Open Geospatial Consortium

Approval Date: 2013-01-18

Posted Date: 2013-02-05

Reference number of this document: OGC 12-139

Reference URN for this document: http://www.opengis.net/def/doc-type/per/ows9-ssi-security-rules

Category: OGC Public Engineering Report

Editors: Jan Herrmann
Andreas Matheus

OGC® OWS-9 Security and Services Interoperability Thread:
-
SSI Security Rules Service 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

i. Abstract

In this engineering report we describe how to administrate XACML v2.0, XACML v3.0 and GeoXACML v1.0.1 access control policies through a “Security Rules Service”. Following the XACML and ISO terminology this service plays the role of a Policy Administration Point (PAP) and is therefore called XACML Policy Administration Point (XACML PAP) or XACML Policy Administration Web Service (XACML PAWS).

After introducing OWS-9’s Common Rule Encoding and motivating all components required to administrate (Geo)XACML policies, we describe the interface of a powerful XACML PAP on a conceptual level. This interface definition could serve as a baseline for a future OASIS or OGC XACML Policy Administration Web Service (e.g. OGC XACML PAWS) specification.

Keywords

ogcdoc, ows9, ssi, security, xacml, geoxacml, pap

What is OGC Web Services 9 (OWS-9)?

OWS-9 builds on the outcomes of prior OGC interoperability initiatives and is organized around the following threads:

- **Aviation**: Develop and demonstrate the use of the Aeronautical Information Exchange Model (AIXM) and the Weather Exchange Model (WXXM) in an OGC Web Services environment, focusing on support for several Single European Sky ATM Research (SESAR) project requirements as well as FAA (US Federal Aviation Administration) Aeronautical Information Management (AIM) and Aircraft Access to SWIM (System Wide Information Management) (AAtS) requirements.

- **Cross-Community Interoperability (CCI)**: Build on the CCI work accomplished in OWS–8 by increasing interoperability within communities sharing geospatial data, focusing on semantic mediation, query results delivery, data provenance and quality and Single Point of Entry Global Gazetteer.

- **Security and Services Interoperability (SSI)**: Investigate 5 main activities: Security Management, OGC Geography Markup Language (GML) Encoding Standard Application Schema UGAS (UML to GML Application Schema) Updates, Web Services Façade, Reference Architecture Profiling, and Bulk Data Transfer.

- **OWS Innovations**: Explore topics that represent either new areas of work for the Consortium (such as GPS and Mobile Applications), a desire for new approaches to existing technologies to solve new challenges (such as the OGC Web Coverage Service (WCS) work), or some combination of the two.

- **Compliance & Interoperability Testing & Evaluation (CITE)**: Develop a suite of compliance test scripts for testing and validation of products with interfaces

implementing the following OGC standards: Web Map Service (WMS) 1.3 Interface Standard, Web Feature Service (WFS) 2.0 Interface Standard, Geography Markup Language (GML) 3.2.1 Encoding Standard, OWS Context 1.0 (candidate encoding standard), Sensor Web Enablement (SWE) standards, Web Coverage Service for Earth Observation (WCS-EO) 1.0 Interface Standard, and TEAM (Test, Evaluation, And Measurement) Engine Capabilities.

The OWS-9 sponsors are: AGC (Army Geospatial Center, US Army Corps of Engineers), CREAF-GeoViQua-EC, EUROCONTROL, FAA (US Federal Aviation Administration), GeoConnections - Natural Resources Canada, Lockheed Martin Corporation, NASA (US National Aeronautics and Space Administration), NGA (US National Geospatial-Intelligence Agency), USGS (US Geological Survey), UK DSTL (UK MoD Defence Science and Technology Laboratory).

License Agreement

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

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

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

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

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

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Table of contents

Figures

Page

ii. Preface

This engineering report was prepared as a deliverable for the OGC Web Services, Phase 9 (OWS-9) initiative of the OGC Interoperability Program. This document presents the results of the work within the OWS-9 Security and Services Interoperability (SSI) thread. It describes how to administrate XACML v2.0, XACML v3.0 and GeoXACML v1.0.1 access control policies.

iii. 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.

iv. Submission and Contribution Contact Points

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

Name	Organization
Jan Herrmann	Secure Dimensions GmbH
Andreas Matheus	Secure Dimensions GmbH

v. Revision History

Date	Release	Editor	Primary clauses modified	Description
2012/07/16	0.1	JH	All	draft of intended structure
2012/07/16	0.2	JH	All	first working draft
2012/11/18	0.3	JH/AM	All	updates in all sections
2012/12/01	0.4	JH	All	updates in section 6.1 and 7.5 according to received comments
2012/12/07	0.5	JH	All	completion of section 8

vi. Changes to the OGC Abstract Specification

The OGC® Abstract Specification does not require changes to accommodate the technical contents of this document.

vii. 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.

Note: Readers not familiar with the XACML 2.0 and GeoXACML 1.0.1 specification and the use of these standards in the context of OGC Web Services are referred to [6], [5], [2] and [3] before reading this report.

OWS-9 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.

The SSI thread addresses, next to others, the question of how to securely and federatively administrate XACML v2.0, XACML v3.0 and GeoXACML v1.0.1 encoded access control policies (cp. [12], Annex B, section 6).

OWS-9 Security and Services Interoperability Thread - The SSI Security Rules Service Engineering Report

1 Overview

Thanks to specifications like XACML v2.0, XACML v3.0 and GeoXACML v1.0.1 it is fairly straight forward to implement powerful access control systems that protect Geo Web Services and spatial data in spatial data infrastructures (SDIs). The underlying hybrid right model of these systems combines rule-, rewrite- and role-based rights models and guarantees that expressive fine grained access rights can be defined and enforced.

The new challenge that arises when using XACML v2.0 and GeoXACML v1.0.1 access control systems is to provide suitable administration systems and models that support the sound administration of the emerging complex access control policies.

In this engineering report we describe how to administrate XACML v2.0, XACML v3.0 and GeoXACML v1.0.1 access control policies through a "Security Rules Service". Following the XACML and ISO terminology this service plays the role of a Policy Administration Point (PAP) and is therefore called XACML Policy Administration Point (XACML PAP) or XACML Policy Administration Web Service (XACML PAWS).

After introducing OWS-9's Common Rule Encoding and motivating all components required to administrate (Geo)XACML[1] policies, we describe the interface of a powerful XACML PAP on a conceptual level. This interface definition could serve as a baseline for a future OASIS or OGC XACML Policy Administration Web Service (e.g. OGC XACML PAWS) specification.

In the definition of the XACML PAWS interface we describe an expressive Layered Administration Model that enables distributed and tractable administration of complex (spatial) access control policies as found in SDIs.

During the implementation chapter 7 we highlight the important design and realization aspects of the OWS-9 PAP implementation and demonstrate its use.

[1] The abbreviation "(Geo)XACML" shall be interpreted as "XACML or GeoXACML".

Closing section 8 concludes this report by mentioning some important work items that need to be addressed in the future.

2 Bibliography

[1] 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

[2] 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.

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

[4] J. Herrmann. Access Control in Service-oriented Architectures applied to spatial data infrastructures. PhD thesis, Technische Universität München, Germany, October 2011.

[5] J. Herrmann and A. Matheus. OWS-6 GeoXACML engineering report. OGC public engineering report, Open Geospatial Consortium (OGC), July 2009.

[6] J. Herrmann and A. Matheus. OWS-8 Authoritative AIXM Data Source Engineering Report. OGC public engineering report, Open Geospatial Consortium (OGC), September 2011.

[7] 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.

[8] International Committee for Information Technology Standards. Information technology - role based access control (rbac). Ansi/incits standard, InterNational Committee for Information Technology Standards (INCITS), 2004.

[9] ISO10181-3 ISO/IEC 10181-3:1996 Information technology – Open Systems Interconnection -- Security frameworks for open systems: Access control framework.

[10] 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.

[11] 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=18241

[12] Request for Quotation (RFQ) And Call for Participation (CFP) OGC Web Services Initiative - Phase 9 (OWS-9) - Annex B OWS-9 Architecture, OGC. February 2012. https://portal.opengeospatial.org/files/?artifact_id=47797.

[13] RFC3198 IETF RFC 3198: Terminology for Policy-Based Management, November 2001. http://www.ietf.org/rfc/rfc3198.txt

3 Terms and Definitions

For the purposes of this document, the following terms and definitions apply. Please note that some terms and definitions are taken from the XACML specification [2] and the XACML v2.0 Multiple Decision Profile v1.0 [10] and are included here for easy reading.

Access control - Controlling access in accordance with a policy

Action - An operation on a resource

(XACML) Attribute - Characteristic of an entity that may be referenced in a predicate or target. A specific instance of an attribute, determined by the attribute name and type, the identity of the attribute holder and (optionally) the identity of the issuing authority

(XACML) 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

(XACML) Authorization Decision Request (ADR) - The request by a PEP or Context Handler to a PDP to render an authorization decision

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”

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

(XACML) evaluation context - The canonical representation of a decision request and an authorization decision

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

Global Authorization Decision Request (global A.D.R.) – an access control decision request referring to one or multiple resources

Global Authorization Decision Response – an aggregation of individual access control decision responses

Individual Authorization Decision Request (individual A.D.R.) – a decision request referring to exactly one resource node

Individual Authorization Decision Response – a decision response referring to exactly one resource node

Obligation - An operation specified in a rule, policy or policySet element that should be performed by the Obligation Handler 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 [13]. This term corresponds to "Access Decision Function" (ADF) in [9].

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 [13]. This term corresponds to "Access Enforcement Function" (AEF) in [9].

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, a condition and obligations. 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 that a rule, policy or policy set is intended to evaluate.

4 Conventions

4.1 Abbreviated Terms

AD	Authorization decision
ADR	Authorization decision request
GeoPDP	PDP implementing GeoXACML
GeoXACML	Geospatial eXtensible Access Control Markup Language
GML	Geography Markup Language
HRP	Hierarchical Resource Profile
MRP	Multiple Resource Profile
OASIS	Organization for the Advancement of Structured Information Standards
OGC	Open Geospatial Consortium
OWS	OGC Web Service
OWS-6/7/8/9	OGC Web Services Initiative, Phase 6/7/8/9
PAP	Policy Administration Point
PDP	Policy Decision Point implementing XACML
PEP	Policy Enforcement Point
SDI	Spatial Data Infrastructure
SOA	Service Oriented Architecture
URL	Uniform Resource Locator

URN	Uniform Resource Names
WFS(-T)	Web Feature Service (-Transactional)
XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language

5 Introduction

Geo Web Services like OGC WFS, WMS or CWS instances and spatial objects (called features) are the resources of SDIs. One central IT-security requirement in these infrastructures is the protection of their resources from unauthorized use. To achieve this goal appropriate access control systems need to be developed and deployed.

Amongst the various existing right models that could be used for these access control systems, appropriate combinations of rule-[2], role- and rewrite-based access rights models have proven to be the right choice in many use cases, as these models support (next to others) the definition of very expressive rights at customizable granularity levels (see [6] for details).

The use of rule-based access control systems however introduces the challenge how to achieve a sound administration of the emerging complex access control policies. The success of these access control systems can only be guaranteed, if the administrators of the policies have confidence or proof, that the formal definitions of the policies reflect the intended authorization semantics.

In large service-oriented architectures like SDIs it can be a very challenging task to achieve this goal. First the administrators of different administrative domains have to address the problem of cooperatively defining huge access control policies that express vast amounts of complex, fine-grained rights for thousands of users/roles and millions of resources. Additionally diverse dynamics (e.g. the creation and deletion of roles, changes to their assigned privileges and the insertion, deletion and modification of the resources that need to be protected) imply complex administrative operations and therefore significantly complicate the administration of these policies.

Given this situation, it is apparent that there is an urgent need to develop concepts and tools that support a sound administration of policies for (Geo)XACML based access control systems in SDIs.

[2] aka. attribute-based or expression-based rights models.

6 Administration of (Geo)XACML based Access Control Systems

6.1 OWS-9 Common Rules Encoding

Before one can develop an appropriate solution for the administration of access rights one has to choose a suitable rights model for the access control system. Previous OWS initiatives and many other projects have shown that a hybrid access rights model, combing rule-, role- and rewrite-based access rights models is the right choice when building access control systems for OGC Web Service based SDIs. The OASIS eXtensible Access Control Markup Language (XACML) v3.0 standard is the most mature standard describing the implementation of such a hybrid access rights model. Due to its maturity, expressiveness and popularity XACML v3.0 has been selected as the core part of OWS-9's Common Rules Encoding.

As the geospatial application domain allows and requires the definition of expressive spatial rights, there is the need to extend the XACML standard (using its standardized extension points) by spatial capabilities. This issue has been addressed in the past within the OGC and a spatial extension of XACML v2.0 – called GeoXACML v1.0.1 – has been defined and standardized. GeoXACML v1.0.1 supports spatial data types and a rich set of spatial functions that can be used to define very complex spatial authorization semantics. Note that the GeoXACML standard v1.0.1 is an extension of the XACML v2.0 standard. However the dependencies of the concepts behind the GeoXACML v1.0.1 specification on XACML v2.0 are very limited. In fact only the XACML introduction and the XACML policy examples used in the GeoXACML v1.0.1 specification are XACML v2.0 specific.

What is currently missing is an GeoXACML v3.0 specification that defines a spatial extension of the XACML v3.0 specification. To address this gap the GeoXACML SWG has agreed to define a new GeoXACML v3.0 specification as soon as OASIS has officially declared XACML v3.0 as approved OASIS standard. Due to the loose coupling between GeoXACML v1.0.1 and XACML v2.0, the content of the GeoXACML v3.0 specification will be very similar to the one of GeoXACML v1.0.1. In detail the geometry data type and the spatial functions of GeoXACML v1.0.1 continue to be the innovative part of XACML's spatial extension. The changes between GeoXACML v1.0.1 and v3.0 will focus on the XACML introduction section and the policy examples that need to be updated from XACML v2.0 to XACML v3.0 syntax. During the writing of the GeoXACML v3.0 specification it must further be decided whether the currently evolving new version of the simple feature specification will be of relevance for the new GeoXACML v3.0 specification.

To conclude: The targeted Common Rules Encoding within OWS-9 is XACML v3.0 and its still-to-be-defined spatial extension GeoXACML v3.0.

6.2 Components needed to administrate (Geo)XACML policies

The successful usage of (Geo)XACML based access control systems requires that all relevant business, legal and normative regulations are implemented correctly in the underlying (Geo)XACML encoded access control policy. It is the duty of the

administrators in charge to ensure that the access control policy will exactly permit all authorized information flows (and only those) between the entities of the system at any point in time.

The achievement of this objective is a very challenging goal, especially in large scale service-oriented architectures like SDIs, where vast and complex access control rules need to be implemented. It is e.g. usually required to define fine-grained, spatial and context dependent rights referring to subjects, features and attributes with specific properties. Next to the complexity and extend of the rights that need to be implemented, diverse dynamics make the administrative tasks even more challenging. Feature instances, data models, users, roles and the required authorization semantics can and do change over time which implies complex operations on the (Geo)XACML access control policies.

Within the following three subsections it is analyzed which components are required to meet the administrative challenges in (Geo)XACML based access control systems. First Policy Administration Points are required, through which the policies can be securely administrated and analyzed (cp. section 6.2.1). Second one needs further access control systems protecting the Policy Administration Points (cp. section 6.2.2). Third an administration model is desirable that introduces a clear, well-defined structure of the resulting sets of administrative services, users/administrators, roles, policies etc. (cp. section 6.2.3).

6.2.1 Policy Administration Points

To support the decentralized administration of policies of (Geo)XACML based access control systems one or multiple Security Rules Services – e.g. implemented as Web Services – need to be provided. Following the XACML and ISO terminology these services play the role of Policy Administration Points (PAP) and are therefore called XACML Policy Administration Points (XACML PAPs) or Policy Administration Web Services (XACML PAWS).

To cover the common requirements of most use cases XACML PAPs need to provide the following classes of functionalities:

Core functionalities At very least XACML PAPs need to support functionalities that allow policy administrators to retrieve, insert, update and delete parts of XACML encoded policies. It should be noted, that the update operation could theoretically be replaced by sequences of corresponding read, delete and insert actions and hence does not necessarily belong to the set of minimal operations. In the following we however regard the update functionality as part of the core functionalities as this operation type significantly reduces the complexity of administrative actions.

Analyze functionalities Next to the core functionalities it is important that XACML PAPs provide operations that support the analysis of (Geo)XACML encoded policies. An XACML PAP should e.g. support analyze functionalities that allow to formally verify if certain access control rules are satisfy-able, in conflict or redundant. Thanks to those analyze functionalities policy administrators can verify semantic properties of parts of the to-be-rolled-out access control policy. Through these analyze functionalities administrators can win confidence that the access control system will behave as expected and will only permit authorized information flows at any point in time.

Optimization and transform functionalities Based on the analyze functionalities various optimization and transform functionalities can and should be realized. Functionalities of this class can e.g. support an automatic simplification and restructuring of XACML policies or an transformation from XACML v2.0 to XACML v3.0 encoding. Simplified policies in turn allow for an easier administration and imply significant performance advantages when evaluating or analyzing the policy.

Testing functionalities Next to the support of analyzes and optimization functionalities testing functionalities should be supported by powerful XACML PAPs. It is e.g. very convenient for policy administrators to have tools available that assist in the manual creation of test XACML authorization decision requests (XACML ADR) and that additionally provide insights in the details of the results when evaluating the test XACML ADRs (e.g. applicable policy elements and the boolean values their <Target> and <Condition> elements evaluated to).Thanks to the availability of detailed test results administrators can identify further semantic mistakes.

More details on the functionalities of the different classes (e.g. signatures, semantics and design principles) will be provided in section 6.3.

Every PAP service will have an associated PAP client providing a convenient graphical user interface (GUI) for the administrators. This GUI is needed to provide improved editing capabilities, to offer a clear view on already defined rules and most importantly to abstract from syntactical details of the access control policy language.

It is important to highlight, that a PAP providing the mentioned functional and graphical capabilities will not solve all difficulties that arise when administrating access control policies. As pointed out in the introduction, defining and maintaining access control rules is a highly complex task. Essential causes for the complexity of the administration of policies come directly from the application domain and are therefore not avoidable (e.g. quantity and complexity of rights, multiple administrators for the policy producing unintended side effects etc.). Hence one major problem area that is not mitigated by the PAP is, which rules need to be defined by whom, how should they be updated etc. In a first step all related parties have to specify informal or judicial descriptions of the needed authorization semantics. Afterwards one or more persons have to implement the authorization semantics according to the given descriptions. Further the administrators have to address the various changes that can occur over time in a coordinated manner. All these steps become particularly demanding when complex policies need to be enforced in large, dynamic and distributed environments. Neither the basic operations, nor the analysis and test functions, nor the GUI will address all involved challenges. In the next section a second central component of an access control strategy is introduced, that is

urgently needed to solve the mentioned issues and thereby achieve a sound administration of access control policies.

6.2.2 Access Control Systems for Policy Administration Points

Another set of components that is needed to successfully administrate access control policies in large, distributed IT-infrastructures are additional access control systems for the PAP components itself. These access control systems allow specifying precisely which administrator is allowed to perform which operations on a specific policy repository. The paragraphs below list some benefits that result from providing access control systems for PAPs:

Distribution of administrative rights The policy of an access control system for a PAP associates suitable subsets of administrative rights to different administrators. Through this divide and conquer strategy one can avoid excessive demands of the individual administrators and one can further realize various administrative security requirements (e.g. avoidance of excessive accumulation of administrative rights).

Controlling the mutual influence of cooperating administrators The presence of more than one administrator usually aims at disjunctively distributing the administrative tasks among them. Nevertheless there are situations where this distribution can and shall not be disjoint. For example overlapping fields of responsibility are needed, if two or more administrators from different domains have to cooperate to come up with an overall access control policy. In those cooperation scenarios it is essential that all involved administrators know the possible influences of other parties on their activities. By defining an access control policy that controls the mutual influence of the cooperating administrators when using a PAP, one can ensure that both parties always comply with the cooperation agreements.

Guaranteeing interoperability of policies The XACML specification defines a general purpose policy language and a corresponding authorization decision request and response language. Both languages can be used in many different ways. This flexibility introduces the risk of loosing interoperability between distributed but related access control systems. Thanks to the presence of access control systems for PAPs, interoperability enhancing agreements can be enforced (e.g. the definition of supported AttributeIds and valid sets of values for these XACML attributes). All policies generated under the policies of the access control systems for the PAPs will be interoperable as they conform to the guidelines establishing the interoperability.

Guaranteeing efficient policy evaluation and analysis Another motivation for access control systems for PAPs is the need to define and enforce certain constraints that rules in

a policy must comply with. These guidelines intend to ensure a certain performance level when enforcing and analyzing the policy.

The arguments presented in the last four paragraphs have shown that a flexible and expressive access control system for the PAPs themselves entails a lot of benefits for the administration of access control policies. We therefore conclude that next to the PAPs, access control systems for these PAPs are required in large scale access control systems.

6.2.3 Administration Model

The last section pointed out the need for access control systems and policies controlling the possible interactions with PAP components. Depending on the requirements of the use case there can be the need to control the administrative tasks on this second type of policies, which would imply a third type of policies. This layering of access control policies can be continued as far up as needed (cp. figure).

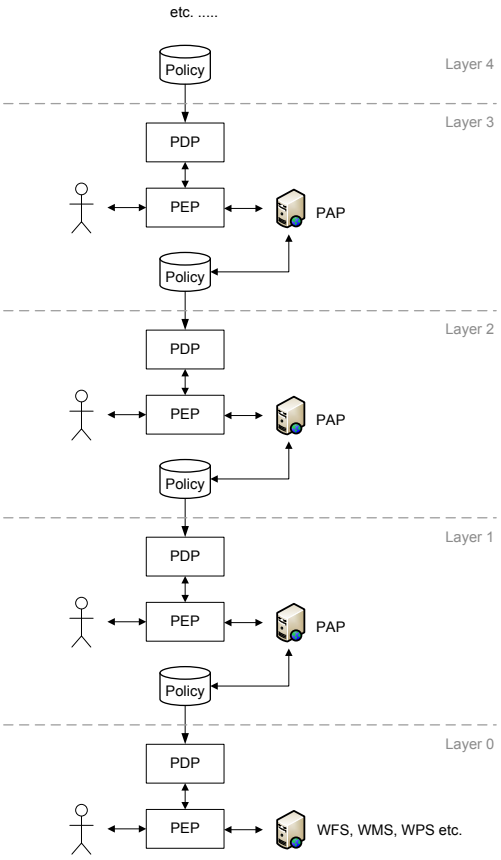

Figure 1 — Layers of interdependent access control policies

To allow and simplify the definition of multiple layers of related policies and to ease the explanation of the effects of access rights referring to administrative actions on policy elements, a well defined administration model is needed.

An administration model defines next to a terminology and its components a set of principles. In an administration system that implements a specific administration model,

all the principles of the corresponding model hold and it therefore inherits the capabilities and properties of the underlying model. It must be highlighted that the term administration service is not a synonym for the term administration system. The later can consist of any number of administration services and can further consist of any number of access control system components and policies.

Section 6.4 introduces the definition of the Layered Administration Model (LAM). The LAM describes how to build an administration system in which one can define and administrate rights, which control the possible interactions on other policies. The resulting policies on the different layers indirectly or directly determine the possible information flows between subjects and services in SDIs.

6.3 Policy Administration Point Interface

At the beginning of the design phase of the XACML PAP service interface one has to choose the right granularity level for the supported operations.

One strategy could be to allow for the most fine-grained operations on a XACML policy tree. The exact operators on the XACML tree structure will depend on the underlying representation. Using an XML DOM engine one could e.g. allow for operations on XML node level. This gives most flexibility but makes the definition of rich administrative rights unmanageable.

Another approach is to define more high-level operations on XACML policy trees that take its logical structure and associated semantics into account. The idea is to stay as detailed as possible but still enough coarse-grained to permit easy administration of administrative rights. Following this approach could e.g. result in the XACML PAP Web Service interface as introduced in the next sections.

6.3.1 CreateFileContainer

createFileContainer(String Store, String containerName)

Policy administrators can create new policy containers of type “file” in a container store by calling the createFileContainer operation. The desired store has to be specified through the “Store” parameter that equals a file-system path to a specific folder (e.g. C:\MyContainerStore).

6.3.2 CreateTableContainer

createTableContainer(String Store, String ContainerName, String createTableDefinition)

Policy administrators can create new policy containers of type “table” in a container store by calling the createTableContainer operation. The desired store has to be specified through the “Store” parameter that equals the URN of a DB server (e.g. www.a-db-server.com:1234/a-db). Through the createTableDefinition parameter the administrator can freely choose one of the supported XML-to-RDBM storage models of the underlying DBMS.

6.3.3 InsertPolicyElement

insertPolicyElement(String containerName, XPath pathToFather, String namespace, XML xacmlPolicyElement)

The insertPolicyElement operation allows inserting a new XACML element into a container. The xacmlPolicyElement parameter defines the policy element to be inserted. This element will get appended as new last child of the nodes, the XPath expression specified in the pathToFather parameter points to. Through a single insertPolicyElement request one can exactly insert one of the following XACML elements at one or multiple locations in the existing policy tree:

- <PolicySet>
- <Policy>
- <PolicySetIdReference>
- <PolicyIdReference>
- <Rule>

Other types of XACML elements can only be added to the XACML policy as children of elements of the types listed above. It needs to be highlighted that with the insertPolicyElement Operation one should only add <PolicySet> and <Policy> elements that represent simple but complete access rights/constraints. It is however recommended to deny through an administrative access control system that <PolicySet> and <Policy> elements can be inserted that define a tree of access rights/constraints, and therefore have children of type <PolicySet> or <Policy>. This will imply that XACML policy trees have to be build step by step though separate insertPolicyElement requests. The enforcement of successive policy generation only is an essential property to be able to support the definition of rich administrative rights.

6.3.4 InsertPolicyElementBefore and InsertPolicyElementAfter

The insertPolicyElementBefore and insertPolicyElementAfter operations are very similar to the insertPolicyElement operation

- **insertPolicyElementBefore(String containerName, XPath pathToSibling, String namespace, XML xacmlPolicyElement)**

- **insertPolicyElementAfter(String containerName, XPath pathToSibling, String namespace, XML xacmlPolicyElement)**

These two operations allow insert new policy elements at specific locations relative to their siblings. The establishment of a certain order between siblings is required as the effects of certain XACML conflict resolution algorithms (e.g. first-applicable) depend on the ordering of elements within the policy tree. Instead of a pathToFather parameter (cp. section 6.3.3) the signatures of both operations contain a pathToSibling parameter that points to the sibling(s), before or after the new policy element shall be inserted.

6.3.5 UpdatePolicyElement

updatePolicyElement(String containerName, XPath pathToNode, String namespace, XML xacmlPolicyElement, Boolean deep)

Through an updatePolicyElement request a policy administrator can replace one (or multiple) existing policy elements by the element specified through the xacmlPolicyElement parameter. As for insert operations, an update operation can only refer to one of the element types listed above. Via the deep parameter it is possible to control whether the update shall be performed completely or locally. Setting the value of the deep parameter to “true” implies that the element node referred to by the pathToNode parameter gets deleted and replaced by the element defined in the xacmlPolicyElement parameter. Setting the value of the deep parameter to “false” implies a local update. This means that the <PolicySet> or <Policy> element as selected by the pathToNode parameter gets replaced by the element defined by the xacmlPolicyElement parameter value. Further all previously existing children of the replaced element node remain children of the new the element node (i.e. the one defined through the xacmlPolicyElement parameter). Setting the deep parameter to “false” thereby allows updating <PolicySet> and <Policy> elements closer to the root without the need to reconstruct the subtree below them explicitly afterwards.

6.3.6 SelectPolicyElement

selectPolicyElement(String containerName, XPath pathToNode, String Namespace, Boolean deep, Boolean dereference)

A selectPolicyElement request allows to select <PolicySet>, <Policy> and <Rule> elements. The deep parameter allows to control whether one selects the whole subtree below the specified element or just the element node. Through the dereference parameter

one specifies whether <PolicySetIdReference> and <PolicyIdReference> elements shall be dereferenced during the read access.

6.3.7 DeletePolicyElement

deletePolicyElement(String containerName, XPath pathToNode, String Namespace, Boolean cascading)

The deletePolicyElement operation one can delete <PolicySet>, <Policy> and <Rule> elements (including all their children elements) from a policy repository. It has to be specified through the cascading parameter if referenced policy elements shall be impacted by the delete or not.

6.3.8 Analyze, Optimize, Transform and Test

Next to the so far introduced basic functionalities that an XACML PAP needs to support it is recommended to implement additional functions that allow to analyze, optimize, test and transform XACML policies. The following function signatures are defined very generic and can be populated with semantics through corresponding parameter values and attached processing logic. This follows the design approach of the OGC Web Processing Service standard.

analyze(String analyzeFunctionName, [String containerName, XPath pathToNode, String Namespace]+)

By submitting analyze requests a policy administrator can analyze parts of an XACML policy. One can e.g. select a <Rule> element of a policy (through the pathToNode parameter) and apply the is-satisfy-able analysis function (through the analyzeFunctionName parameter). Note that the underlying logic engine of the PAP needs to support the required logic reasoning.

optimize(String containerName, XPath pathToNode, String Namespace, String optimizeFunctionName, XML optimizeFunctionParameters, Boolean in-place)

The optimize function allows to optimize parts of an XACML policy following a pre-defined strategy (in-place or via return).

transform(String containerName, XPath pathToNode, String Namespace, String transformFunctionName, XML transformFunctionParameters)

The transform function allows transforming parts of an XACML policy following a pre-defined strategy and returns the transformed policy subset. This is e.g. a convenient function to realize the transformation from XACML v2.0 encoded policies into XACML v3.0 encoded ones.

test(String containerName, XPath pathToNode, String Namespace, XML xacmlAutorisationDecisionRequest, Boolean evaluationTrace)

The test operation allows evaluating parts of a policy against test XACML authorization decision requests specified by the administrators. The evaluation trace parameter specifies the amount of runtime evaluation information that shall be returned.

Import & Export interface

To import a whole policy set one can optionally provide an import function. This however corresponds logically to a simple insertPolicyElement request. For convenience reasons one could also provide an export function that uses the selectPolicyElement or transform function.

6.4 The Layered Administration Model

The Layered Administration Model (LAM) describes how to build an administration system in which one can define and administrate rights, which control the possible interactions on other policies. The resulting policies on the different layers indirectly or directly determine the possible information flows between subjects and services in SDIs. After the formal definition of the LAM (cp. section 6.4.1) some central properties of this administration model are discussed (cp. 6.4.2).

6.4.1 Definition

Definition 1: Layer x

As the name "Layered Administration Model" reveals, the entity type "Layer" represents the core of the model. A layer is the aggregation of a set of related entities. Layer 0 is the lowest layer and depending on the requirements of the use case one can add any number of additional layers. The tuple below represents the formal definition of layer x[3]:

Layer x := (C_x, O_x, R_x, S_x, S_+, PEP_x, PEP_*, PDP_x, PDP_*, SU)

- $C_x := \{c_x \mid c_x$ is a container on layer x only (short: a L_x-Container)$\}$
- $O_x := \{o_x \mid o_x$ is an object on layer x only (short: a L_x-Object)$\}$
- $R_x := \{r_x \mid r_x$ is a role on layer x only (short: a L_x-Role)$\}$
- $S_x := \{s_x \mid s_x$ is a service on layer x only (short: a L_x-Service)$\}$

[3] x is used as a variable of type positive integer

- S_{+} := {s_{+} | s_{+} is a service on layer x (x > 0) that is also part of other layers (except layer 0) (short: a L_{+}-Service)}
- PEP_x := {pe_x | pe_x is a PEP on layer x only (short: a L_x-PEP)}
- PEP_{*} := {pe_{*}| pe_{*} is a PEP on layer x that is also part of other layers (short: a L_{*}-PEP)}
- PDP_x := {pd_x | pd_x is a PDP on layer x only (short: a L_x-PDP)}
- PDP_{*} := {pd_{*} | pd_{*} is a PDP on layer x that is also part of other layers (short: a L_{*}-PDP)}
- SU := {su | su is a subject that can (amongst others) activate L_x-Roles}

Note that the elements of the sets C_x, O_x, R_x, S_x, PEP_x and PDP_x are part of exactly one layer. In contrast the elements of the sets S_{+}, PEP_{*}, PDP_{*} and SU are part of multiple layers. The support of the entity types S_{+}, PEP_{*} and PDP_{*} results from the fact that security components often need to be reusable in different contexts.

Definition 2: L_x-Container

A container (e.g. a database table or a file) is a repository that contains certain objects. Containers on layer x are called L_x-Containers. The LAM requires that all L_x-Containers have the following properties:

Requirement 2.1 The sets of L_x-Containers of the different layers are always disjoint.

Requirement 2.2 L_0-Containers hold objects of the application domain like building, street and POI features.

Requirement 2.3 Objects in L_0-Containers are never part of the knowledge base of a PDP.

Requirement 2.4 The objects in L_x-Containers (x > 0) describe L_x-Access-Rights. L_x-Access-Rights refer to L_{x-1}-Roles and L_{x-1}-Containers or L_{x-1}-Services respectively (see definition 4, 5 and 7).

Requirement 2.5 Labels for L_x-Containers (x > 0) start with the prefix "L*[integer]*_container". The placeholder *[integer]* has to be replaced by a positive natural number, which indicates the number of the layer the container belongs to.

Definition 3: L_x-Object

All Objects that can be added to an L_x-Container are called L_x-Objects. Objects that are exchanged with L_x-Services that do not use containers (e.g. OGC WPS instances) are also called L_x-Objects.

Definition 4: L_x-Role

There is a well defined set of roles on each layer that are called L_x-Roles. The following role specific requirements hold in a LAM conformant administration system:

Requirement 4.1 All access rights in a LAM compliant administration system are defined in accordance to role-based rights models. Therefore privileges are always assigned to roles exclusively.

Requirement 4.2 The set of L_x-Roles on the different layers is always disjoint.

Requirement 4.3 L_x-Role identifiers start with the prefix "urn:lam:role:layer:*[integer]*:". The placeholder *[integer]* has to be replaced by a positive natural number that indicates the layer the role belongs to.

Definition 5: L_x-Access-Right

L_x-Access-Rights (x > 0) define the permitted or denied interactions on layer x-1. An L_x-Access-Right (x > 0) always refers...

a) to subjects that have at least activated one L_{x-1}-Role and

b) to interactions with L_{x-1}-Containers or L_{x-1}-Services.

Requirement 2.4 guarantees that all access rights in LAM conformant knowledge bases are L_x-Access-Rights. Hence administrators always have the certainty that any interactions on a specific layer x (i.e. with L_x-Containers or L_x-Services) can only be performed by subjects that can activate L_x-Roles. From the subjects point of view this assures that the activation of an L_x-Role will only enable them to perform interactions on layer x.

Note that the LAM does not require that interactions on L_x-Containers or L_x-Services can only be conducted if the subject has exclusively activated L_x-Roles. This flexibility allows the subject to have activated L_x-Roles from different layers when interacting on layer x.

Definition 6: L_x-Policy (x > 0)

An L_x-Policy (x > 0) is a set of L_x-Access-Rights that determine the behaviour of PDP components using that knowledge base. L_x-Policies are defined by L_x-Objects that are stored in L_x-Containers (x > 0).

The set of services on a layer x consists of a set of L_x-Services and of a set of L_+-Services.

Definition 7: L_x-Service

Services that belong to exactly one layer are called L_x-Services. The characteristic property of L_x-Services is that they can only enable access on L_x-Containers.

Requirement 7.1 All services on layer 0 exclusively belong to this layer and are therefore always L_0-Services. This requirement ensures that the set of services on layer 0 is always disjoint with the set of services on the layers above.

The result from definition 7 and the requirements 7.1 and 2.3 is that services on layer 0 can never be used to perform administrative operations on access rights. L_0-Services only enable access on L_0-Containers (cp. definition 7) and the objects in these containers are never part of the knowledge base of a PDP (cp. requirement 2.3). Common L_0-Services in the SDI use case are e.g. OGC Web Services like WFS, WMS, SOS or WPS instances.

Requirement 7.2 The services on the layers 1 to N (i.e. L_x-Services ($x > 0$) and L_+-Services) only provide functionalities for the administration of access control policies, because they can only access L_x-Containers ($x > 0$) (cp. requirement 2.4).

Requirement 7.3 Only L_x-Services may have the following two properties:

1. The operations provided by an L_x-Service may abstract from the underlying L_x-Containers (cp. e.g. all OGC WFS operations). The effect of these abstractions is that subjects calling these operations interact with L_x-Containers without specifying them through arguments in the service requests.

2. L_x-Services may provide functionalities that are independent of any data container (cp. e.g. calculations over OGC WPS or CTS instances).

Definition 8: L_+-Service

A service instance that is used on more than one layer is called L_+-Service.

The direct result of definition 8 and requirement 7.1 is that L_+-Services can only belong to Layers from 1 to N. Hence L_+-Services are always policy administration services that are used on multiple layers.

Requirement 8.1 L_+-Services have to provide functionality for the administration of L_x-Containers and L_x-Access-Rights of different layers.

Requirement 8.2 An individual interaction with an L_+-Service always refers to exactly one layer.

Requirement 8.3 The L_x-Containers that are involved in an interaction with a L_+-Service shall always be identifiable based on the request and response. It is important to highlight that this requirement does not have to hold for L_x-Services.

Requirement 8.4 L_+-Services shall never support access to L_0-Containers.

The explanations on L_x-Services and L_+-Services reveal the reason for the two alternatives for the anchoring of L_x-Access-Rights on layer x (cp. definition 5b).

Rights that control access on L_{x-1}-Services are anchored on layer x because of the definition of the term L_{x-1}-Service. The anchoring of a right on layer x that refers to a L_+-Service can however only be established if this right explicitly refers to L_{x-1}-Containers. This fact requires that an L_x-Access-Right either has to refer to L_{x-1}-Containers or L_{x-1}-

Services. Note that this circumstance is the reason why L_+-Services may neither have property number one nor two, as expressed under requirement 7.3.

Next to the services there is a set of PEP and PDP instances on each layer, that can be divided in L_x-PEP and L_*-PEP and L_x-PDP and L_*-PDP components respectively.

Definition 9: L_x-PEP

An L_x-PEP belongs to exactly one layer and can hence only be used for the protection of L_x-Services.

Definition 10: L_*-PEP

An L_*-PEP belongs to more than one layer and is a security proxy for services of multiple layers.

Definition 11: L_x-PDP

An L_x-PDP is exclusively part of layer x. The knowledge base of L_x-PDPs consists solely of a set of L_{x+1}-Access-Rights. Consequently L_x-PDPs can only reply to authorization decision requests that where generated by PEPs on layer x and refer to interaction attempts on Lx-Containers or L_x-Services respectively.

Definition 12: L*-PDP

An L_*-PDP is used on more than one layer and evaluates authorization decision requests coming from PEPs of different layers. This implies that an L_*-PDP has to use L_x-Policies of different layers. The requirements of the LAM also imply that only L_x-Access-Rights of layer x are applied when evaluating an authorisation decision request.

Definition 13: Subject

Subjects initialize interactions with services and can in general not be exclusively associated with one layer.

It is e.g. useful if administrators of an L_1-Policy, that controls read access to features of a WFS, are allowed to have themselves read access to the data of these features. For this kind of preliminary information retrieval, they activate a suitable L_0-Role. Afterwards they activate an L_1-Role that allows them to add new rights to the L_1-policy.

6.4.2 Properties

The following sections briefly discuss some properties of LAM conformant administration systems.

6.4.2.1 Generic usability

The definition of the LAM is very generic. This allows LAM based administration systems to be used for the administration of access rights referring to any type of resources. Additionally the LAM supports the use of free to choose and different rights models on the individual layers. The only requirement towards the used rights models is that the model of choice must include a role-based model. The support of services, PEPs and PDPs that can be used on multiple layers further enables a very flexible design of the administration system infrastructure.

6.4.2.2 Effect and structure of L_x-Access-Rights

L_x-Access-Rights referring to read and delete actions on layer x-1

An L_x-Access-Right that controls which L_{x-1}-Objects can be read or deleted, can only have an influence on the possible read or delete interactions on layer x-1. The conditions under which specific subjects are allowed to perform read or delete operations on a certain subset of L_{x-1}-Objects can be based on arbitrary properties of the L_{x-1}-Objects.

L_x-Access-Rights referring to insert or delete actions on layer x-1

L_x-Access-Rights referring to read or delete actions on layer x-1 define next to others structural and content-dependant conditions (so-called L_{x-1}-Rights-Templates), that any L_{x-1}-Access-Right must fulfill in order to be insert-able in an L_x-Policy by specific subjects. An L_2-Access-Right could e.g. state that the L_1-Role r_1 shall only be allowed to insert default permit or deny XACML <Rule> elements at certain locations in the policy tree. The only degree of freedom left for the role[4] r_1 is to choose the effect of the default rule and its placement in the policy tree.

Having instances of L_{x-1}-Rights-Templates (i.e. L_{x-1}-Rights), that refer to insert or update operations on L_{x-2}-Rights, implies that these L_{x-1}-Rights-Templates not only have a direct influence on the possible L_{x-1}-Rights but also an indirect influence on the L_{x-2}-Rights that can be inserted in L_{x-2}-Policies.

A L_x-Right p_x could e.g. state that an L_{x-1}-Role r_{x-1} shall be allowed to insert L_{x-1}-Rights into an L_{x-1}-Policy that grant an L_{x-2}-Role r_{x-2} read access on L_{x-2}-Rights. This L_x-Right p_x has, next to its direct influence on the possible interactions on layer x-1 (in the example: control of insert L_{x-1}-Rights operations of the L_{x-1}-Role r_{x-1}), also an indirect influence on the achievable interactions on layer x-2. The L_x-Right p_x ensures that the L_{x-2}-Role r_{x-2}

[4] Roles are personified to simplify the formulation. A sentence like "A role xyz is allowed to..." has to be interpreted as "A subject that has activated role xyz...".

can at most have read access on L_{x-2}-Rights. The definition whether the L_{x-2}-Role r_{x-2} shall have read access, and for which subset of the L_{x-2}-Rights, is however solely in the area of responsibility of the L_{x-1}-Role r_{x-1}.

We now extend the example given above by an L_x-Right q_x, that additionally grants the L_{x-1}-Role r_{x-1} to define L_{x-1}-Rights, that in turn grant the L_{x-2}-Role r_{x-2} to insert L_{x-2}-Rights, that further grant a certain L_{x-3}-Role r_{x-3} to define L_{x-3}-Rights, that in the end allow a specific L_1-Role r_1 to define some L_1-Rights that refer to a specific WFS instance. Note that the described L_x-Right q_x only has a direct influence on the possible L_{x-1}-Rights but has an indirect influence on the possibly emerging rights on all layers below.

The example demonstrated that policies on layer x set the frame for the possible interactions on all underlying layers. It is therefore appropriate to say, that the policies on a layer x span an interaction space for each layer below. An interaction space of layer x defines the set of policy compliant interactions on layer x. In LAM based administration systems, L_x-Rights, created in the interaction space of layer x can only restrict the interaction spaces that were already spanned by the policies of the layers above.

6.4.2.3 Horizontal and vertical distribution of administrative tasks

The rights defined in the policies of the different layers ensure that the right-specific requirements of the LAM always hold. Additionally one can add LAM conformant right definitions than realize a flexible horizontal and vertical distribution of the administrative tasks and rights respectively.

Horizontal distribution

To ensure that the administrators of L_x-Rights can handle the required administrative tasks, it is necessary to divide the work between multiple administrative roles and administrators. It is the aim to find an appropriate distribution that ensures that the tasks/rights assigned to the individual administrative roles can be handled by the subjects activating these roles. Further the set of assigned privileges must not give too much power to individual subjects.

The distribution of privileges between L_x-Roles on Layer x is called *horizontal distribution of rights*. This horizontal distribution is defined by the L_{x+1}-Policies. Assuming that the languages used to define the L_{x+1}-Rights support the definition of expressive, fine-grained rights, it is possible to implement a horizontal partitioning in multiple dimensions. One could e.g. grant an L_1-Role to insert, read, delete and update L_1-rights that refer to specific...

- computers (e.g. to the computer with an IP-address equal "123.123.123.123")

- service classes (e.g. to services of type WFS and WMS)
- service instances (e.g. to a WFS under www.awfs.com)
- operations (e.g. to WFS GetFeature and Transaction/insert operations)
- feature classes (e.g. to building, street and POI feature classes)
- features (e.g. to building features within the USA)
- attribute classes (e.g. to the location and price attribute of the building feature type)
- attributes (e.g. to the price attribute of building features with a value less-than 1 million)
- request parameters (e.g. to the featureVersion attribute of GetFeature queries)
- L_0-Roles and subject properties (e.g. to all L_0-Roles in domain A)
- Environment states (e.g. to accesses between 8 am to 6 pm).

In large service-oriented architectures, the number of L_0-Roles, L_0-Services, L_0-Objects etc. on layer 0 is in general very large. It is therefore required to achieve an adequate horizontal distribution of the administrative tasks on layer 1 by defining corresponding L_2-Policies. In cases where the administration of the needed L_2-Policies is still very complex, it might be helpful to add additional layers and to define adequate horizontal distributions.

Vertical distribution

Through L_x-Rights that refer to insert or update operations on layer x-1 one can already indirectly predefine parts of or constraints on the authorization semantics that are to be expressed on layer x-1. Thanks to this property of the definable rights in LAM based administration systems one can distribute administrative tasks not only horizontally but also vertically -- i.e. between roles of multiple layers.

The *vertical distribution* of rights greatly simplifies the definition and administration of rights, as the administrators on the different layers only have to address certain aspects of the right definition tasks. By consecutively reducing the interaction spaces, the freedom of choice of the administrators is reduced step by step. The elimination of unnecessary degrees of freedom when defining rights enhances the likelihood that the policies are defined as intended.

It is important to highlight, that the capability of LAM based administration systems, to support a vertical distribution of the administrative tasks, is an important feature in the large and hierarchically organized SDIs and SOAs. In these use cases the definition of rights occurs on multiple organizational levels and the subjects on higher levels often only specify mandatory but very broad and incomplete guidelines. These guidelines are than step by step refined on the subsequent organizational layers. By defining policies on the different layers one can naturally model these administrative structures and chains. At

the bottom of the hierarchy L_1-Policies will arise, that always fulfill all the guidelines that were defined by the administrators of the organizational levels above.

Further details on how to implement and use LAM and XACML compliant administration systems implementation can be found in [4].

7 OWS-9 Policy Administration Point Implementation

This section describes the conceptual design and implementation of the OWS-9 Policy Administration Point (PAP) which provides functions over the Web to create, update, delete, import, export and transform GeoXACML v1.0.1 and XACML v2.0 policies. This proof-of-concept implementation of an XACML PAP represents a slightly simplified realization of the basic PAP interfaces as introduced in section 6.2.1.

The PAP is implemented as a web-application with a thin client interface for a web browser. It also implements a simple web service interface to enable other systems, clients or web services, in particular a Policy Decision Point (PDP), to obtain a particular policy for authorization decision making.

Chapter 7 is structured according to the ODP breakdown: Engineering, Computational, Information Viewpoint. The chapter also includes a section on interoperability between policies and PDP implementations and a section addressing the transformation of XACML v2.0 and GeoXACML v1.0.1 policies to XACML v3.0 conformant representations.

Note: Because GeoXACML v1.0.1 is a superset of XACML v2.0, it is up to the policy writer if a GeoXACML or a XACML policy gets created. It should also be pointed out, that a XACML compliant policy will run on any XACML or GeoXACML compliant PDP, but a GeoXACML policy will never run on a XACML PDP!

7.1 Engineering Viewpoint

The purpose of the PAP can best be described by the following two use cases:

7.1.1 Use Case 1: “Policy Administration

The PAP provides a web-based application thin-client that provides a tree view of a policy or policies and which can be used for policy administrators to create, update and delete XACML 2.0 and GeoXACML 1.0.1 policies.

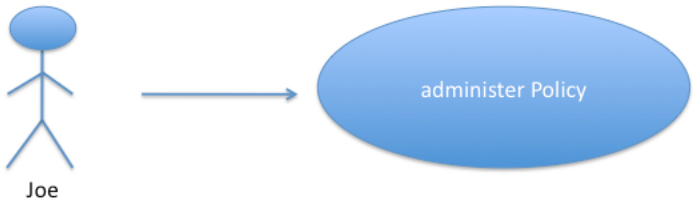

Figure 2 — Use Case 1 Engineering Viewpoint

7.1.2 Use Case 2: "Policy Obtainment

The PAP serves as a Web Service to allow any other system, service or client to obtain a particular policy.

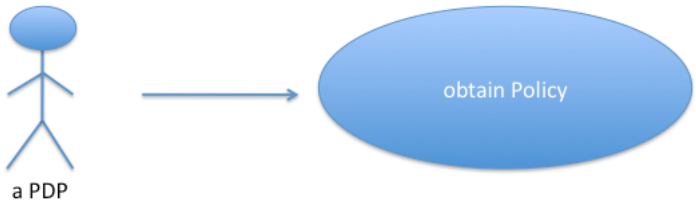

Figure 3 — Use Case 2 Engineering Viewpoint

7.2 Computational Viewpoint

The PAP provides two interfaces, one for a web-application to display a tree view of XACML 2.0 and GeoXACML 1.0.1 policies and one for requesting existing policies. The high level use of these interfaces is illustrated in the figure below.

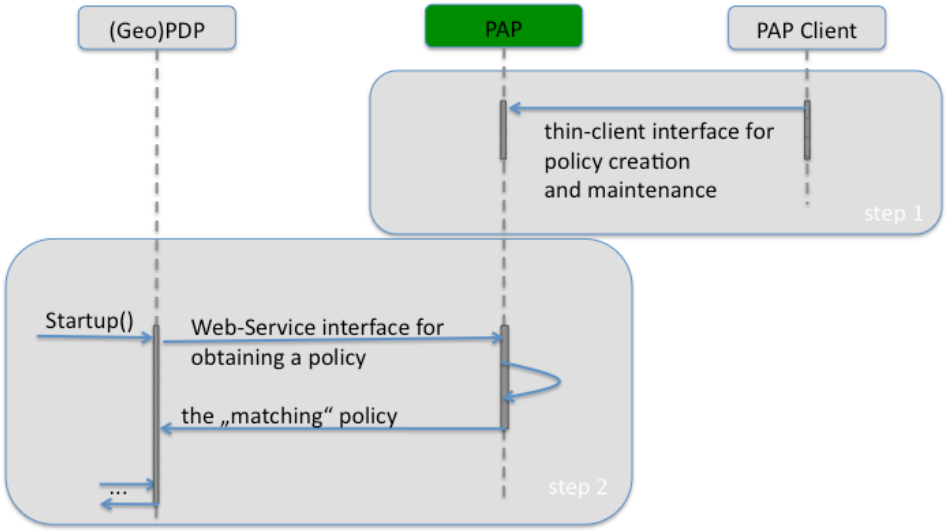

Figure 4 — High level interactions with the PAP

7.2.1 Thin-Client Interface

This interface can only be leveraged by a web-browser supporting HTML and JavaScript. For rendering, CSS is used. The PAP is deployed with a thin-client that can be consumed by any JavaScript enabled Web Browser via the URL:

http://ows9.secure-dimensions.org/xs/index.html

The thin client provides all required functions to support the administration of policies. Pro-active menus ensure that only valid policies can be created. In addition, a list of data types and functions (using short names for readability) is available via pull-down menu to ensure correct spelling. Furthermore, the functions are structured such that only some can be used for match making where others can also be used in a complex condition.

The current version of the PAP provides basic support, sometimes described as RESTful. Please note that it is currently not supported to create or change policies via the RESTful interface. To switch to the RESTful interface, please refer to the following URL:

http://ows9.secure-dimensions.org/policy_sets/

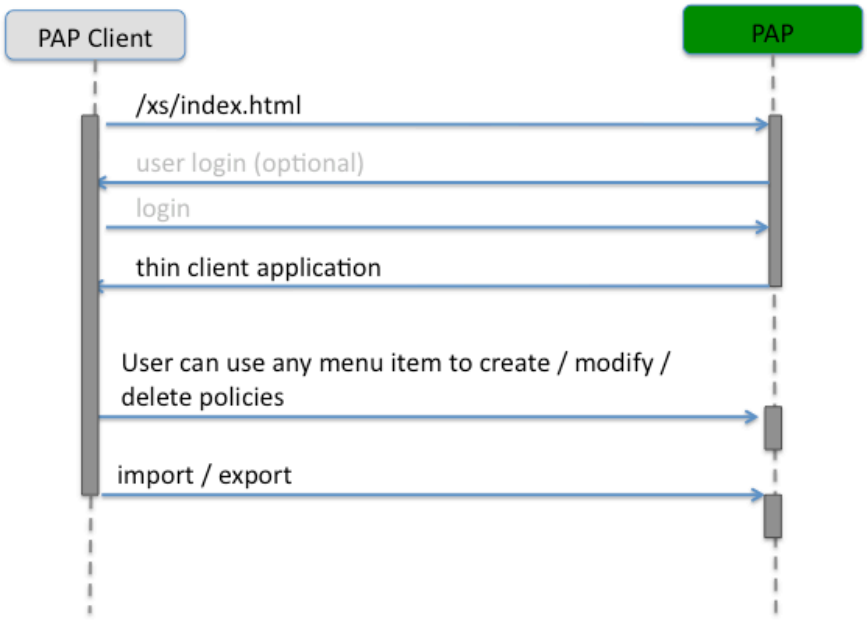

Figure 5 — User client interaction with the PAP

7.2.2 Web Service Interface

The Web Service interface can be executed via HTTP/Get using Key-Value-Pair encoding to shape the request. Currently, this interface can be executed using the following URL:

http://ows9.secure-dimensions.org/cgi-bin/PAP

The following parameters (keys) are supported:

- `PolicySetId=value` tasks the PAP to return the policy where the attribute "PolicySetId" equals *`value`*. The root element of the returned policy is a <PolicySet> element

- `PolicyId=value` tasks the PAP to return the policy where the attribute "PolicyId" equals *`value`*. The root element of the returned policy is <Policy> element.

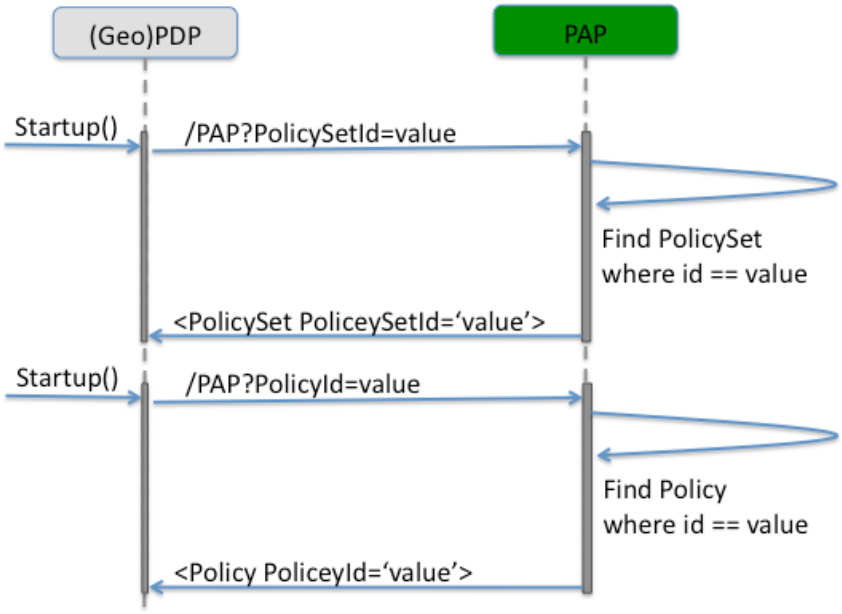

Figure 6 — Interaction with the PAP's Web Service Interface

7.3 Information Viewpoint

The PAP must ensure that the creation and modification of policies is such that the exported XACML 2.0 or GeoXACML 1.0.1 policy is compliant to the XML Policy Schema published by OASIS. In order to provide that, each XML element of the Policy is stored as information item that has relationships to other work items. These relationships provide the logic of the policy concerning its structure and – of course – its semantics.

The PAP's internal information objects are created 1:1 from the Policy Schema and are stored into a MySQL database. The functionality provided by the thin-client enables the user to insert / modify / delete (for admins only) rows in a table of the database.

7.3.1 Internal design

Internally, the PAP uses a MySQL database for storing the policies. For the purpose of storage, the PAP leverages a table structure that is driven by the XML Schema defined by XACML 2.0.

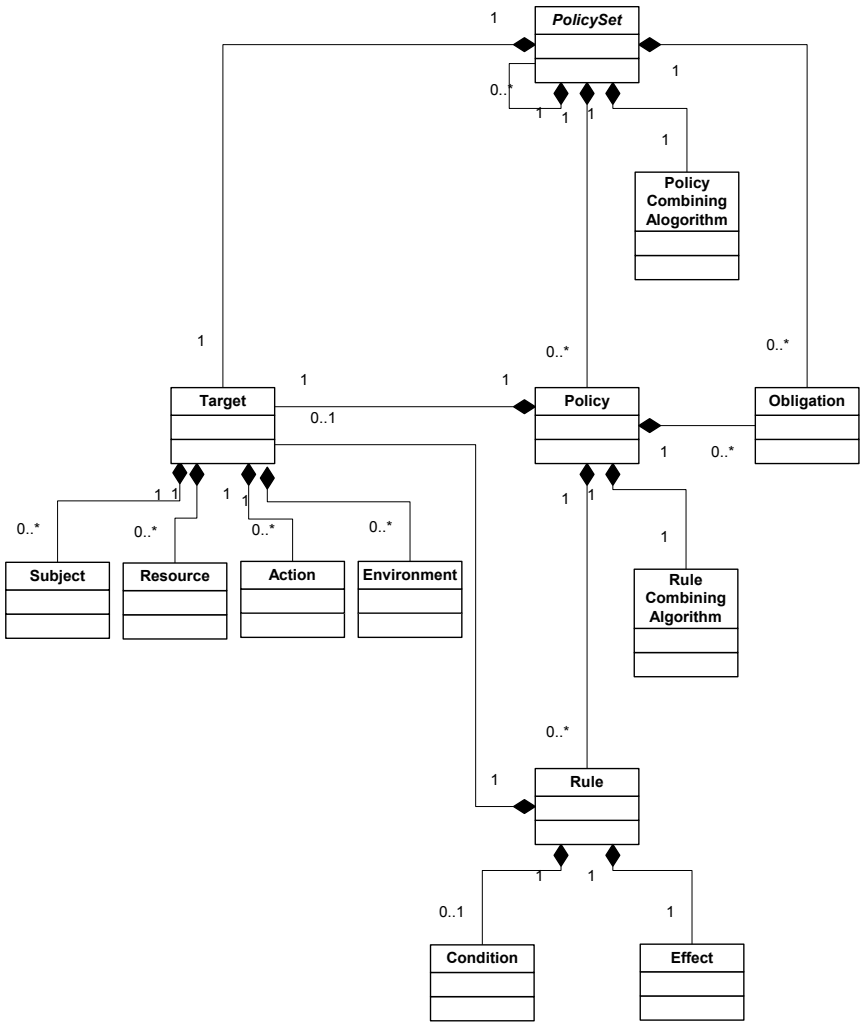

Figure 7 — Policy structure describing the XML Schema structure for policies

The most important aspect of the database design is that it serves three purposes:

1) Policy import: An entire policy file can be uploaded to the PAP and is inserted into the database.

2) Policy export: Policies (in parts or full) can be exported as a XACML or GeoXACML file constructed from the database.

3) The support to a policy administrator by providing functions to copy & paste parts of a policy.

It is not feasible to include a figure of the Entity-Relationship diagram because the level of detail is too large. However, we plan to provide details at the end of OWS-9.

7.3.2 User client

The user interface for the policy administrator uses a tree like representation of the policy. It thereby hides the XML from the user and avoids

1) typos, as the PAP provides a list of all XACML / GeoXACML data types and functions

2) incorrect structuring of the policy, as pro-active menus are in place that prevent the creation of wrong policy elements or the use of wrong functions

But most importantly, it supports the user to copy & paste parts of existing policies which enables the template approach: A Template policy can be copied and inserted in the appropriate position of the policy tree.

An “Expand All” / “Collapse All” function can be applied to any part of the policy tree. This increases the visibility of important parts of the policy.

For the purpose of differentiating access rights to the content of the database, the user client can only be used after login:

1. The user admin has the right to create and delete elements from the policy

2. Any registered user has limited rights: delete is not possible

Note: At the moment, the PAP runs in collaborative mode which means that all registered users share the same view on any existing policy.

The following figure illustrates the user client displaying parts of a policy.

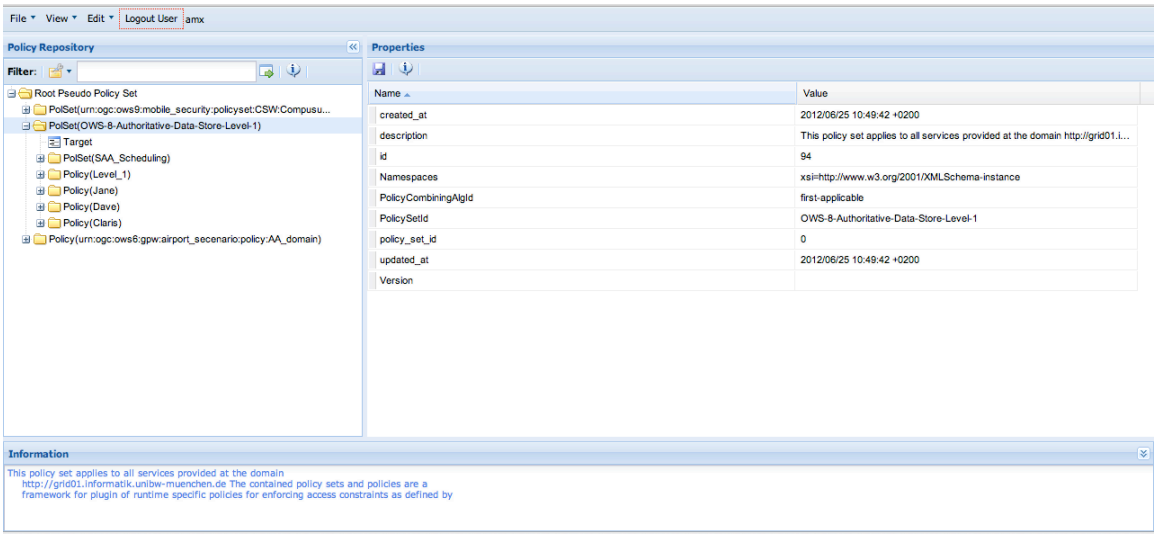

Figure 8 — PAP User Client screenshot

A short demo video is uploaded to the OWS-9 portal which introduces the basic functions: https://portal.opengeospatial.org/files/?artifact_id=51510

7.4 Web Service interface

The Web Service interface is a simple HTTP/GET endpoint which execution can be controlled thru parameters. To obtain an XML encoding of a policy (in parts or full), the use of the following service parameter is supported

- `PolicySetId` will return a <PolicySet> containing a GeoXACML/XACML policy where the PolicySetId attribute's value equals the value from the request parameter. The root element of the policy is <PolicySet>.

- `PolicyId` will return a <Policy> containing a GeoXACML/XACML policy where the PolicyId attribute's value equals the value from the request parameter. The root element of the policy is <Policy>.

The following URL is an example to obtain a <PolicySet> based policy:

http://ows9.secure-dimensions.org/cgi-bin/PAP?
PolicySetId=urn:ogc:ows9:mobile_security:policyset:CSW:Compusult

```
<PolicySet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:oasis:names:tc:xacml:2.0:policy:schema:os access_control-xacml-2.0-policy-schema-os.xsd"
  xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os"
 xmlns:xacml context="urn:oasis:names:tc:xacml:2.0:context:schema:os"
 PolicySetId="urn:ogc:ows9:mobile_security:policyset:CSW:Compusult"
 PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:first-applicable">
  <Description>Compusult CSW definitions</Description>
```

The following URL is an example to obtain a <Policy> based policy:

http://ows9.secure-dimensions.org/cgi-bin/PAP?
PolicyId=urn:ogc:ows9:mobile_security:policy:request:RoleA

```
<Policy PolicyId="urn:ogc:ows9:mobile_security:policy:request:RoleA
  RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
  <Description>Policy for matching the REQUEST context -> PEP will receive an
    obligation to request MRP on the RESULT context ...</Description>
```

7.5 Interoperability

The implemented PAP enables policy administration including ALL XACML v2.0 data types and functions including all those which are marked optional in the standard. Regarding GeoXACML v1.0.1, the PAP provides the data type geometry and all functions from both GeoXACML v1.0.1 conformance classes (BASIC and STANDARD). This provides the ultimate superset of data types and functions to the policy administrator.

In order to perform authorization decisions by a PDP, it must be ensured that the PDP has implemented at least all data types and functions used in a particular policy instance. The following table provides more details on the interoperability, regarding the use of XACML v2.0 and GeoXACML 1.0 conformance classes.

Table 1 — Policy / PDP interoperability

Policy using data type + functions from	XACML based PDP	GeoXACML based PDP
XACML 2.0 Core	XACML 2.0 core	XACML 2.0 core
XACML 2.0 MRP / HRP	XACML 2.0 MRP / HRP	XACML 2.0 MRP / HRP
XACML 2.0 RBAC	XACML 2.0 RBAC	XACML 2.0 RBAC
GeoXACML 1.0 BASIC	Not interoperable	GeoXACML 1.0 BASIC
GeoXACML 1.0 STANDARD	Not interoperable	GeoXACML 1.0 STANDARD

Because the XACML 2.0 Core standard lists many data types and functions as optional to be implemented, the policy administrator must check on a policy instance basis if the PDP has implemented the appropriate optional data types and functions.

7.6 XACML v2.0 to XACML v3.0 Translation

Next to the mentioned functionalities the PAP provides a “transform” function that allows administrators to transform any XACML v2.0 policy into a semantically equal but XACML v3.0 encoded form. Applying the “transorm” function on GeoXACML v1.0.1 policies will result in a XACML v3.0 policy + spatial extension as defined in the GeoXACML v1.0.1 specification (cp. section 6.1).

The proof-of-concept realization of the “transform” function can currently be accessed sending HTTP POST (with content-type text/xml) messages to:

http://ows9.secure-dimensions.org/cgi-bin/XACML2to3

Accessing the service can hence take place using a HTTP POST plug-in for your browser (e.g. the "DEV HTTP CLIENT" extension for Google Chrome). Alternatively you could also use a Linux command like curl, so e.g. curl -X POST @filename http://ows9.secure-dimensions.org/cgi-bin/XACML2to3 where filename is the name of the file containing

the policy you want to get translated. In both cases the response will contain the XACML 3.0 compliant policy.

For testing purposes we have successfully transformed all 372 XACML 2.0 policies from the 2.0 conformance testing (published by OASIS) and various GeoXACML 1.0 policies including the OWS6, OWS8 and the current OWS9 Mobile Thread policy for CSW.

Example policies for OWS-9 can be found here:
XACML 2.0: https://portal.opengeospatial.org/files/?artifact_id=50587
XACML 3.0: https://portal.opengeospatial.org/files/?artifact_id=50588
(is the transformation result of the above)

GeoXACML 1.0: https://portal.opengeospatial.org/files/?artifact_id=50591
XACML 3.0 + spatial: https://portal.opengeospatial.org/files/?artifact_id=50592
(is the transformation result of the above)

8 Future Work items

The following subsections briefly describe some interesting topics that need to be addressed next.

8.1.1 Standardisation of an XACML Policy Administration Web Service

The (Geo)XACML PAP interfaces described in this ER should be used as a starting point for a standardized PAP. A PAP service standard will provide interoperability of administrative tasks in GeoXACML based access control systems and will further support an easier applicability and implementation of XACML or GeoXACML based access control systems.

8.1.2 XACML PAWS Clients with spatial fuctionalities

Another interesting future work item is the development of a spatial PAP Client that specifically addresses the spatial characteristics of features and spatial rules. Embedding such a spatial PAP Client in an OTS GIS client will leverage many advantages during the administration of (spatial) rules for geospatial data objects. A policy administrator could e.g. easily visualize the features affected by a XACML rule or could use the GIS edit functionalities to create geometries that can then be used to define GeoXACML policy elements.

8.1.3 GeoXACML 3.0

As pointed out in section 6.1 there is the need to derive a XACML v3.0 compliant spatial extension. The GeoXACML SWG therefore needs to bring forward work towards an XACML v3.0 compliant OGC GeoXACML v3.0 standard.

