all to represent information on OWS requests or responses. Instead PEPs and policy writers should always use the ResourceContent/AttributeSelector approach to represent and reference OWS specific information. Please note that this

guideline is crucial to be standardizes and requires compliance by all parties trying to protect OWS by (Geo)XACML in order to achieve real interoperability.

7.2 Analysis of XACML and Related Profiles in the OWS Context

7.2.1 The Analysis of the Hierarchical and Multiple Resource Profile of XACML 2.0 in the Context of Access Control for OGC Web Services

In section 6.5 we introduced the Multiple and Hierarchical Resource Profile of XACML 2.0 and explained that both profiles apply when XACML is used to control access in OWS environments. Unfortunately we discovered various shortcomings when using these profiles in the OWS context. The aim of this section is to briefly summarize the shortcomings of the current version of the Multiple and Hierarchical Resource Profile of XACML 2.0 and to deduce general suggestions for improvement.

The shortcomings are listed by line number in the order as they occur in the Hierarchical or Multiple Resource Profile of XACML 2.0 document. Text blocks marked critical indicate the most serious shortcomings. After the description of the shortcoming a short suggestions how to improve the situation is added.

7.2.1.1 Shortcommings in the Hierarchical Resource Profile of XACML 2.0 and Sugestions

1. Line 80: comment
   *"An authorization decision that denies access to an interior node **does not** imply that access to its descendant nodes is denied."* ([3], S.3).
   If access to an interior node is denied but access to children nodes is possible implies that the XML tree structure will get destroyed by deleting the access restricted interior node. Therefore the tree structured XML resource becomes a forest (at least two trees whereby the new trees are the sub-trees under the restricted interior node) and thus will not conform to the original XML schema of the request or response anymore. We argue that in the OWS use case such a behavior is not needed or wanted.
   Conclusion: In the OWS use case it is possible to say that: An authorization decision that denies access to an interior node **does** imply that access to its descendant nodes is also denied.

2. Line 99: interoperability problem (critical)
   "[XPath] expressions can be used to reference nodes in this document in a standard way, and can provide unique representations for a given node in the document." ([3], S. 3).

   Forcing to use XPath expressions referring to exactly one node in the <ResourceContent> element limits the representation possibilities, which is a step in the right direction. But this still leaves some flexibility for no real reason and might therefore cause interoperability problems. Example:
   A resource-id attribute value in an individual decision request might be:
   `/Request[1]/Resource[1]/ResourceContent[1]`

```
/wfs:FeatureCollection[1]/gml:featureMember[1]/Building[1]
```

A rule e.g. allowing access to building nodes will have a predicate like the following:

```
regexp-string-match( resource-id,
/Request[1]/Resource[1]/ResourceContent[1]/wfs:FeatureCollection\[
\d+\]/gml:featureMember\[\d+\]/Building\[\d+\]$)
```

By using such a predicate all decision requests referring to "Building" nodes will match and the rule will get evaluated. In this example we used the abbreviated XPath syntax to refer to exactly one node. In case the Context Handler uses unabbreviated XPath syntax when deriving the individual ACDRs from a global ACDR the rule won't match any more.

Conclusion: In order to get a unique representation for a node in the document it should be specified more precisely which syntax the XPath values of the resource-id attribute have to conform. We recommend to use a syntax as shown in the example (i.e. /elementNodeName[integer]/../elementNodeName [integer] in case of element nodes and /elementNodeName[integer]/../@attributeNodeName in case of attribute nodes[2]. Additionally the XPath expressions should always refer to the `<xacml-context:Request>` element as its context node. Thus they will always start with /Request... (c.p. e.g. line 4907 in the core XACML specification). Note that in this manner the profile won't contradict the core specification in line 257-258 anymore and will directly support the new concepts of XACML 3.0 (multiple XXX-Content elements).

3. Line 179-182: interoperability problem
   The relaxation softens the guidelines of the profile. As the profile is mandatory anybody not willing to follow the profile can do so. But having this relaxation in the profile means that one can be conformant to the profile and still represent the identity of nodes in an alternative way which will destroy interoperability. A question related to this fact is the actual intend of the profile: Should it either be a best-practice document how to use XACML for hierarchical resources like XML or/and should it be a standard guaranteeing interoperability when using XACML in such scenarios? From the OWS use case both (at least the latter) would be the actually needed document in order to be able to implement interoperable access control systems for OGC Web Services.

4. Line 185-186: unclear usefulness
   A node in an ACDR is referenced by the resource-id attribute. Should the xml-node-id attribute ever be used? It seems that at least in the OWS use case this attribute is never used.

5. Line 239: unclear usefulness
   How should the xml-node-req attribute ever be used?

[2] See shortcoming 3 in 7.2.1.2 for the reason why to differentiate between the node type.

6. Line 240-243: contradiction (critical)
   On the one hand side the resource-parent, resource-ancestor and resource-ancestor-or-self XACML attributes are optional to implement (c.p. line 242). On the other side the resource-parent, resource-ancestor and resource-ancestor-or-self XACML Attributes shall be present in an ACDR (c.p. line 251). For more comments on these attributes see the next shortcoming – number 7.

7. Line 261-289: unclear usefulness (critical)
   Following the first way of how to represent a hierarchical resource - i.e. as an XML document included in the <ResourceContent> element (c.p. line 87-89) - we don't see any need for the resource-parent, resource-ancestor and resource-ancestor-or-self XACML Attributes in this XML resource case. The same information is already included in the decision request's <ResourceContent> element where the whole XML document is included. The question is therefore why do you have to include more generalized information (you loose the detailed relationship information between the nodes) as explicit XACML Attributes in the same decision request? In our OWS use case these attributes are of no use and forcing to use them is an unnecessary burden.
   Additionally it remains unclear for the reader where they are actually useful. If they refer to the second way (c.p. line 89-91) of how to represent a hierarchical resource (remember: this alternative is not suited for the OWS use case) there should at least be some additional explanations in which situations they might or might not be useful in order to avoid confusion of the readers of the profile.
   Another question is if the introduction of these attributes fit into section "3.1 Nodes in an XML document" or should they be introduced in 3.2?

8. Line 367-379: limited functionality despite the existence of a better solution (critical)
   Given the situation that the value of the resource-id attribute contains an XPath expression that refers to exactly one node of the XML document (included in the <ResourceContent> element), a policy writer can define predicates that apply to one or multiple nodes in a XML document by using the value of the resource-id attribute in an special AttributeSelector. The use of the such a Selector is described by the following example:
   `XMLSelector(string-one-and-only(ResourceAttributeDesignator(resource-id), arbitraryOffset).`
   Additionally an XML attribute of this selector (as usual for XACML AttributeSelectors), is the datatype attribute. Therefore each node (or subtree[3]) selected by this sector will be casted to a datatype and a Bag of this datatype will be returned. This mechanism allows selecting arbitrary node sets (Note that it is not even restricted to text nodes only!) of the hierarchical resource and the resulting bags can in turn be evaluated by all the common XACML Bag (or higher-order bag functions).
   Conclusion: Having a simple XMLAttributeSelector supporting the function

[3] See section 7.2.2.

described above allows policies to evaluate any node(s) in the XML document. This in turn means that such a general, easy to understand and implement and powerful mechanism could replace the less expressive techniques described in section 4.1 and 4.2 of the profile. Of course the xpath-node-match function section as mentioned in 4.2 can still be used but again the mechanism described above is much more expressive. E.g. assume you want to define a predicate in a rule's condition element saying an XML node (e.g. Building) can be accessed only if one if its child element (e.g. owner) equals "Bob" and another child element (e.g. status) equals "for sale". Trying to define predicates that refers to certain nodes in the hierarchical resource is not possible with the mechanisms included in the current version of the profile. The reason for this is that you loose the connection/relation of the nodes being evaluated in separate predicates (e.g. the connection of the attributes owner and status that belong to the same building). Therefore, it is important to add another AttributeSelector making use of the resource-id attribute value in combination with an arbitrary offset.
The example in Appendix A demonstrates how things work with the mechanism described here.

7.2.1.2 Shortcommings in the Multiple Resource Profile of XACML 2.0 and Sugestions

1. Line 136-139: interoperability problem (critical)
   This block is an elegant way of achieving, that a programmer has some degree of flexibility when implementing a Context Handler supporting this profile. At the same time it restricts the flexibility so that the results are the same as if one follows the model as described in the profile.
   From the interoperability point of view it is important that the syntax and semantics of a global and individual ACDR are standardized. This standardization further has a direct influence on the way how to define the policies in XACML. The question that arises here is, does this block really ensure interoperability? If the implementation of a Context Handler does not even construct individual ACDR and uses instead a proprietary way of doing the same, than there is a risk that the policies will be defined differently. One time the rules will correspond to the needs of the proprietary alternative Context Handler implementation and one time they will follow the model described in the profile. This fact therefore results in loosing the interoperability of two XACML policies although they are both XACML and Multiple resource profile conformant.
   Solution: the profile should give additionally standardized guidelines how to define XACML policies that match the derived individual ACDR.

2. Line 145-208 (Section 2.1 and 2.2): redundancy, mixture of guidelines how to deal with XML and non-XML resources (critical)
   In case of XML high-level resources the use of the functionality of scope:xml = Children or Immediate can also be achieved by the alternative described in section 2.2 (i.e. scope=xpath-expression and resource-id is an appropriate XPath

expression; i.e. evaluating to the immediate node or some children nodes). The functionality of “descendant” can’t be achieved by the mechanism in section 2.2 and is therefore also needed for XML resources. It seems that a clearer separation of the XML and non-XML use case (like in the hierarchical resource profile or like in Section 3) might help clarify things.
Solution:
XML resources:
scope = {descendant, xpath-expression} & resource-id
Note: It is an unnecessary restriction that in case of descendant, the value of resource-id has to point at exactly one node (c.p. line 164).
non-XML resources:
scope = {descendant, children, immediate} & resource-id

3. Line 145-208(Section 2.1 and 2.2): limited functionality (critical)
   In case of XML high-level resources, the mechanisms to derive individual ACDR from a global ACDR for multiple resources are too limited. Assume you want to generate an individual ACDR for all **element** nodes that are descendant of the node specified by resource-id. In this case the profile in its current version is imprecise about what descendant actually means. But, are all nodes (i.e. element nodes, text nodes, attribute nodes etc.) the descendants?
   Solution:
   To clarify this either an additional attribute is necessary or you extend the possible scope values by:
   elementNode-descendant, attributeNode-descendant, textNode-descendant, commentNode-descendant,... and all possible subsets of these. The original descendant scope-value will then have the meaning of really all descendant nodes no matter of the node type.

4. Line 174: unclear or imprecise formulation (critical)
   Following the profile an individual ACDR shall have one resource element that refers to a single resource (in case of XML, an XML node). This node is specified by the resource-id attribute in the individual ACDR. Therefore, it is unclear how and why a resource element of an individual ACDR can have more than one resource-id attribute (c.p. line 174: “...at least one..”).

5. Line 191: confusing URN
   Is urn:oasis:names:tc:xacml:2.0:profile:multiple:xpath-expression a good name for the attribute value of scope? In the XACML specification the word xpath-expressions is used either in the meaning urn:oasis:names:tc:xacml:2.0:**profile:multiple**:xpath-expression or in the meaning urn:oasis:names:tc:xacml:2.0:**data-type**:xpath-expression. Without any explicit comment about the difference of the two we encountered, that this naming is confusing for people not very familiar with the specification and the profiles.
   Solution: add a footnote or comment.

6. Line 218: usefulness
   It is unclear where to use the URI identifier urn:oasis:names:tc:xacml:2.0:profile:multiple:multiple-resource-elements. Is it just useful as kind of metadata in a description of an access control system product? If so a little note of the intended use of this URN might be helpful to clarify things.

7. Line 223: overloaded and therefore misleading formulation (critical)
   The term individual resource request is used for two different things. In the whole profile it represents generated individual ACDRs depending on the individual resources as identified by scope and resource-id attribute values of the original global ACDR for multiple resources. In this section the term individual resource request is used for requests that are derived dependant on the number of <Request> elements.
   Solution: Reformulate section 2.3.3 or replace individual by separate in Line 223, 225, 226, 228 and the first occurrence in line 230. Additionally it might be helpful to add a short definition or explanation of this matter.

8. Line 209-234: Comment
   Having the XML resource use case in mind we are wondering if there is a real use case or benefit by using multiple <Resource> elements. Basically what happens is that a PEP packs more than one XACML request element in one decision request and the corresponding Context Handler immediately un-packs this “combined” multiple resource request and generates separate global request contexts one for each request element. Is this an advantage compared to sending the global decision requests one after the other? Are there use cases where it does make sense to pack more than one global request in one “combined” multiple resource request?

9. Line 235-283: bad solution and writing error (critical)
   a) bad solution: entire-hierarchy:xml should not be a possible value for the scope attribute:
   The scope attribute is not the right place to specify whether the results should be combined to a single response. The scope attribute is used to specify the different ways how to process decision requests for multiple resources. The desire of combining individual results to one single result has only indirectly something to do with this (more precisely: it is just a prerequisite that the original corresponding global ACDR is a decision request for multiple resources). It might for example be the desire of a user that he wants to use option 2 (i.e. 2.2 Nodes identified by XPath→scope:xml=...multiple:xpath-expression) and to union all individual responses to a global, single combined response. Here the user is lucky as this is the special case intended by the profile (at least more or less – c.p. next problem item 9 b). What if the user wants to use option 1 (i.e. 2.1 Nodes identified by “scope”→scope:xml=Children) and wants to get the union of all individual responses to a global, single combined response. In this case the scope attribute should be on the one hand side Children but on the other entire-

hierarchy:xml. Because of this source of conflict and unnecessarily reducing flexibility we would recommend to use another attribute (e.g. combine-results-of-a-multiple-request of datatype Boolean) that acts as a switch and helps the Context Handler to decide whether to union all individual responses to a global response or to forward the individual responses directly.
b) writing error:
We strongly assume and recommend that the intended meaning of line 269 is like described in section 2.2. That means that the line 268, 269 should say:
"…such that the <AttributeValue> evaluates to a node set that represents **multiple *nodes* in the <ResourceContent> element."** (instead of:…"such that the <AttributeValue> evaluates to a nodeset that represents ~~exactly one ***node*** in the <ResourceContent> element)~~. This correction will achieve consistency with the semantic description in line 272, 273. Additionally it might be more precise to also explicitly name that the multiple <Resource> element case can also occur in combination with such a request and will upfront be processed as described in section 2.3.
In summary:
Input:
Baseline is the receive of a global ACDR for multiple resources (c.p. 2.1 or 2.2) or a combined global ACDR for multiple resources (i.e. multiple <Resource> elements that in turn represent the significant part of multiple resource requests).
Output:
Step 1: derive the individual ACDR according to 2.1-2.3
Step 2: combine the individual response to a global response according to line 243-252 (see problem 10 for a comment on this combining algorithm).
A rewriting of section 3.1.3 after eliminating the error in Section 3.1.2 seems to be most appropriate.

10. Line 248: intended functionality?
    In case one individual result is indeterminate, should the combined result really be deny or maybe indeterminate?
    Solution: More precise explanations how to combine the results in case of errors have to be added to the specification.

11. General comment: (critical)
    All XPath expressions that are values of resource-id should follow the naming as introduced in (7.2.1.1 item 2). Additionally as mentioned in 7.2.1.1 under item 2 they should start with "/Request..." as only this follows the core XACML specification and furthermore this will contribute to uniqueness/interoperability and will simplify further extensions that might allow multiple variants of <ResourceContent> elements (e.g. <EnvironmentContent> element – c.p. 8.1.2)

12. General comment:
    What is the intention of sections referring to non-XML resources? From the OWS use case point of view the reason for a profile/specification is to achieve interoperability and clear rules how to do something. When critically examining the sections of the profile about non-XML resources, all it says is that however

the hierarchical resource is represented and however the individual resources are identified, use the scope attribute to choose between option 1,2,3 and the resource-id attribute to help identify the individual resources. Clearly, that is some sort of guideline but is it of any use? It is easy to imagine how different the policies and request contexts (e.g. the scope and resource-id values) could look like following this guideline. Having in mind that the access control process will only show the intended behavior if and only if the request contexts fit to the policies one can image that so much freedom will hardly result in interoperable solutions. The fuzziness becomes apparent if you compare how much more precise you can and have to be in the XML resource use case. What are the use cases where you use the multiple and hierarchical profile for hierarchical non-XML resources at that abstract level and what benefits result from that? Might it be helpful to separate the XML resource use case in completely separate specifications and afterwards derive an abstract profile (similar to an abstract super class in an object oriented model?). The OWS use case requires an (OGC) Web Services profile for GeoXACML that would ideally be build on top of a profile for XML resources of XACML (or alternatively on top of the XML resource relevant sections of the updated multiple and hierarchical resource profile of XACML). See section 10.2 for further thoughts in this direction.

7.2.2 Shortcommings in the XACML 2.0 Specification and Suggestions

In this section we will list some ideas for a change request of XACML 2.0 that resulted from our analysis of how to use XACML to protect OWS. Note that these ideas will enable a much more powerful use of XACML when used to control access to OGC services.

When using the <ResourceContent> element to represent Web Service specific information (i.e. the WS-request or -response) the current XACML AttributeSelector definition is not sufficient (and not exactly defined in the current version of the XACML 2.0 core specification - c.p. page 66 and 78/79 in [1]). As introduced and motivated under 7.1.1.2 and 7.2.1.1 (item 8), in the OWS use case we need an XMLAttributeSelector that returns arbitrary node sets as specified by the XPath expression within the selector (e.g. a sub tree of a WFS response representing the geometry of a building).

The AttributeSelector mechanism as currently described in the specification is only intended to select text nodes (even if this is not specified unambiguously in the specification – c.p. page 66 and 78/79 in [1]). In our use case it is not sufficient and an unnecessary restriction to return text nodes only.

Note that such an XMLAttributeSelector, even returning the root of sub-trees of nodes, does not necessarily imply that XACML needs to be augmented by a new data type like urn:...:xml or urn:…:elementNode(set), even if this seems to be a good way to follow. For example in the GeoXACML use case we have the datatype geometry that uses a

XML sub-tree (specified through its root element node) as input for its constructor. This means that if the datatype attribute of the XMLAttributeSelector is geometry and the nodeset will never occur explicitly as data object. It will instead be directly casted to the geometry datatype. Nevertheless an XACML attribute of type elementNode might be a conceptually cleaner solution.

Additionally it would be a powerful feature to allow a XACML AttributeSelector or the additionally needed XMLAttributeSelector to pass the XPath expression as a parameter. Combined with a concatenate function one could do things like: XMLAttribute-Selector(concad(resource-id, relativePathFromTheCurrentAC-NodeAs-SpecifiedBy-Resource-id). This is needed to allow rules based on ResourceContent of the form: permit access to building node if the building price is >100 and owner = Bob. If you use two different selectors to implement this semantic you loose the connection of the two attributes that belong to the same building. Solving this issue though predicates within the XPath expression has limitations because of the limited set of functions you can use in XPath predicates.

Another new feature that seems to be a promising and a profitable extension of the XACML language is to allow XML encoded context information under the environment element of an ACDR. At the moment you can only define XACML Attributes under the <Environment> element in an ACDR that offers -- for the reasons as explained under 7.1.1.1 -- only very limited functionality. In order to model more sophisticated application environment information in an ACDR, it would be very helpful to extend the XACML Environment element similar to the Resource element. Imagine an EnvironmentContext Element under XACML request's Environment elements and the corresponding XMLEnvironmentAttributeSelector. This is a consequent extension of what XACML currently provides for resources and is very useful in the OWS use cases (e.g. access semantics on a current disaster situation expressed through a XML document). Section 8.1 will explain this topic in more detail in the context of the requirements of the OWS-6 GPW airport scenario.

7.3 Access Control and the Multiple Protocol Encodings of OGC Web Services

Another important aspect that was indentified during the analysis of how to use XACML or GeoXACML to control access to OWS are problems, occurring due to the different protocol encodings supported by some OGC Services.

The authorisation semantics for an OWS are usually independent of the used protocol encoding. The same access rights need to be fulfilled no matter if the user sends a key-value-pair (KVP) encoded request or a corresponding XML encoded request.

As discussed in the previous sections, OWS specific information should be represented under the <ResourceContent> element. This works fine if we have XML encoded OWS requests and responses. There are OWS that support requests in a KVP encoding next to

the XML encoding (e.g. WFS). In these cases the corresponding specification describes how to represent a KVP encoded request in an XML encoding[4]. It is therefore always possible to transform a KVP encoded requests into an XML encoded representation that is semantically equivalent. Having this situation it is always possible for a PEP to include a XML encoded representation of the OWS request in the ACDR under the <ResourceContent> element. If we can assume the existence of KVP and XML encodings and a transformation from KVP to XML, than the access control rules will always have the same syntax independent of the actual encoding of the submitted OWS request. The resulting strong advantages are apparent and therefore such a transformation should always take place.

It is important to mention that it is not advisable to use the KVP representation as "normal form". Doing so will necessarily result in the corresponding XACML Attribute representation and therefore significantly reduce the possible authorization semantics. The problems one encounters when using XACML attributes is similar to the problems that occur when using the KVP encoding (c.p. 7.1.1.1). The KVP encoding implies the encapsulation of actually more fine-grained and structured information. From the access control perspective the values form the new atomic information entities. As they are much more coarse-grained as they actually are, one loses a lot of expressiveness when defining access rights. If we take for example an ordinary WFS getFeature request with a <Filter> expression. Encoding this request in KVP implies values that are actually XML encoded (e.g. Filter = <Filter….><within>…). All filter information will be represented as one XACML attribute and hence be a single atomic information block. This reduces the possible authorisation semantics. Think of an access right allowing to define spatial filters in WFS requests if the SRS of the used geometries within the filter is WGS 84. This authorization semantic is not expressible anymore as the SRS XML attribute information cannot be selected. Obviously this is just one example but it explains the core of the problem behind the KVP encoding.

The following table summaries the different possible cases.

Supported normative encodings	Consequence for the PEP
XML only	include as-is in the ACDR under <ResourceContent>
XML or KVP	if XML then include as-is in the ACDR under <ResourceContent> else if KVP then transform to XML and include in the ACDR under <ResourceContent>
KVP only	then transform to XML include in the ACDR under

[4] Note that it still needs to be proven that each OGC specification providing KVP and XML request encodings really uniquely (directly or indirectly) describes normative bijective transformation rules.

	<ResourceContent>

Table 5: Transformations of OWS requests in the PEP depending on the used encoding

As you can see in the table above case 3 (row 3) needs a transformation to XML in order to support maximal expressive authorisation semantics. To the best of our knowledge the only OWS that falls into this category is the WMS. The WMS standard doesn't define a normative XML schema for its interfaces. The lack of this normative schema definition gives some flexibility in how to transform the KVP request into an XML encoding. This in turn, as explained in section 7.1, can result in interoperability problems as the policy writer doesn't know the encoding of the WMS request in the ACDR and therefore has different alternatives how to define the corresponding access rules. Because of the strong dependency between the ACDR syntax und semantics and the rules without clear guidelines how to transform a KVP encoded WMS request into an XML encoded request, there is a risk of loosing interoperability. To solve this problem the WMS standard (e.g. in version 1.4) should define a normative XML encoding for its request interface to allow for a standardized, bijective transformation between KVP encoded and XML representations of WMS requests (see 10.2 for further discussion on this issue). Note that we regard a special case solution that uses XACML attributes to encode the KVP pairs directly in an ACDR in the WMS use case as not appropriate. Such a special treatment for one service (that lacks the XML definition for unclear reasons) is confusing and more importantly contradicts a uniform, general use solution with optimal expressive power (c.p. 7.1.1.2).

8 Access Control with GeoXACML in the GPW Airport Scenario and for OWS

In this chapter we will show how to implement the needed authorization semantics for the OWS-6 GPW airport scenario and how GeoXACML can be used in general to control access to OWS. Section 8.1 discusses how to express the context dependant rules of the OWS-6 GPW airport scenario. Chapter 8.2 will demonstrate how to use GeoXACML to express access control rules for OWS. Section 8.2 will also focus on how to implement the needed authorization semantics for the GPW airport scenario. Section 8.3 will list further example rules that demonstrate the capabilities of GeoXACML to protect OGC Web Services.

8.1 Context Dependant Rules in the OWS-6 GPW Airport Scenario

One special requirement in the GPW airport scenario is the support of access control rules dependant on the current disaster situation. The access control system should for example allow the definition of access control rules expressing rights like, permit first responders access to building feature data if the building is within a certain distance of a current disaster location and not exceeding a given security marking. The special characteristic of these context dependant rules is that in this case the context consists of complex spatial objects (e.g. incident objects with location attributes). In order to support these spatial context dependant rules we identified two different approaches that will be described in the next two subsections.

8.1.1 Hardcode disaster state information in the GeoXACML rules

The idea behind this solution is to transform the actually spatial context dependant authorization semantics into content dependant rules. To do so, the rule has to contain the constants that represents the needed information from the current spatial context. The following abstract example illustrates the general idea of hardcoding the disaster state information in the rules:

(+, FR, read, BuildingData, if

/building/location is-within-distance 10, **incidentLocationA** or

/building/location is-within-distance 10, **incidentLocationB** or

/building/location is-within-distance 10, **incidentLocationC**

)

Note that incidentLocationX is a constant value representing the current location of a incident. Trough the hardcoded context information the actually context dependant semantics are expressed though a content dependant rule.

The advantage of this approach is that the semantics of the access control policy is completely encoded inside the GeoXACML rules and therefore supports audit at any point in time, as the rules are time invariant and self-contained. It also is benifisher if the policy is to be exchanged with other authorization domain, in particular of other jurisdictions. The disadvantage of this approach is that all rules relating to the state of the disaster have to be updated after every disaster state transition. Imagine the fire at incidentLocationA was extinguished and the fire at incidentLocationB became much bigger. This means that the hardcoded spatial constants in the rule above expressing the context information have to be updated correspondingly in order to enforce the authorization semantics after the change of the disaster situation. In case you are using a powerful PAP supporting intelligent verification of the policy, after each update of the disaster state dependant rules, the policy has to be revalidated.

8.1.2 GeoXACML Rules Reference the Externally Modelled Disaster State Information

This approach peruses the idea that the spatial context information (i.e. the disaster state information in the airport scenario) is modeled and represented externally (from the policy point of view). The rules just contain generic references to this externally modeled context information objects. Thus the difference to the approach introduced in the section above is that no spatial context information is hardcoded inside the rules.

Following this approach there are two ways how to implement and reference the externally modeled context information.

Option 1:

One option could be to retrieve the externally modeled disaster state information through a special ExternalDataSelector mechanism that can be used inside the rules. The abstract examples below show how this selector mechanism could look like in cases where the context information is stored in a WFS or DB.

Example 1: context information is stored in a WFS

```
<ExternalDataSelector>
 <DataSource>WFS_XXX</DataSource >
 <Request>
  <wfs:GetFeature service="WFS" version="1.1.0" …>
   <wfs:Query typeName="myns:incidents">
    <wfs:PropertyName>myns:location</…>
   </wfs:Query>
  </wfs:GetFeature>
 </Request>
</ExternalDataSelector>
```

Example 2: context information is stored in a DB

```
<ExternalDataSelector>
 <DataSource>ODBC_entry_4_oracleDB</DataSource >
 <Request>
   select a.shape.sdo_polygon from incident_table a
 </Request>
```

```
</ExternalDataSelector>
```

The disadvantage of this option is obvious. Every time a rule matches, this sub-request (i.e. the dereferencing of the ExternalDataSelector) has to be processed. Obviously this will result in a very poor performance and a single point of failure as the process of deriving the authorization decision depends on the proper functioning of the DB/WFS.

Option 2:

An improved option to implement and reference the externally modeled context information that also guarantees up-to-date state information during rule evaluation is to represent the context state information in the ACDR. In this case the PEP or Context Handler inserts an <EnvironmentContent> element under the XACML environment element in the ACDR similar to the <ResourceContent> element used to represent an OWS request or response. The needed spatial context information is included in the ACDR as child element of the <EnvironmentContent> element.

Note that using XACML attributes under the environment element to represent the spatial context information will imply all the disadvantages that appear when using XACML Attributes. Hence XACML attributes are not usable in this case for the same reasons as explained in section 7.1.1.1.

Using an <EnvironmentContent> element to represent arbitrary complex spatial context information in an ACDR with a corresponding AttributeSelector eliminates the disadvantages of option 1. External context information has to be retrieved by the PEP or Context Handler at most only once per ACDR and not every time, a spatial context dependant rule gets evaluated (details see 8.1.3). Furthermore option 2 provides a very flexible and expressive approach to express context dependant rules. Arbitrary, semantically rich, XML schema based, XML encoded context information can be represented in an ACDR. In addition this information can be flexibly referenced by XPath expressions resolved through a corresponding AttributeSelector inside the rules. Altogether this allows for the definition of arbitrary complex spatial context dependant rules. Note that compared to the hardcoding approach the rules do not have to be updated when the disaster situation changes but it has to be ensured that the injected values are associated to the current state. It should further be mentioned that these kinds of context dependant rules have only references as function parameters and therefore no rule literals/constants. This fact might result in limited policy validation possibilities.

In order to guarantee interoperability when supporting the spatial context dependent rules and thus get the enhanced expressive power when defining rules, the extension of XACML by the <EnvironmentContext> element and a suitable AttributeSelector has to be standardized. The good news here is that work is in progress to incorporate these ideas in the new version of XACML 3.0 (c.p. 10.2) are ongoing.

8.1.3 Conclusion

The discussion in the last two sections unveiled that the needed spatial context dependant rules in the airport scenario can be supported either through the approach of hardcoding the disaster state information in the GeoXACML rule or by referencing the externally modeled disaster state information in the rules.

Following the first approach (approach 1) the rules itself have to be updated when the disaster situation changes. In the latter approach (approach 2) the updated disaster situation is incorporated by the PEP or Context Handler.

It dependants on the concrete implementation how and how often the update of the rules (approach 1) or the <EnvironmentContext> element (approach 2) is performed. In approach 2 for example it is feasible to query the environment context information from a repository again for every ACDR or in regular intervals. Alternatively the update could be initiated by the repository or the software responsible for the administration of the environment context information. Assume e.g. a disaster management system as the administration software of the current disaster situation. The operator of such a disaster management system could initiate an update message to the access control system (i.e. to the PEP or Context Handler in approach 2 or to the PAP in approach 1) every time the disaster situation was updated. Figure 3 shows how the configuration phase of the access control system in order to correspond to the current disaster situation could look like.

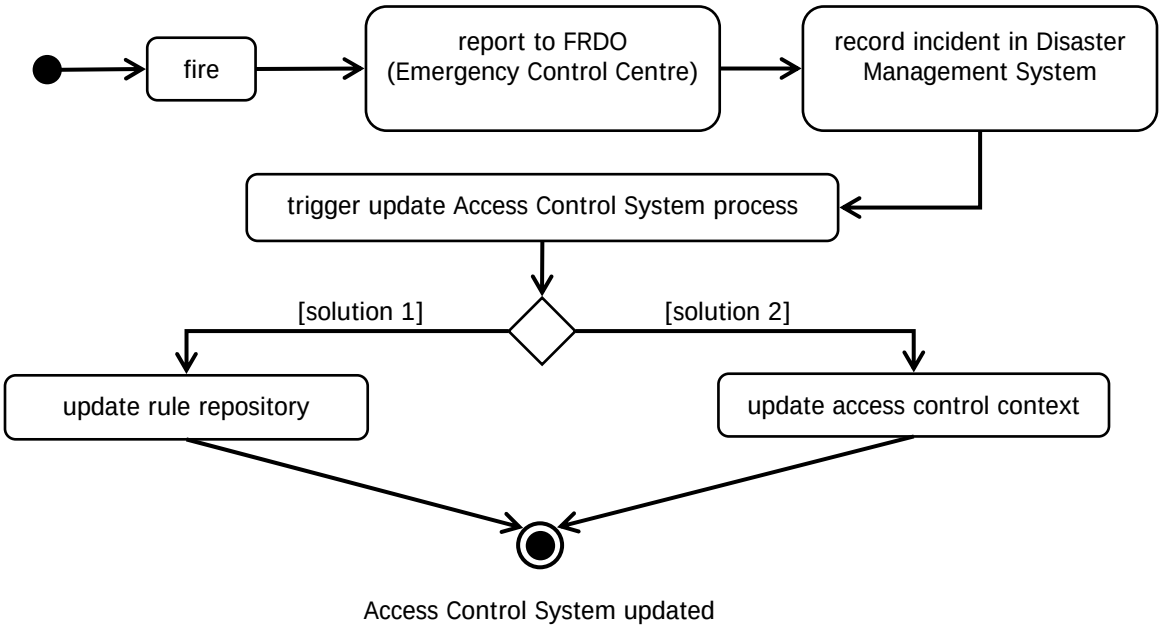

Figure 3: Configuration phase of the access control system after disaster state change

As one can see, both approaches have its different favors depending on the actual use case. As approach 2 is currently not yet standardized we decided to use approach 1 in the GPW airport scenario to express the spatial context dependant rules.

8.2 GeoXACML in the OWS-6 GPW Airport Scenario

This section will describe how the needed authorization semantics of the GPW airport scenario can be implemented through GeoXACML rules.

8.2.1 Use Case Description

Security Domains

1) LA (local authority domain)

2) AA (airport authority domain)

3) FL (federal level domain)

Players

1) First responders (FR)

2) First responders Dispatch Office (FRDO)

3) Airport Authority (AA)

4) Federal Level (FL)

Data Sources

City level data (city-wfs)

http://services.interactive-instruments.de/ows6/cgi-bin/ows6city-wfs.exe

Airport Authority data (airport-wfs)

http://services.interactive-instruments.de/o ws6/cgi-bin/ows6airport-wfs.exe

Use Case Description:

First responders (FR) sent to a building fire, they receive basic map coverage (WMTS) and city level 3D vector coverage (WFS). Once at the fire they determine the need for more detailed building coverage. They also determine wind conditions are such that heavy smoke is being pushed in the direction of the adjacent airport. They determine the need for more detailed airport data, up to date weather data and imagery coverage. First responders dispatch office (FRDO) searches (catalog) for building data, airport data, weather data and imagery. AA has detailed airport coverage, weather data, and plume generation service. FL has detailed building data and imagery (both WCS and JPIP-WCS) services.

Attempts to access directly both the Airport Authority (AA) and the Federal Level (FL) data holdings are denied because of access restrictions. (1)

FRDO then contacts AA and FL to request access to data holdings. Direct access to FL building data denied but AA agrees to act as intermediary. FL allows JPIP streaming of georeferenceable imagery data to the FRDO.

FL considers AA to be a separate but trusted domain and accepts the AA request for building data. (2)

AA accepts FRDO request for detailed building data. (3)

AA provides FDRO temporary access to their weather service and plume processing service.

AA provides full access to airport data which falls within 200 meters of the building fire location. (4)

AA provides limited access to remaining airport data out to a distance of 500 meters based on attribute level security marking tags (anything above unclassified is restricted). (5)

After review of AA data FRDO determines fuel tanks on the airport property exist within the 200 meter "hot zone" which present a potential hazard. FDRO requests georectified imagery over the area to set up a safety perimeter.

8.2.2 The Security Architecture in the GPW Airport Scenario from the Access Control Perspective

Figure 4 shows the components of the IT architecture of the airport emergency response scenario from the access control perspective.

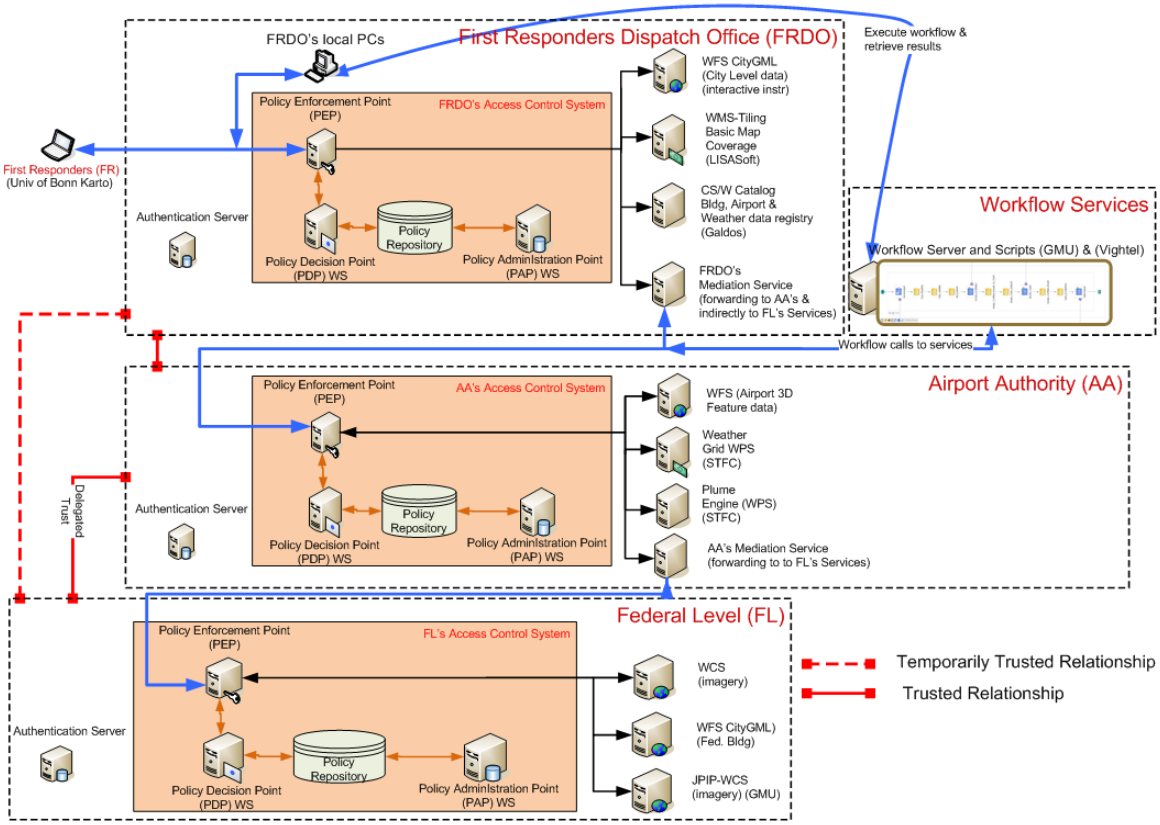

Figure 4: Access Control in the OWS-6 GPW Airport Emergency Response Scenario

The list below explains the access control relevant workflow within this architecture:

- A First Responder (FR) wants to access FL's WFS.
- The First Responder authenticates itself to his identity provider through an Authentication Server. It is not our work item to analyse the different ways how this can be done and which of these options are suitable for OWS based infrastructures. We assume a mechanism similar to the SAML single-sign-on approach for each domain. Hence the user, the first responder, gets back an security assertion that will be forwarded with the WS-requests so that other service providers, if confident of the origin of the assertion, can trust that the user authenticated successfully somewhere else and that he is who he pretends to be. Note that from the access control perspective it doesn't make a difference if the authentication information or other security relevant information is a SAML assertion or maybe a X.509 certificate or comparable assertions. The only important requirement is that all access control relevant security information (like the user-id, the authentication method, the activated roles and so on.) is stored or transmitted in a semantically well defined and structured data object, and that this data is accessible or dereferenceable by the PEP (through a resolving mechanism provided between the PEP, the Authentication Server and maybe the user.). To simplify the description of the access control relevant workflow we assume that

the user authenticated somewhere the access control system trusts and that he provides a security assertion together with its WS request.

- After requesting the FRDO Catalogue Service he retrieves the URL of a virtual FL_WFS_in-FRDO Server in the FRDO domain that is represented by the FRDO Mediation Service. This server acts as a proxy and will simply forward all requests to another service – here to the virtual FL_WFS-in-AA (the AA_Mediation Service) in the AA security domain.

- The FR submits a getFeature request to the virtual FL_WFS_in-FRDO (i.e. FRDO's Mediation Service).

- This request is intercepted by the PEP of FRDO's access control system and checked for sufficient access rights. Note that although the service is actually in the FL domain, FRDO can add local access control checks. The intention behind these checks is to verify if the request of FR conforms to FRDO's local policy. If so, FRDO can be certain that it is valid according to its policy to forward the request to FL over AA on behalf of FRDO.

- The transformation of the identity bound to the WFS request is performed in the Mediation Service after it passed FRDO's PEP. The Mediation Service will forward the requests to the virtual FL_WFS_in-AA (the AA_Mediation Service) in the AA security domain. The Mediation Service is only exchanging the identity of the requestor from originally FR to FRDO. FRDO allows to forward requests under its name on behalf of FR if they fulfil FRDO's local access control policy.

- In AA's security domain a similar processing occurs and in the end FR's getFeature request will arrive at FL's WFS, processed accordingly and the response will be returned through the same chain of intermediaries (i.e. FL's PEP, AA's Mediation Service, AA's PEP, FRDO's Mediation Service, FRDO's PEP, FR client). The FR user will then receive the WS response or the intersection of requested and accessible features (according to FL's, AA's and FRDO's policy).

Note that the WFS response also passes the access control systems in each security domain. This is necessary as based on the request not all access rights can be verified. This means that after the pre-processing access control step (i.e. access control on the WS-request) a post-processing access control step (i.e. access control on the WS-response) is necessary. Again it should be noted that the access control systems in each security domain allow expressing local access policies on top of the policies enforced by the domains below.

The advantage of the approach taken here (i.e. cascading access control systems) is that each security domain just needs to maintain its local policies for the entities known in its security domain. In the scenario for example FL's access control rules just refer to the AA role. FL doesn't need to know any other users (e.g. the FRs or FRDO users). Hence there is no need to have access control semantics in FL's access control system that refer to entities of other security domains or that come from the AA or FRDO security domain (as it would be the case if you just use one global

access control system for the whole scenario). This clearly reduces the administrative burden in each security domain as the principle of divide and conquer is applied and directly related domains trust each other. That means that FL allows the AA role to perform access on their WFS under certain conditions (FL's access control policy). Further the agreements between FR and AA allow AA to delegate certain subsets of their rights to their own clients (the FRDO level). AA's local policy is used to control which of their clients and how they are allowed to act in their name on FL's service. Through the cascading access control systems from the different security domains a flexible delegation of rights and arbitrary subsets of rights can be achieved next to the mentioned encapsulation of each security domain.

8.2.3 Resources to be Protected and their Characteristics

The airport-wfs serves different feature types of AA. The features have the attribute icism:classification that can have the value S (secret) or U (unclassified). Also, each feature has a geometry property gml:boundedBy that expresses the extension of the building. For the use case "fire at the airport" we need to know the location of the fire, the perimeter 1 (diameter approx. 2.3km) and perimeter 2 (diameter approx. 5km) as well as the unclassified and secret building locations.

Building S-142 (fire location): Lat: 29.963745015416 Long: -90.029951432619

Building S-43 (secret): Lat: 29.955591586530 Long: -90.041238946616

Tower P-117 (secret): Lat: 29.962934206895 Long: -90.040554658313

ControlTower P-114 (secret): Lat: 29.961663017762 Long: -90.047710997151

Building P-122 (unclassified): Lat: 29.980976028057 Long: -90.055682191645

Building S-42 (unclassified): Lat: 29.953919753173 Long: -90.038718605622

Building S-48 (unclassified): Lat: 29.965441464116 Long: -90.052766143423

Building S-50 (unclassified): Lat: 29.960640507207 Long: -90.046099218309

Building S-51 (unclassified): Lat: 29.959687658430 Long: -90.045929305801

Building S-52 (unclassified): Lat: 29.956665408509 Long: -90.044013002542

Building S-54 (unclassified): Lat: 29.956724022722 Long: -90.047227675901

Building P-129 (unclassified): Lat: 29.936454859858 Long: -90.011186413724

Distances between the fire location and the secret buildings/towers:

Building S-142 to Building S-43: 1.42 km

Building S-142 to Tower P-117: 1.00 km

Building S-142 to Tower P-114: 1.73 km

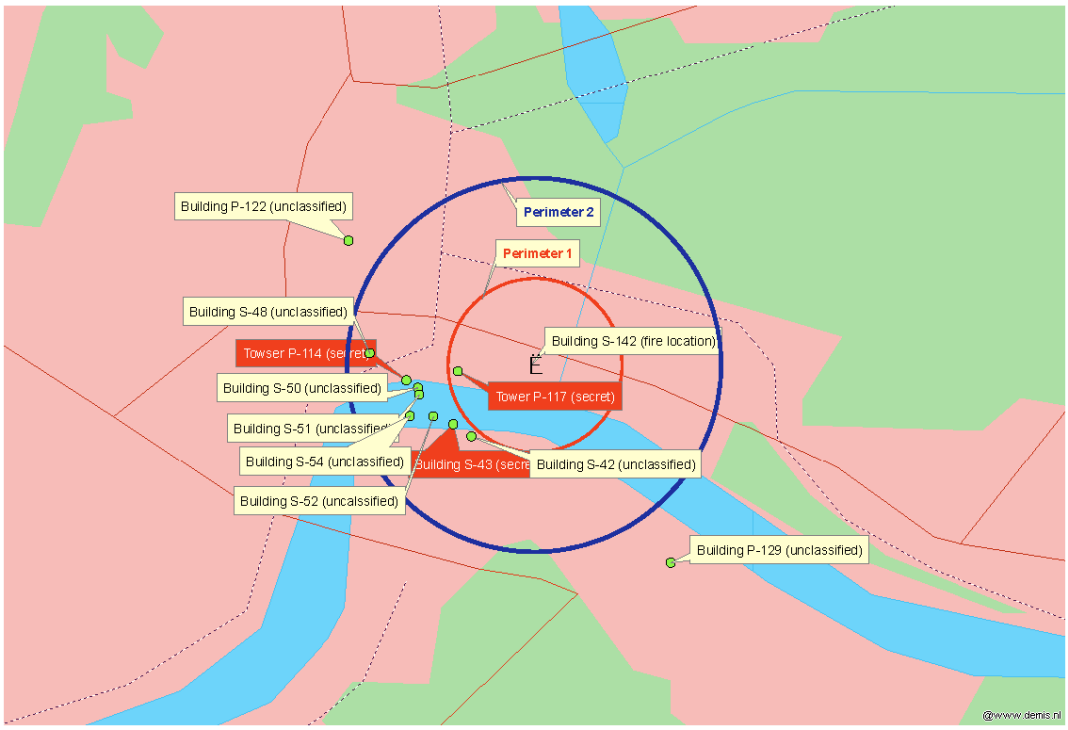

Figure 5: Location of buildings and perimeter geometry

Diameter for Perimeter 1: 2.3 km (radius 1.15km)

Diameter for Perimeter 2: 5.0 km (radius 2.50km)

8.2.4 Access Rights for the Airport Authority WFS (airport-wfs)

- Role FR, FL access to WFS is denied - derived from (1)
- Role FRDO access to WFS is permitted if features of any type are within perimeter 1 (valid for all feature types and independent from the classification) - derived from (4)
- Role FRDO access to WFS is permitted if features of any type are within perimeter 2 and the classification is U (not "S") - derived from (5)

8.2.5 Policy Test Suite

The above access rights for the AA domain are declared in AA.geoxacml that is included in the ZIP File that is distributed with this document. (c.p. 8.2.6). In order to determine the correctness the following test cases can be defined and are to be executed against the policy.

Basically, we have to determine unclassified and secret features and their distance to the fire. Iterating through the possibilities unveil the test cases:

Test Case #	Result	Description	Features
1a	Permit	ALL features are within the perimeter 1	Tower P-117
1b	Deny	At least one feature is outside the perimeter 1	Tower P-117, Building S-43
1c	Deny	ALL features are outside perimeter 1	Tower P-114, Tower P-117

Table 6: Test Case #1 - <wfs:FeatureCollection> does contain classified features only

Test Case #	Result	Description	Features
2a	Permit	ALL features are within the perimeter 2	Building S-51, Building S-52
2b	Deny	At least one feature is outside the perimeter 2	Building S-51, Building P-129
2c	Deny	ALL features are outside perimeter 2	Building P-122, Building P-129

Table 7: Test Case #2 - <wfs:FeatureCollection> does contain unclassified features only

Test Case #	Result	Description	Features
3a	Permit	ALL secure features are within perimeter 1 and ALL unclassified features are within perimeter 2	Tower P-117, Building S-50
3b	Deny	At least one secure feature is outside perimeter 1 and ALL unclassified features are within perimeter 2	Tower P-117, Building S-43, Building S-50
3c	Deny	ALL secure features are outside the perimeter 1 and ALL unclassified features are within perimeter 2	Tower P-114, Building S-43, Building S-50
3d	Deny	ALL secure features are within perimeter 1 and at least one unclassified feature is outside perimeter 2	Tower P-117, Building S-50, Building P-122
3e	Deny	ALL secure features are within perimeter 1 and ALL unclassified features are outside perimeter 2	Tower P-117, Building P-129, Building P-122
3f	Deny	At least one secure feature is outside perimeter 1 and ALL unclassified features are within perimeter 2	Tower P-117, Building S-43, Building S-50
3g	Deny	At least one secure feature is outside perimeter 1 and at least one unclassified feature is outside perimeter 2	Tower P-117, Building S-43, Building P-129, Building S-50
3h	Deny	At least one secure feature is outside perimeter 1 and ALL unclassified features are outside perimeter 2	Tower P-117, Building S-43, Building P-129, Building P-122
3i	Deny	ALL secure features are outside perimeter 1 and ALL unclassified features are within perimeter 2	Tower P-114, Building S-43, Building P-129, Building P-122

3j	Deny	ALL secure features are outside perimeter 1 and at least one unclassified feature is outside perimeter 2	Tower P-114, Building S-43, Building P-129, Building P-122, Building S-48
3k	Deny	ALL secure features are outside perimeter 1 and ALL unclassified features are outside perimeter 2	Tower P-114, Building S-43, Building P-129, Building P-122

Table 8: Test Case #3 - <wfs:FeatureCollection> does contain a mix of unclassified and secure features

8.2.6 Reference to the Implementation of the GeoXACML Rules of the Airport Scenario

Please find the GeoXACML Policy that declares the above access rights and a corresponding ACDRs for the "fire at the airport" scenario in the ZIP File that is distributed with this document. Adding it to the appendix of this document was not reasonable due to the size of the policy and the corresponding demo ACDRs

The spatial constraints are based on geometries that approximate the perimeter 1 and 2 circles as a GML polygon. Further the following is assumed:

1. The Authorization Decision Request (ADR), issued by the PEP, does contain the role's value inside the XACML Attribute "urn:oasis:names:tc:xacml:2.0:subject:role" . The possible values are:
    a. "FR" reflecting requests from the first responders
    b. "FRDO" reflecting requests from the dispatch office
    c. "AA" reflecting requests from the airport authority
    d. "FL" reflecting requests from the federal level
2. The response from the WFS (the <FeatureCollection>) is contained in the <ResourceContent> element of the ACDR
3. The "urn:oasis:names:tc:xacml:1.0:resource:resource-id" contains the value of a XPath expression that points to the features to be determined by the PDP

8.3 Example Rules Demonstrating the Expressiveness and Capabilities of GeoXACML

As mentioned in 6.4 GeoXACML is a very powerful language and allows expressing very complex authorization semantics. Therefore it is difficult to give a comprehensive overview of what kind of access rules can be expressed through GeoXACML. Nevertheless the following examples try to give an impression of the expressive power of the language in the OWS context.

This section is structured according to different types of access control rules. If applicable each such section is further divided into two subsections showing the application of the rules for two different kinds of services – a Web Feature Service and a Web Map Service. To cover a broad spectrum of examples the rules referring to Web Feature Services are pre- and post-processing rules and the rules for Web Map Services are as a matter of fact of course pre-processing rules.

The following listing will start with some very basic rules and more sophisticated rules will follow. To keep the examples as simple as possible we have abstained from extensively defining rules combining the various different rule types. The reader should therefore keep in mind that rules of different type can be combined arbitrarily. Additionally it should be noted that we tried to define simple, self-explaining rules that nevertheless give an understanding of the versatility of GeoXACML.

A highly simplified pseudo syntax is used to express the access control rules in the following examples in order to keep them as short as possible. In Appendix A the reader can find two fully GeoXACML conformant rule examples and an example of a corresponding valid XACML access control decision request. The semantics of the used pseudo syntax is described by the following tupel:

(Effect, subject/role, expression specifying a node-set, condition).

8.3.1 High-level Rules Referring to Subjects, Roles, Security and Transport Information

1. (-, Junior Employee, any-node)
2. (+, staff, any-node, if AuthenticationMethod = SAML),
   (-,staff, any-node, if AuthenticationMethod = Username/Password)
3. (+, staff, Service-IP = 141.48.116)

8.3.2 Context Dependant Rules

1. (+, staff, any-node, if access between 8am and 5 pm)
2. (-, staff, any-node, if ACS-Environment.State equals “disaster level 2”)
3. (-, staff, any-node, if user submitted more than 100 requests today)

See [2] section 10.2.8 for a comprehensive list of functions that are supported to define context dependant conditions in rules.

8.3.3 Rules Referring to Arbitrary Nodesets of Different Node Type

8.3.3.1 WFS

Rules referring to **element** nodes:

- pre-processing:
  1. (+, staff, /GetFeature)
  2. (-, staff, /Transaction/Delete)
- post-processing:
  3. (+, staff, /FeatureCollection/FeatureMember/Building),
  4. (-, staff, /FeatureCollection/FeatureMember/Street)
  5. (-, staff, /FeatureCollection/FeatureMember/Building/Owner)
- hybrid (i.e. fusion of pre- and post-processing rule):
  6. (+, staff, *WFS*) = (+, staff, /GetFeature or /Transaction or GetCapabilities or /FeatureCollection or…)
     Note: this rule is an example for a rule referring to XML nodes in the WS-request and WS–response. The nodes must be an indicator, that the request or response refers to the corresponding service – here WFS. For a WFS the expression specifying the matching node-set has to reference all the possible "operation type" elements in requests and the possible return elements in the WFS responses.

Rules referring to **attribute** nodes:

1. (+, staff, /FeatureCollection/FeatureMember/Building/Geometry/@srs),
2. (-, staff, /FeatureCollection/FeatureMember/Building/@FeatureID)

Rules referring to **text** nodes:

1. (-, staff, /FeatureCollection/FeatureMember/Building/Owner/text())
   Note: The difference between this rule and the corresponding element rule (see No. 5 above) is that here the element node <Owner> will be in the response set and access is only denied to the child text node of the <Owner> element.

Note: According to [1], [3] and [4] regular expressions based on the XPath syntax, that match the access control decision request's resource-id attribute are used to refer to a specific node-set (e.g. reg-exp-match(resource-id, /FeatureCollection \[\d+\]/FeatureMember\[\d+\]/Building\[\d+\]). See Appendix A for a syntactically valid GeoXACML rule.

8.3.3.2 WMS

Defining rules referring to XML nodes in a WMS request is conceptually equal to the WFS case and examples are therefore omitted. An example of a rule referring to an **element** node in a WMS request is (+, staff, /GetMap)).

8.3.4 Content Dependant Rules

In order to simplify the examples the condition expression of a content dependant rule in pseudo syntax was simplified. In reality the XPath expression in conditions is composed of a "base path" (the XPath expression represented by the special resource-id XACML attribute) and a relative path from this context node. The resource-id attribute is used to get a unique context node (c.p. GeoXACML rule in correct syntax in Appendix A.1). Additionally it should be noted that the content dependant rules in this section refer to non spatial content (i.e. XML nodes that represent parameters that do not have geometric characteristics) and therefore these access rights can be declared and enforced with XACML and GeoXACML. See [2] section 10.2.8 for a comprehensive list of functions that are supported to define content dependant conditions in rules.

8.3.4.1 WFS

- **Pre-processing:**
  1. (+, staff, /GetFeature, /GetFeature/Query/@typeName = 'Building' or 'Street')
     "The staff user is allowed to request objects of FeatureType 'Building' or 'Street'".
  2. (-, staff, /GetFeature, /GetFeature/Query/@typeName = 'Building' or 'Street' and /GetFeature/Query/PropertyName = Geometry)
     "The staff user is not allowed to request the 'Geometry' attribute of objects of FeatureType 'Building' or 'Street'".

- **Post-processing:**
  3. (+, staff, /FeatureCollection/FeatureMember/Building, /FeatureCollection/FeatureMember/Building/Owner = 'Bob')
     "The staff user is allowed to request building objects whose owner is 'Bob'".
  4. (+, staff, /FeatureCollection/FeatureMember/Building, /FeatureCollection/FeatureMember/Building/Price <= '1000000')
     "The staff user is allowed to request building objects whose price attribute is larger than '1000000'".
  5. (-, staff, /FeatureCollection/FeatureMember/Building, /FeatureCollection/FeatureMember/Building/@id = '12345')
     "The staff user is not allowed to request the building object with id '12345'".

8.3.4.2 WMS

The GetMap request contains many parameters that have influence on how the map is rendered and about its format. As these parameters do not have geometric characteristics, access rights can be declared and enforced with XACML. How access rights on the spatial parameters in a WMS request can be defined is shown in section 3.5:

1. (+, staff, /GetMap, /GetMap/Output/Format = 'image/jpeg' or 'image/gif')
   "The staff user is allowed to see maps in the format "jpeg" or "gif".
2. (+, staff, /GetMap, /GetMap/Output/Style = 'black/white')
   "The staff user is allowed to render the map with style 'black/white'".
3. (+, staff, /GetMap, /GetMap/StyledLayerDescriptor/NamedLayer/Name = 'WATER' or 'STREET')
   "The staff user is allowed to see the layer 'WATER' or 'STREET'.
4. (+, staff, /GetMap, /GetMap/Size/Width <= '600' and /GetMap/Size/Height <= '480')
   "The staff user is allowed to request maps where the width is less than or equal to 600 pixels and the height is less than or equal to 480 pixels".
5. (+, staff, /GetMap, /GetMap/BoundingBox/@srsName = 'wgs84')
   "Users acting in the name of the role 'staff' can submit getMap requests with a BoundingBox Element whose coordinates refer to the 'wgs84' spatial reference system".

8.3.5 Spatial Access Control Rules

Note that all rules expressed in the following examples can only be expressed in GeoXACML – the spatial extension of XACML. It is not possible to express them in XACML directly as XACML doesn't allow the expression of spatial access control rules. It is important to highlight that the spatial predicates can refer to spatial attributes in the WS-request or response and to spatial data in the context. The latter case is combination of two rule types (context dependant rule type and spatial rule type). In order to have a simple classification we didn't introduce extra rule types for rules combining the basic rule types. At this point it is worth mentioning that the content dependant rule type can be seen as a superclass of the spatial rule type.

8.3.5.1 Spatial Access Control Rules Using Topologic Functions

8.3.5.1.1 WFS

Access can be restricted based on the geometry/area of requested features. The selection for which features a spatial access rule applies can be performed through topological predicates in a rule's condition part.

1. (+, staff, /FeatureCollection/FeatureMember/Building, /FeatureCollection/FeatureMember/Building/Geometry within 'US')'
   "The staff user is allowed to request building objects that are within the 'US'".
2. (+, staff, /FeatureCollection/FeatureMember/Building, /FeatureCollection/FeatureMember/Building/Geometry disjoint 'SecretArea')'
   "The staff user is allowed to request building objects whose Geometry is disjoint with a 'secretArea'".
3. (+, staff, /FeatureCollection/FeatureMember/Street, /FeatureCollection/FeatureMember/Street/Geometry within 'California' and crosses 'HW_No.5')
   "The staff user is allowed to request street objects that are within 'California' and crosses the 'highway number 5'".

From the examples given above it should be clear how spatial access rules containing topological predicates in their condition parts can be defined. GeoXACML supports the following topological predicates/functions in spatial rule conditions: equals, disjoint, touches, crosses, within, contains, overlaps, intersects. Note that geometries of type Point, LineString, Polygon and MultiPoint, MultiLineString, MultiPolygon are valid parameters of these topologic predicates.

8.3.5.1.2 WMS

Based on the area for which the user requests a map (the BBOX parameter of the GetMap request), access to a WMS can be protected. A GeoXACML Rule snippet for example 4) can be found in Appendix A.2.

1. (+, staff, /GetMap, /GetMap/BoundingBox within 'Area_A')
   "The staff user is allowed to request a map inside of a defined area A"
2. (-, staff, /GetMap, /GetMap/BoundingBox intersects 'Area_B')
   "The staff user is not allowed to request a map that intersects a defined area B."
3. (-, staff, /GetMap, /GetMap/BoundingBox intersects 'BorderLine_US_Canada')
4. (-, staff, /GetMap, /GetMap/BoundingBox crosses 'BorderLine_US_Canada')
   "The staff user is not allowed to request a map if the border line between the US and Canada crosses the area of interest (BBOX parameter of the request) (cross border map).

5. (-, staff, /GetMap, /GetMap/BoundingBox contains 'Any-of_the_hotSpots')
   "The staff user is not allowed to request a map if a "hot-spot" is within the area of interest (BBOX parameter).

8.3.5.2 Spatial Access Control Rules Using Scalar Geometric Functions

In GeoXACML you can use the scalar geometric functions area, length and distance in a rule's condition to express authorization semantics like in the examples below.

8.3.5.2.1 WFS

1. (+, staff, /FeatureCollection/FeatureMember/Building, area(/FeatureCollection/FeatureMember/Building/Geometry) <= 100)
   "The staff user is allowed to request building objects that have a base area smaller or equal than 100m²".
2. (+, staff, /FeatureCollection/FeatureMember/Street, length(/FeatureCollection/FeatureMember/Building/Street) <= 20000)
   "The staff user is allowed to request street objects that have a length smaller or equal than 20 km".

8.3.5.2.2 WMS

Access to a WMS can be controlled based on the resolution of the requested map. The resolution of the map can be calculated from the area of interest (parameters BBOX and SRS) as well as the WIDTH and HEIGHT parameters.

1. (+, staff, /GetMap, area(/GetMap/BoundingBox) <= 5000)
   "A staff user may only request maps where the area of interest is less than 5 km^2.
2. (+, staff, /GetMap, (/GetMap/StyledLayerDescriptor/NamedLayer/Name = 'WATER') AND (divide(mult(/GetMap/Size/Width, /GetMap, /GetMap/Size/Height), area(/GetMap/ BoundingBox)) <=1.5))
   "A staff user is allowed to request a map of layer 'WATER' if the resolution is less than 1.5 pixel per $m^{2"}$.
3. (+, staff, /GetMap, distance(/GetMap/BoundingBox, 'Any_hot-spot' <= 1000)
   "A staff user is allowed to request maps if the shortest distance from any "hot-spot" to the area of interest is greater than a 1km".

8.3.5.3 Spatial Access Control Rules Using Constructive Geometric Functions

In GeoXACML you can use constructive geometric functions like Buffer, Boundary, Union, Intersection, Difference, SymDifference, Centroid or ConvexHull to express authorization semantics like in the examples below.

8.3.5.3.1 WFS

1. (+, staff, /FeatureCollection/FeatureMember/Building, withinDistance((centroid(/FeatureCollection/FeatureMember/Building/Geometry), 'hot-spot', 100)
   "The staff user is allowed to request building objects, whose centroid is within 100 m of a hot-spot location".
2. (+, staff, /FeatureCollection/FeatureMember/Building, intersects(buffer(/FeatureCollection/FeatureMember/Building/Geometry, 100),'hot-spot')
   "The staff user is allowed to request building objects, whose 100 meter buffer intersects with a hot-spot location".

8.3.5.3.2 WMS

Whenever a policy defines an area for which a user is permitted to see a map, using a polygon, it is important to state permissive rights that allow requesting maps that overlap the permitted area "just a little".

1. (+, staff, /GetMap, divide(area(difference(/GetMap/BoundingBox, 'area_A')), area('area_A') <= 0.20) "The staff user is allowed to request maps if the area of interest is not more than 20% outside the permissive area".

8.3.5.4 Spatial Access Control Rules Using Miscellaneous Geometric Functions

The following pre-processing rule for transactional WFS operations summarizes the use of all miscellaneous geometric functions provided by GeoXACML.

1. (-, staff, /Transaction/Insert/Building,
   IsSimple(/Transaction/Insert/Building/Geometry) and
   IsClosed(/Transaction/Insert/Building/Geometry) and
   IsValid(/Transaction/Insert/Building/Geometry))
   "The staff user is allowed to insert Building features if the geometry of the feature is simple, closed and valid.

9 Future Work Items

9.1 Returning Access Control Process Information to the User and Binding Security Related Information to the Request

It is still an open question how security related information like a simple access denied message or the notification that the OWS response was filtered because of insufficient permission would be returned in a standardized way to the requestor. The conceptual problem is how to bind security responses form different security services (e.g. the access control system) to the actual OWS response. Should the information be included inside the OWS response itself? Should there be a SecurityServiceReport next to the actual OWS response? If so how to bind the two information entities together? Obviously further research is needed in this direction.

As the problem of binding security related information to an OWS response occurs not only in the access control domain but also in other security domains it seems most appropriate to work on a general solution that fits the requirements of all security related services. As all existing and future OWS will need to support minimal requirements in order to be able to provide a generic solution how to bind security information to the OWS response, it seems that the appropriate place for the results of research in this direction needs to be written down in the OWS Common Implementation Standard.

Note that the problem how to bind access control process results or security information in general to the actual OWS response is closely related to the problem of how to bind security information to an OWS **request**. A general solution for the problem of binding security process results to an OWS response should ideally also be applicable to bind security information to an OWS request.

It should also be mentioned that after a general transfer mechanism was worked out it will be helpful or even necessary to think of standardized security codes for the information that is returned by the security services. One could e.g. define URNs that express generic security service responses (e.g. urn:…:access-control-system:access-denied or urn:…:authentication-service:provide-username-password). Thanks to such unique and obviously to be standardized security related codes, client software could (maybe even automatically) react adequately to such a response.

In this line of work it should also be addressed that services should to be able to expose their security requirements to requesting clients as part of the service capabilities metadata. Solutions have to be worked out how this information can be included in some standardized way in the getCapabilities documents of the service by the PEP securing the service.

9.2 Interplay of the Access Control Service with other Security Services

Another interesting and important direction for future research is to make sure that a sound an interoperable interplay of the access control services with other security services like e.g. an authentication or encryption service is guaranteed. Developing a best practise paper describing when to use which services and how they can be combined will support an easy implementation of a comprehensive security architecture for an OWS based infrastructure.

9.3 XACML Obligations

XACML provides the mechanism of obligations on the policy element level (! note that it is not provided on the XACML rule level.). If one assumes that there is always only one rule in a policy (note that this destroys XACML separation of the policy and rule element) and one further misuses the obligation mechanism to provide the actual access control, then it is possible to perform additional access control during the pre-processing phase for content dependant access control semantics – i.e. access control based on the OWS request. Note that the use of obligations during post-processing is not necessary as everything can be done directly in GeoXACML (i.e. perform the access control within the access control system).

The idea behind obligations is that inside the access control system no real content dependant access control takes place. Instead, the filter part of a request is extended (more precisely combined with the “and” operator) by additional filters stored inside the obligation element.

Through such a conjunction of the original filter of the request with the “obligation filter” (expressing the authorisation semantics) the intended content dependant rules will hopefully (i.e. a correct processing in the OWS assumed) be enforced. Further research is necessary in order to find out if and under which circumstances the obligation mechanism makes sense for a certain OWS. Further a performance comparison where access control can be performed on the request through obligation based filter extension or alternatively on the response through native GeoXACML needs to be conducted. It will be interesting to see whether it is faster to process a very complex filter inside an OWS or to perform the access control on the response. Another important fact that needs to be clarified is if such a use of obligations conforms to the philosophy of OASIS XACML WG.

9.4 PAP Web Service

Probably the most important future work item that needs to be addresses next to all language relevant work items is to develop a powerful Policy Administration Point Web Service. Detailed research has already started in other projects in this direction and needs be introduced and tested in future OWS initiatives.

Given the number of features to be secured, the number of roles, the complexity of the policies and the number of distributed administrators, the need for a sophisticated Policy Administration Service hiding most of the complexity from the policy writer becomes

clear. This need becomes even more obvious if one takes the dynamic of the feature data base, the rule repository and the WS-users and administrators into account. Another important aspect of a PAP can be seen in a federation of protected services, when Policies from different security domains should be harmonized, combined or analyzed to ensure that users or services have all the access rights to execute a particular orchestration of services but not more.

9.5 Further Future Work Items

In the following we will simply list additional future work items that were also identified while working on this report. It is highly desirable to address the issues below as soon as possible as research in these fields will help clarify the overall understanding of the problem domain of access control for OGC web services and will be necessary to support an optimal, mature access control solution for the different OWS, authorization needs and application domains.

- How to provide access control for getCapabilities requests and for Catalogue Services. What is the relation between the access control rules for the Service and its metadata. Can the access rules on metadata be automatically inferred?

- What are the special requirements when providing access control for workflow services and how could they be met?

- Start work towards a new GeoXACML 1.1 version (possible topics):

    - support of 3D spatial rules and enablement of access control for 3D features

    - new classes of spatial relations (e.g.: an additional layer of fine-grained topological relations that focuses on the special characteristics of complex geometries)

    - relations based on orientation e.g. (permit, if building is north of Munich)

    - formal definition of GeoXACML's spatial relations on features in the "GML domain"

- Write executive summaries of GeoXACML and provide tutorials with example scenarios for different OWSs to make it as easy as possible for potential users to implement GeoXACML based access control systems for their OWS based spatial data infrastructure.

10 Conclusion

10.1 Summary

During OWS-6 we conducted a detailed analysis how XACML and GeoXACML and related standards and how they could be applied to secure OGC Web Services. It could be shown that the existing XACML standards can, despite of their general usefulness, be improved to better address the needs of access control in the OWS context. This engineering report contains all results that where identified during the detailed analysis performed during OWS-6. Additionally it explains in detail different solutions to improve the current situation. It further gives comprehensive recommendation how to use the access control techniques to control access for OWS. Next to these guidelines it is shown how to enhance interoperability when using the guidelines in this report. Based on the findings in this ER it is possible to derive further documents (e.g. a OWS profile of GeoXACML) that will provide the needed refinements of existing standards that address the special conditions of the OWS use case. Further a list of future work items was deduced that would provide a good orientation the next steps to be performed by access control related working groups within the OGC.

10.2 Next Steps

Next to the mentioned future work items in section 9 the following tasks have to be addressed urgently.

10.2.1 Definition of a OGC Web Service Profile of GeoXACML

Thanks to this report we have a detailed analysis and sound solution how to generate ACDR from OWS requests and responses and how to define the corresponding rules without reducing the expressive power of (Geo)XACML. Based on these findings we can and have to start working on an OGC Web Service profile of GeoXACML that will standardize the guidelines how to use GeoXACML to protect OGC Web Services and thereby providing the needed enhanced interoperability. Note that the benefit of this profile will not only be the enhancement of interoperability in GeoXACML based access control systems for OWS. It also supports an easier applicability and implementation of XACML or GeoXACML in OWS environments, as the guidelines in the profile that precisely describe how to use the language, would be less generic.

The following topics are suggested for an “OGC Web Service Profile of GeoXACML”

- guidelines for interoperable access control decision requests in the OWS context
    - allow OWS request and response information in <RescourceContent> elements only

  - in case of KVP encoded requests transform the request into its XML encoded representation

  - specify unique rules for resource-id values

- guidelines for interoperable access control rules for OWS

  - definition of an XMLAttributeSelector (in cooperation with XACML WG)

  - unique indicators for aggregated rule semantics
    e.g.: (+, WFS) should be identified by the existence of a getFeature or FeatureCollection XML element in <RescourceContent>

- guidelines how to use XACML's obligation mechanism in an OASIS conformant way in the OWS use case

  - define precisely how pre-processing rules have to look like

- harmonize a obligation based approach with the XACML and GeoXACML standard

In order to address all these topics and achieve the standardization of the urgently needed OWS profile of GeoXACML we need an appropriate organizational frame. Currently OGC members are forming a persistent GeoXACML SWG that will have all work items mentioned in section 10.2.1 and 10.2.2 on its agenda.

10.2.2 Cooperation and Coordination with OASIS' XACML WG

As explained in section 7 it is necessary to improve OASIS Multiple and Hierarchical resource profile of XACML and the XACML specification itself, in order to handle the complexity of the access control for OWS use case adequately. Therefore, the results of this report have to be introduced to the OASIS' XACML WG. In general it is necessary and helpful to coordinate OGC's work on GeoXACML with the work of the OASIS XACML WG in order to harmonize the closely related work of both standardization bodies. This coordination work should be part of the GeoXACML SWG.

10.2.3 Cooperation with other OGC Working Groups

All work items introduced in section 7.3 should be addressed in cross Working Group sessions and teleconferences. Members of the Security DWG, OWS Common DWG and GeoXACML SWG should cooperate and coordinate their work in order to generate a general-use and harmonized solution. Some topics that need to be addressed in such a cooperating group are:

- develop unique guidelines

  - how to bind return values of the access control process and other Security Services with OWS responses and
  - how to bind security information to OWS requests

- define standardized exception codes
- ensure through guidelines that every OGC Web Service standard that allows also KVP encoded requests also has a mandatory, standardized request XML Schema. In this case the definition of normative bijective transformation rules between KVP and XML encoded requests is a mandatory part of the corresponding specification.
- interplay of GeoXACML with other services of OWS' security architecture
- specification of minimal requirements for OWS standards in order to support the sound and strait forward applicability of generic security solutions for OWS

Appendix

Appendix A

A.1: A GeoXACML Rule Example for WFS

The following GeoXACML rule is the implementation of a rule similar to example in 8.3.5.1.1, rule number 1: (+, anySubject, /FeatureCollection/Feature-Member/Building, /FeatureCollection/FeatureMember/Building/Geometry within 'US')' "Any user is allowed to request building objects that are within the 'US'".

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Rule RuleId="spatialBuildingRule WITHIN Polygon Polygon"
Effect="Permit">
  <Target>
    <Subjects><AnySubject/></Subjects>
    <Resources>
      <Resource>
        <ResourceMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:regexp-string-match">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">/Request[1]/Resour
ce[1]/ResourceContent[1]/wfs:FeatureCollection\[\d+\]/gml:featureMemb
er\[\d+\]$</AttributeValue>
          <ResourceAttributeDesignator
DataType="http://www.w3.org/2001/XMLSchema#string"
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions><AnyAction/></Actions>
  </Target>
  <Condition FunctionId="urn:ogc:def:function:geoxacml:1.0:geometry-
within">
    <Apply FunctionId=" urn:ogc:def:function:geoxacml:1.0:geometry-
one-and-only">
      <Apply FunctionId=http://www.geoxacml.com#XMLSelector
DataType="urn:ogc:def:dataType:geoxacml:1.0:geometry">
<!-- the XMLSelector concatenates its two to string arguments and returns a
bag that is generated from the selected xml node-set that is casted to the
specified datatype before  - it's a workaround as xacml 2.0 does not yet
support the xml datatype -->
        <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-
only">
          <ResourceAttributeDesignator
DataType="http://www.w3.org/2001/XMLSchema#string"
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
        </Apply>
        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">/myns:Building/myn
s:Geometry/gml:Polygon</AttributeValue>
      </Apply>
    </Apply>
    <AttributeValue
DataType="urn:ogc:def:dataType:geoxacml:1.0:geometry">
```



```
      <gml:Polygon
srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
        <gml:outerBoundaryIs>
          <gml:LinearRing>
            <gml:coordinates decimal="." cs="," ts=" ">20,380 40,200
240,140 360,260 340,380 200,400 100,400 ...the area representing the
US...  20,380</gml:coordinates>
          </gml:LinearRing>
        </gml:outerBoundaryIs>
      </gml:Polygon>
    </AttributeValue>
  </Condition>
</Rule>
```

Note that this rule uses the extended AttributeSelector functionality as described in 7.2.1.1 item 8.

A.2: A GeoXACML Rule Example for WMS

The following rule is the GeoXACML conformant implementation of a rule similar to example 8.3.5.1.2, rule number 4: (-, staff, /GetMap, /GetMap/BoundingBox crosses 'BorderLine_US_Canada').

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Rule RuleId="spatialWMS-pre-
Processing Rule crosses Polygon LineString" Effect="Permit">
  <Target>
    <Subjects><AnySubject/></Subjects>
    <Resources>
      <Resource>
        <ResourceMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:regexp-string-match">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">/Request[1]/Resour
ce[1]/ResourceContent[1]/ogc:GetMap\[\d+\]$</AttributeValue>
          <ResourceAttributeDesignator
DataType="http://www.w3.org/2001/XMLSchema#string"
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions><AnyAction/></Actions>
  </Target>
  <Condition FunctionId="urn:ogc:def:function:geoxacml:1.0:geometry-
crosses">
    <Apply FunctionId=" urn:ogc:def:function:geoxacml:1.0:geometry-
one-and-only">
      <Apply FunctionId=http://www.geoxacml.com#XMLSelector
DataType="urn:ogc:def:dataType:geoxacml:1.0:geometry">
<!-- the XMLSelector concatenates its two to string arguments and returns a
bag that is generated from the selected XML node-set that is casted to the
specified datatype before  - it's a workaround as xacml 2.0 does not yet
support the XML datatype -->
        <Apply
FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-
only">
          <ResourceAttributeDesignator
DataType="http://www.w3.org/2001/XMLSchema#string"
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"/>
        </Apply>
```

```
        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">/
ogc:BoundingBox</AttributeValue>
      </Apply>
    </Apply>
    <AttributeValue
DataType="urn:ogc:def:dataType:geoxacml:1.0:geometry">
      <gml:Linestring xmlns:gml="http://www.opengis.net/gml"
gid="nUSBorder" srsName="EPSG:4326">
        <gml:coordinates cs=" " ts=", ">7.15782 53.60969, 7.06519
52.32382, 5.96825 51.80972, 5.73957 50.88717, 6.21186
50.07928</gml:coordinates>
      </gml:Linestring>
    </AttributeValue>
  </Condition>
</Rule>
```

Note that this rule uses the extended AttributeSelector functionality as described in 7.2.1.1 item 8.

A.3 A GeoXACML and Multiple and Hierarchical Resource Profile Conformant Global Access Control Decision Request

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Request>
  <Subject>
    <Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
DataType="rfc822Name">
      <AttributeValue>bob@acompany.com</AttributeValue>
    </Attribute>
    <Attribute
AttributeId="urn:oasis:names:tc:xacml:2.0:subject:role"
DataType="string">
      <AttributeValue>junior-staff</AttributeValue>
    </Attribute>
  </Subject>
  <Resource>
    <ResourceContent>
<!-a ows request or response (e.g. wfs-response for post-processing
ac)->
      <wfs:FeatureCollection ...>
        <gml:boundedBy>
          <gml:Box
srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
            <gml:coordinates>-180.0,-90.0
180.0,90.0</gml:coordinates>
          </gml:Box>
        </gml:boundedBy>
        <gml:featureMember>
          <BUILDING>
            <Geometry>
              <gml:Polygon
srsName="http://www.opengis.net/../epsg.xml#43">
                <gml:outerBoundaryIs>
                  <gml:LinearRing>
                    <gml:coordinates decimal="." cs="," ts=" ">
                      -120.000000,65.588264 ...-120.000000,65.588264
                    </gml:coordinates>
```



```
                  </gml:LinearRing>
                </gml:outerBoundaryIs>
              </gml:Polygon>
            </Geometry>
            <Name>White House</Name>
            <Buildyear>1792 </Buildyear>
            <Owner>state</Owner>
            <NoStoreys>3</NoStoreys>
          </BUILDING>
        </gml:featureMember>
        <gml:featureMember>
          ...
        </gml:featureMember>
        ...
      </wfs:FeatureCollection>
    </ResourceContent>
    <Attribute AttributeId="resource-id"
DataType="string"><AttributeValue>/Request[1]/Resource[1]/ResourceCon
tent[1]/
wfs:FeatureCollection[1]/gml:featureMember[1]</AttributeValue>
    </Attribute>
    <Attribute AttributeId="scope" DataType="string">
      <AttributeValue>Descendants</AttributeValue>
    </Attribute>
  </Resource>
  <Action />
</Request>
```