

# Open Geospatial Consortium Inc.

Date: 2009-10-09

Reference number of this document: OGC 09-035

Version: 0.3.0

Category: OGC® Public Engineering Report

Editor(s): Rüdiger Gartmann, Lewis Leinenweber

## OGC® OWS-6 Security Engineering Report

Copyright © 2009 Open Geospatial Consortium, Inc.

To obtain additional rights of use, visit <http://www.opengeospatial.org/legal/>.

### 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:	OpenGIS® Engineering Report
Document subtype:	NA
Document stage:	Approved for Public Release
Document language:	English

## **Preface**

This Engineering Report describes work accomplished to investigate and implement security for OGC web services during the OGC Web Services Testbed, Phase 6 (OWS-6).

Suggested additions, changes, and comments on this draft 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.

The changes made in this document version, relative to the previous version, are tracked by Microsoft Word, and can be viewed if desired. If you choose to submit suggested changes by editing this document, please first accept all the current changes, and then make your suggested changes with change tracking on.

*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 and Change Requests into the OGC 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 for their 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)

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, LISAssoft, Luciad, Lufthansa, NOAA MDL, Northrop Grumman TASC, OSS Nokalva, PCAvionics, Snowflake, Spot Image/ESA/Spacebel, STFC, UK, UAB CREAM, Univ Bonn Karto, Univ Bonn IGG, Univ Bundeswehr, Univ Muenster IfGI, Vighetel, Yumetech.

<b>Contents</b>	<b>Page</b>
1 Introduction.....	1
1.1 Scope .....	1
1.2 Document contributor contact points .....	2
1.3 Revision history.....	2
1.4 Future work .....	2
2 References.....	2
3 Terms and definitions .....	4
3.1 Security domain.....	4
3.2 Authentication .....	5
3.3 Authorization.....	5
3.4 Non-repudiation.....	5
3.5 Proxy .....	5
3.6 Trust.....	5
4 Conventions .....	5
4.1 Abbreviated terms .....	5
5 Security Overview .....	6
6 Security Requirements.....	6
6.1 Confidentiality.....	6
6.2 Authenticity .....	7
6.3 Integrity .....	7
6.4 Non-Repudiation .....	7
6.5 Access Control .....	8
6.6 Audit/Auditing.....	8
7 Cryptography .....	8
7.1 Symmetric Cryptography .....	8
7.2 Asymmetric Cryptography .....	8
7.2.1 Encryption.....	9
7.2.2 Digital Signatures.....	9
8 Trust .....	9
8.1 Introduction .....	9
8.2 Public Key Infrastructure (PKI) Concept.....	10
8.3 Trust Establishment.....	11
8.3.1 Direct Trust .....	11
8.3.2 Direct Brokered Trust .....	11
8.3.3 Indirect Brokered Trust.....	12
9 Abstract Security Architecture.....	12
9.1 Security Extensions to Existing OGC Architecture .....	12
9.2 Static View: Security Interactions.....	13

9.3	Dynamic View: Security Workflows .....	14
10	SOAP Binding .....	15
10.1	Relevant Standards .....	15
10.2	Protocol .....	16
10.3	Interactions .....	16
10.3.1	Binding to a Secured Service .....	16
10.3.2	Authorization .....	30
11	RESTful Security for OGC Web Services.....	32
11.1	Bindings.....	33
11.2	Relevant Standards & Specifications .....	33
11.3	Standards and Protocols .....	33
11.4	Interactions .....	38
11.4.1	Register User with OpenID.....	39
11.4.2	Establish Trust .....	39
11.4.3	Registering Application with OpenID .....	42
11.4.4	Creating a Certificate Authority (CA) .....	42
11.4.5	Create a Certificate Request.....	43
11.4.6	Extract Public and Private Keys .....	44
11.4.7	Create a Web Application Entry on the OpenID Server.....	44
11.4.8	Cross-Domain Authority Delegation .....	46
11.4.9	Application Consumer OAuth request.....	47
11.4.10	GeoBPMS Data Provider (SPS) Response .....	48
11.4.11	Target Service Request to Service Provider .....	50
12	XACML Policy Models.....	50
12.1	XACML Policy Language Model .....	51
12.2	XACML Data Flow Policy Model .....	52
12.3	Obligations .....	54
12.4	Spatial Authorization.....	54
12.4.1	GeoXACML .....	55
12.4.1.1	GeoXACML's Geometry Model and Spatial Functions.....	55
12.4.1.2	Summary of GeoXACML's Capabilities .....	56
12.4.2	XACML with Spatial Obligations .....	56
12.4.2.1	WMS Example .....	57
12.4.2.2	WFS Example.....	62
13	OWS-6 Use Cases.....	73
13.1	OWS-6 WS-Security Deployment .....	73
13.2	Within One Security Domain .....	75
13.3	Between Trusted Security Domains .....	75
13.4	Between Un-trusted Security Domains (Trust Establishment) .....	78
13.5	Between Un-trusted Security Domains (Forwarding).....	79
14	RFQ Use Cases .....	80
14.1	Within One Security Domain .....	81
14.2	Between Trusted Security Domains .....	83
14.3	Between Un-trusted Security Domains (Trust Establishment) .....	85

15	Future Work and Unsolved Issues .....	86
15.1	Security Metadata .....	87
15.1.1	Technical Metadata .....	87
15.1.2	Content Metadata .....	87
15.1.3	Capabilities .....	88
15.2	Security Error Messages .....	88
15.3	Blocking or Filtering .....	89
15.4	Standardization of Obligations .....	89
15.5	Architecture of the Access Control System .....	90
15.6	Altering Results .....	90
15.7	Performance .....	90
15.8	OGC Service Types .....	91

<b>Figures</b>	<b>Page</b>
<b>Figure 1, Abstract Security Components .....</b>	<b>13</b>
<b>Figure 2, Abstract Security Interactions .....</b>	<b>15</b>
<b>Figure 3, Binding to a Secured Service .....</b>	<b>17</b>
<b>Figure 4, Example GetMetadata Request .....</b>	<b>17</b>
<b>Figure 5, Example WS-MetadataExchange Response .....</b>	<b>18</b>
<b>Figure 6, Example WS-Policy description .....</b>	<b>20</b>
<b>Figure 7, Example RequestSecurityToken request .....</b>	<b>22</b>
<b>Figure 8, Example RequestSecurityToken response .....</b>	<b>26</b>
<b>Figure 9, GetCapabilities SOAP request with SAML holder-of-key assertion and signature (Example) .....</b>	<b>29</b>
<b>Figure 10, Authorization Process .....</b>	<b>31</b>
<b>Figure 11, RESTful Workflow and Security Deployment .....</b>	<b>32</b>
<b>Figure 12, OpenID Server User Profile Entry/Update .....</b>	<b>39</b>
<b>Figure 13, Establish Cross-Domain Trust .....</b>	<b>40</b>
<b>Figure 14, Create a new Web Application Entry on the OpenID Server .....</b>	<b>45</b>
<b>Figure 15, OpenID Application Consumer Key .....</b>	<b>45</b>
<b>Figure 16, OpenID Server Interface to enter User's Identity Profile .....</b>	<b>46</b>
<b>Figure 17, Delegation of Authority using OAuth and OpenID .....</b>	<b>47</b>
<b>Figure 18, XACML Language Policy Model .....</b>	<b>51</b>
<b>Figure 19, XACML v1.0 Data Flow Model .....</b>	<b>53</b>
<b>Figure 20, Policy for WMS with Spatial Obligations .....</b>	<b>60</b>
<b>Figure 21, WMS GetMap Response .....</b>	<b>61</b>

<b>Figure 22, Restricted GetMap Response .....</b>	<b>62</b>
<b>Figure 23, WFS GetFeature Request submitted to PEP .....</b>	<b>66</b>
<b>Figure 24, PDP Authorization Decision Request .....</b>	<b>67</b>
<b>Figure 25, PolicySet used for authorization decision .....</b>	<b>71</b>
<b>Figure 26, Authorization Decision Response with Obligation (OGC Filter) .....</b>	<b>72</b>
<b>Figure 27, WFS GetFeature Request with Filter Encoding.....</b>	<b>73</b>
<b>Figure 28: Basic Security Deployment .....</b>	<b>74</b>
<b>Figure 29: Actual OWS-6 Deployment.....</b>	<b>75</b>
<b>Figure 30: Security between Trusted Security Domains .....</b>	<b>76</b>
<b>Figure 31: Interactions between Trusted Domains .....</b>	<b>76</b>
<b>Figure 32: Trusted Domains, alternative 1 .....</b>	<b>77</b>
<b>Figure 33: Trusted Domains, alternative 2 .....</b>	<b>78</b>
<b>Figure 34: Trust Establishment between Non-Trusted Domains.....</b>	<b>79</b>
<b>Figure 35: Un-trusted Domains - Forwarding.....</b>	<b>80</b>



# OGC® OWS-6 Security Engineering Report

## 1 Introduction

### 1.1 Scope

This Engineering Report describes work accomplished during the OGC Web Services Testbed, Phase 6 (OWS-6) to investigate and implement security measures for OGC web services. This work was undertaken to address requirements stated in the OWS-6 RFQ/CFP originating from a number of sponsors, from OGC staff, and from OGC members.

The tasks undertaken to satisfy these requirements provided results related to three different approaches:

- Web services security using XACML policies with spatial obligations and related software implementations;
- Web services security using GeoXACML policies and related software implementations; and
- RESTful web services security using OpenID / OAuth and related software implementations.

Each approach and its solution provided opportunities to experiment with existing security specifications and standards to demonstrate applicability, interoperability and to identify potential implementation and standards issues where future work may be required.

The outcome from these solutions, which was based on a variety of technology, standards, and engineering design choices, offers insights into ways to apply existing security standards from W3C, OASIS, and others with the architecture of OGC web services and standards.

## 1.2 Document contributor contact points

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

Name	Organization
Rüdiger Gartmann	con terra GmbH
Lewis Leinenweber	BAE Systems
Jan Hermann	Technische Universität München
Pat Cappeleare	Vightel

## 1.3 Revision history

Date	Release	Editor	Primary clauses modified	Description
2008/11/17	0.0.1	RG	All	Document initialized
2009/04/01	0.0.2	RG	12	Policy encoding
2009/04/03	0.0.3	RG	15	Unsolved issues
2009/06/15	0.0.4	LEL	4,7,8,10, 13	General edits; added RFQ Use Cases
2009/06/18	0.0.5	LEL	Various	Minor edits
2009/06/19	0.0.6	LEL	Various	Minor edits
2009/07/15	0.0.7	LEL	11	Added RESTful security; added reference material for GeoXACML and ER
2009/07/17	0.0.8	LEL	Various	Overall document edits and update
2009/08/03	0.0.9	RG	Various	Several edits
2009/08/12	0.0.10	LEL	Various	Minor edits
2009/10/08	0.3.0	Carl Reed	Various	Ready document for posting as Public ER

## 1.4 Future work

Improvements in this document are desirable to address open issues; to correct errors or enhance existing document content.

See also section 15 for a description of web service security-related issues arising during this testbed and for topics to be considered for future testbeds.

## 2 References

The following documents are referenced in this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

As this document is not an implementation specification, there are no normative references. However, the following documents are considered to be relevant to this report and useful for the reader.

[1] OGC 06-121r3, OpenGIS® Web Services Common Specification

[2] ISO 19105:2000, Geographic information — Conformance and Testing

- [3] OGC 06-107r1, OWS-4 Trusted GeoServices IPR
- [4] OGC 04-095, OpenGIS Filter Encoding Implementation Specification, Version 1.1
- [5] OGC 07-158, [Wrapping OGC HTTP-GET/POST Services with SOAP](#), OGC Discussion Paper.
- [6] Common Criteria, [http://www.niap-ccevs.org/cc-scheme/cc\\_docs/](http://www.niap-ccevs.org/cc-scheme/cc_docs/)
- [7] eXtensible Access Control Markup Language (XACML) Version 2.0, OASIS Standard, 1 Feb 2005, [http://docs.oasis-open.org/xacml/2.0/access\\_control-xacml-2.0-core-spec-os.pdf](http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf)
- [8] eXtensible Access Control Markup Language (XACML) Version 1.1, OASIS Committee Draft, 7 August 2003, <http://www.oasis-open.org/committees/xacml/repository/cs-xacml-specification-1.1.pdf>
- [9] 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](http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-rbac-profile1-spec-os.pdf)
- [10] eXtensible Access Control Markup Language (XACML) Version 1.0, OASIS Standard, 18 February 2003, <http://www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.0.pdf>
- [11] OGC 04-095, OpenGIS® Filter Encoding Implementation Specification
- [12] Security Assertion Markup Language (SAML) V2.0, OASIS Standard, 15 March 2005, <http://docs.oasis-open.org/security/saml/v2.0/saml-2.0-os.zip>
- [13] Web service Description Language (WSDL) v2.0, <http://www.w3.org/TR/2007/REC-wsdl20-20070626/>
- [14] Web Services Addressing (WS-Addressing), W3C Member Submission, 10 August 2004, <http://www.w3.org/Submission/2004/SUBM-ws-addressing-20040810/>
- [15] Web Services Metadata Exchange, W3C Member Submission, 13 August 2008, <http://www.w3.org/Submission/2008/SUBM-WS-MetadataExchange-20080813/>
- [16] Web Services Policy 1.2 - Framework (WS-Policy), W3C Member Submission, 25 April 2006, <http://www.w3.org/Submission/2006/SUBM-WS-Policy-20060425/>
- [17] Web Services Security: SOAP Message Security 1.1 (WS-Security), OASIS Standard Specification, 1 February 2006, <http://www.oasis->

[open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf](http://open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf)

- [18] WS-SecurityPolicy 1.2, OASIS Standard, 1 July 2007, <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.pdf>
- [19] WS-Trust 1.3, OASIS Standard, 19 March 2007, <http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.pdf>
- [20] XML Encryption Syntax and Processing, W3C Recommendation, 10 December 2002, <http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/>
- [21] XML Signature Syntax and Processing (Second Edition), W3C Recommendation, 10 June 2008, <http://www.w3.org/TR/2008/REC-xmldsig-core-20080610/>
- [22] OpenID Authentication 2.0 – Final, [http://openid.net/specs/openid-authentication-2\\_0.html](http://openid.net/specs/openid-authentication-2_0.html)
- [23] OpenID Attribute Exchange 1.0 – Final, [http://openid.net/specs/openid-attribute-exchange-1\\_0.html](http://openid.net/specs/openid-attribute-exchange-1_0.html)
- [24] OpenID Simple Registration Extension 1.0, [http://openid.net/specs/openid-simple-registration-extension-1\\_0.html](http://openid.net/specs/openid-simple-registration-extension-1_0.html)
- [25] OAuth Core 1.0 Specification, <http://oauth.net/core/1.0/>
- [26] NIST Guide to Secure Web Services, SP800-95, August 2007, <http://csrc.nist.gov/publications/nistpubs/800-95/SP800-95.pdf>
- [27] Guide for the Security Certification and Accreditation of Federal Information Systems, NIST 800-37

NOTE This OWS Common Specification contains a list of normative references that are also applicable to this Implementation Specification.

### **3 Terms and definitions**

For the purposes of this Engineering Report, the definitions specified in Clause 4 of the OWS Common Implementation Standard [OGC 06-121r3] shall apply. In addition, the following terms and definitions apply.

#### **3.1 Security domain**

An environment or context that is defined by security policies, security models, and a security architecture, including a set of system resources and set of system entities that are authorized to access the resources. An administrative domain may contain one or more security domains. The traits defining a given security domain typically evolve over time.

### **3.2 Authentication**

Verification that a potential partner in a conversation is capable of representing a person or organization.

### **3.3 Authorization**

Determination whether a subject is allowed to have the specified types of access to particular resource.

### **3.4 Non-repudiation**

Assurance that the sender of information is provided with proof of delivery and the recipient is provided with proof of the sender's identity, so neither can later deny having processed the information.

### **3.5 Proxy**

An agent that acts on behalf of a requester to relay a message between a requester agent and a provider agent. The proxy appears to the provider agent Web service to be the requester.

### **3.6 Trust**

The characteristic that one entity is willing to rely upon a second entity to execute a set of actions and/or to make set of assertions about a set of subjects and/or scopes.

## **4 Conventions**

### **4.1 Abbreviated terms**

Most of the abbreviated terms listed in Subclause 5.1 of the OWS Common Implementation Standard [OGC 05-008] apply to this document, plus the following abbreviated terms.

CA	Certificate Authority
GeoPDP	Geospatially-enabled Policy Decision Point
GeoXACML	Geospatial eXtensible Access Control Markup Language
GPW	GeoProcessing Workflow
IC-ISM	Intelligence Community Metadata Standard for Information Security Marking
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point

PKI	Public Key Infrastructure
SAML	Security Assertion Markup Language
SOAP	Simple Object Access Protocol
STS	Security Token Service
XACML	eXtensible Access Control Markup Language

## 5 Security Overview

Security has several aspects, which have to be addressed separately. OWS-6 is focused on access control, which may require security features such as confidentiality, integrity, authenticity and non-repudiation of the service communication

Not all security features are required in every use case. Moreover, security requirements have to be evaluated and security measures have to be derived individually. Thus, a security framework defines a toolbox, providing a set of security mechanisms, all of them fulfilling different security requirements. After deciding on the required level of security, a system's designer is then able to select those security features needed in a certain use case.

This testbed did not attempt to define any concrete set of security requirements or attack scenarios but examined one approach to the use of the technologies.

## 6 Security Requirements

Security requirements differ for different applications. Thus, an actual security solution always has to match the individual security requirements. This section discusses the most relevant security requirements within OWS-6 and beyond, being relevant for securing OGC Web services. These requirements only address message exchange, which is relevant when defining Web service interfaces and protocols. There may be other requirements addressing physical or organizational protection as well as the protection of data, but this is out of scope for this document.

### 6.1 Confidentiality

Providing confidentiality means protecting messages against unauthorized reading. It has to be ensured that only the designated communication partners (typically the sender and the receiver of a message) can access the content of a message.

Confidentiality is provided by encryption, either on message level or on transport level. See section [7.2.1](#) for further details.

## 6.2 Authenticity

Authenticity provides evidence for the actual origin of the communication message with the authenticated party. A variety of methods and standards that may be used to provide authentication for OGC web services as shown below:

- HTTP Authentication
- Session Management / Cookies
- SAML
- Shibboleth
- OpenID
- WS-Security

For authentication a variety of different mechanisms exist. Typically, authenticity can be guaranteed on message level by applying digital signatures to messages. These signatures are validated against the public key of the sender, and thus valid signatures can only be generated by the owner of the corresponding private key. Therefore, if a signature can be validated against a certain public key, the owner of this key pair has to be the originator of this signature.

On transport level, authenticity can be provided by requiring an adequate certificate during the handshake of the secure connection, which is derived by a trusted root certificate. Once this connection is established, authenticity is provided for all messages being submitted by the communication partner who provided this certificate (sender and/or receiver).

In OWS-6, development and demonstrations were focused on use of WS-Security, SAML and OpenID.

## 6.3 Integrity

Integrity protects messages against unnoticed modifications. Typically, integrity is provided by the use of digital signatures. These signatures are tightly bound to the message to be protected. Whenever there was a modification of this message after the signature was applied, a validation of this signature will fail.

If security on transport level is provided, integrity is ensured once the secure communication session is established.

## 6.4 Non-Repudiation

Non-repudiation provides evidence for the existence of this message. Non-repudiation is ensured by storing messages together with a valid signature of the sender. If a sender denies having submitted a certain message afterwards, the receiver can expose the signed message. Since nobody but the sender would be able to generate this signature, this stored message proves that the message was signed by the holder of the public key corresponding to this signature.

## 6.5 Access Control

Access control is not directly related to message exchange, but nevertheless it is one of the key requirements within OWS-6. Access control aims at authorizing service requests. For access controlled services incoming requests are matched against policies which define access rights to certain resources for certain subjects (requestors). If these access rights cover the requested action, access is permitted, otherwise access will be denied.

## 6.6 Audit/Auditing

A family of security controls in the technical class dealing with ensuring activity involving access to and modification of sensitive or critical files is logged, monitored, and possible security violations investigated. [27].

# 7 Cryptography

Cryptography is a key technology to meet the security requirements described in section 6. This section briefly discusses cryptographic methods used for encrypting and signing messages.

## 7.1 Symmetric Cryptography

Symmetric cryptography is sometimes called "secret-key cryptography" (versus public-key cryptography) because the entities that share the key, such as the originator and the recipient of a message, need to keep the key secret. Keeping the shared key secret entails both cost and risk when the key is distributed. Thus, symmetric cryptography has a key management disadvantage compared to asymmetric cryptography<sup>1</sup>.

Since asymmetric cryptographic algorithms for encryption tend to be computationally intensive, WS-SecureConversation specification was developed. This specification allows use of faster symmetric algorithms for message-level security, making it well-suited for OGC web services and clients where large volumes of messages may be exchanged between many clients and a few data or service providers.<sup>2</sup>

## 7.2 Asymmetric Cryptography

Asymmetric cryptography (also known as Public Key Cryptography) uses a pair of keys for any participant of a secure communication. One key has to be kept secret (private key), while the other key is public (public key) and has to be shared with the communication partners.

---

<sup>1</sup> IETF RFC 2828, <http://www.ietf.org/rfc/rfc2828.txt>

<sup>2</sup> NIST Guide to Secure Web Services, SP800-95, August 2007



The basic principle of asymmetric cryptography is that cipher text encrypted with one key of this pair can only be decrypted with the other one. If a message is encrypted with a public key it can only be decrypted with the corresponding private key and vice versa.

This characteristic can be used for encrypting and signing messages, as shown in the following examples.

Typically, asymmetric cryptography is used for the initialization of a communication and for exchange of a session key, which then is used to continue the communication via symmetric cryptography due to its better performance.

### **7.2.1 Encryption**

User A wants to send an encrypted message to user B. User A encrypts the message with B's public key using asymmetric cryptography, so only B is able to decrypt this message with his own private key.

Since asymmetric cryptography needs far more computation than symmetric encryption, both technologies are typically used in combination. First, the initiator of the communication creates a session key which is asymmetrically encrypted and sent to the communication partner. Now, that a session key is securely exchanged, it can be used together with symmetric encryption for the following communication.

### **7.2.2 Digital Signatures**

Digital signatures rely on asymmetric cryptography, which requires a pair of cryptographic keys for each participant involved in secure communication. In contrast to section 7.2.1, sending a signed message requires asymmetric encryption of the message with the sender's private key. That allows anybody to decrypt the message with the sender's public key and thus proves that nobody else could have encrypted this message but the owner of the corresponding private key.

For performance improvements, instead of encrypting the whole message, typically a hash value is computed out of the message and then this hash value is signed while the message itself is sent in clear text.

Of course, encryption and signatures can also be used in combination, by first computing a signature with the sender's private key, and then encrypting the signed message with the recipient's public key.

## **8 Trust**

### **8.1 Introduction**

Web services standards are inherently flexible and have allowed several architecture models to evolve: a brokered trust model, a pair-wise trust model, a federated trust model, and a perimeter defense model. While these models use the term trust, they are

limited to being able to trust the identity of the service. Being able to establish a Web service's identity does not mean that the service itself is inherently trustworthy. There is always the possibility that a Web service has entered an erroneous state or has been compromised.

When trust relationships span multiple organizations, the requirements for individual Web services will vary. Even if a provider's identity is trusted, does not mean that its content may not contain malicious or erroneous content. Likewise, even if a requester is represented by a trusted identity, does not ensure that its requests may not contain or be replaced by similarly malicious or erroneous content. Nevertheless, identifying and authenticating Web services is an essential step in establishing trust. Each trust model provides different benefits and drawbacks, allowing trust to be supported in a wide variety of environments.

The Trust Model used in OWS-6 is based on the WS-Trust specification. This model is based on a process in which a Web service can require that an incoming message prove a set of claims (e.g., name, key, permission, capability, etc.). If a message arrives without having the required proof of claims, the service **SHOULD** ignore or reject the message. A service can indicate its required claims and related information in its policy as described by [WS-Policy] and [WS-PolicyAttachment] specifications.

## **8.2 Public Key Infrastructure (PKI) Concept**

The concept of asymmetric cryptography (public key cryptography) was already introduced in section 7.2. It allows arbitrary communication partners to exchange encrypted and/or signed messages, as long as the public keys of the communication partners are known.

This implies the need for a mechanism to securely exchange public keys between communication partners and to verify the validity of a received public key. Since the knowledge about these keys is a prerequisite to communicate securely, a secure way for the key exchange is needed. Sending a public key by email or downloading it from a web page via a potentially insecure communication channel such as the Internet does not fulfill these requirements.

Public Key Infrastructures (PKIs) solve this problem by introducing Certificate Authorities (CAs). These CAs are assumed to be trustworthy either per se or within a certain security domain. CAs issue digital certificates for owners of asymmetric key pairs, asserting that the actual public key belongs to a certain owner. Such a certificate is signed by the CA to ensure authenticity and integrity (see sections 6.2 and 6.3).

Once a communication partner trusts a CA, it is assumed that he also trusts all certificates being issued by this CA. So for exchanging public keys, communication partners may now exchange their certificates including those keys and the according identity information provided by the CA. If there is a valid signature of the CA on this certificate, the communication partners can be sure to have received the right public keys.

Of course, PKIs still need the public keys of the CAs to be known, but this affects only a very limited number of public keys (those of the CAs) instead of all keys of all communication partners, and so called root certificates, including the identity and the public key, of the most popular CAs are already included in many software components such as operating systems and browsers, so a user does not even have to care about how to get those certificates.

### 8.3 Trust Establishment

Trust is always needed for secure communication. Depending on the already existing trust relationships of the communication partners, it can be distinguished between different kinds of trust establishment.

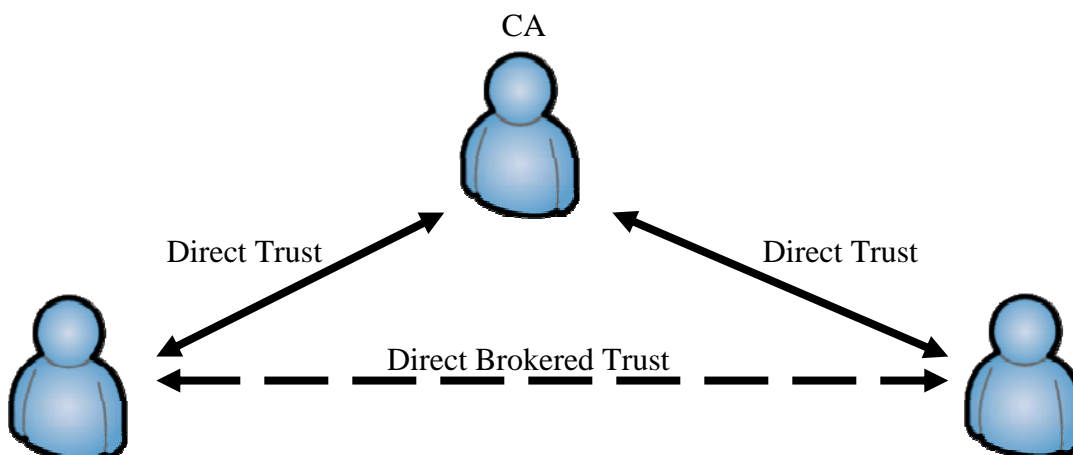
#### 8.3.1 Direct Trust

Communication partners have a direct trust relationship, if their public keys are already exchanged and there is no need for exchanging any additional information as a prerequisite for secure communication. Thus, no CA is needed for establishing trust.



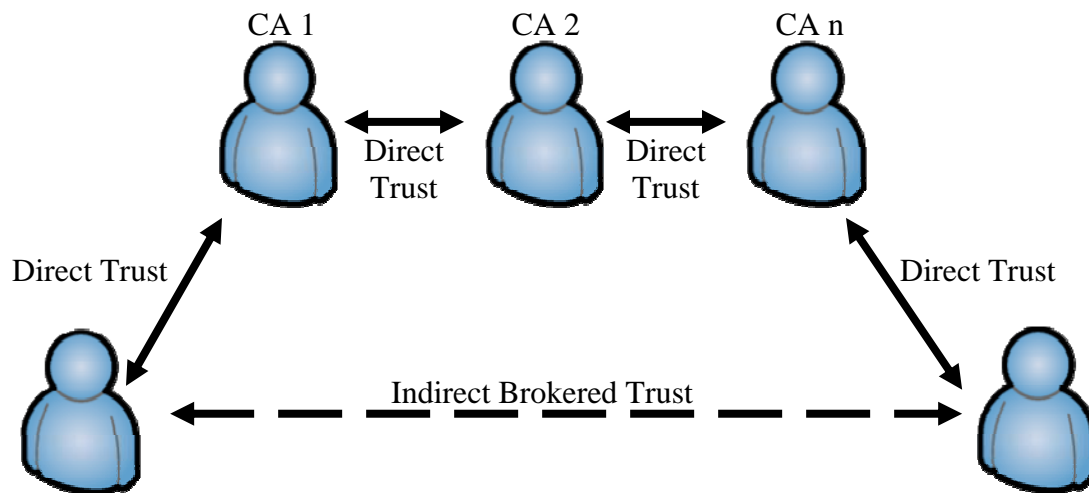
#### 8.3.2 Direct Brokered Trust

If two communication partners have no pre-established trust relationship, trust can be established by brokering. Trust brokers in PKIs are typically CAs, where a commonly accepted CA vouches for the identity of both communication partners.



### 8.3.3 Indirect Brokered Trust

Indirect brokered trust is an extension of direct brokered trust, where is no single CA which has a trust relationship to both communication partners. Thus, trust has to be brokered between several CAs, resulting in a trust chain between the communication partners.



## 9 Abstract Security Architecture

A security architecture is never self-contained, rather it being an overlay to a domain architecture. OGC defines service architecture, is based on the OWS Common standard [1] and extended by other implementation standards.

This section discusses the security approach actually taken within the OWS-6 testbed.

### 9.1 Security Extensions to Existing OGC Architecture

In the OGC, a security architecture should leverage the existing OGC Standards by defining generic security extensions being applicable to all OGC standards based on OWS Common [1]. On an abstract level, such extensions should be independent of specific technology bindings, leading to a common abstract security architecture for OGC web services.

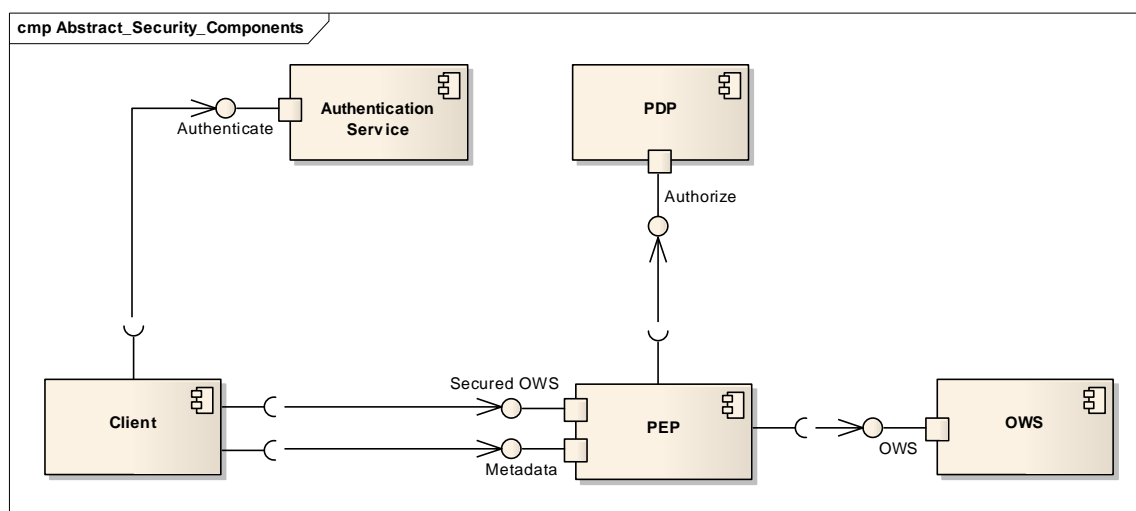
For the testbed approach to access control using the OGC service model, the following extensions are needed:

- **Security metadata**  
Services should be able to expose their security requirements to requesting clients as part of the technical service metadata. This may include

- requiring certain security tokens (such as identity tokens or license tokens)
  - requiring the use of transport level security
  - requiring the use of message level encryption
  - requiring requests to be digitally signed
- **Protocol extensions**  
To submit security-related information such as security tokens, extensions to the OGC service protocols should be defined, allowing such information to be submitted together with request and/or response messages.
  - **Additional error definitions**  
If a client fails to fulfill a service's security requirements or if access is denied to a certain request, the service may respond with an appropriate error message, indicating the reason for the failure of the request. Although error messages need to be specified, their use should be optional, since error messages may lead to undesired information leakage (e.g. a message indicating that access to a certain resource is denied nevertheless indicates that this resource actually exists).

## 9.2 Static View: Security Interactions

The security components used to provide access control in OWS-6 are shown in Figure 1. This architecture relies on the access control architecture defined by OASIS in the 'eXtensible Access Control Markup Language' (XACML) standard [10].



**Figure 1, Abstract Security Components**

The OWS to be secured is protected by a Policy Enforcement Point (PEP), which is responsible for receiving requests, analyzing them and delegate access decisions to the Policy Decision Point (PDP).

The PDP receives an authorization request and matches it against the available policies. The decision is enforced by the PEP. In case of a 'Permit' decision the request is

forwarded to the OWS. A 'Permit' decision can include obligations which have to be fulfilled by the PEP. Whenever the PEP is unable to fulfill an obligation it has to deny access to the OWS, as it is required by a 'Deny' decision from the PDP.

A PDP may also respond with 'Indeterminate' or 'NotApplicable', both of them also resulting in a denial of access due to the denial biased characteristics of the PEPs used in this testbed. .

### 9.3 Dynamic View: Security Workflows

Requesting a secured service typically requires one to get information about the service's security requirements. Usually a service would at least require an authentication in order to apply policies to a certain request. Figure 2 shows an initial request for the service's preconditions. This step can be omitted if the client already knows the service's security requirements, or if the information was already received from a different source, e.g. a catalog.

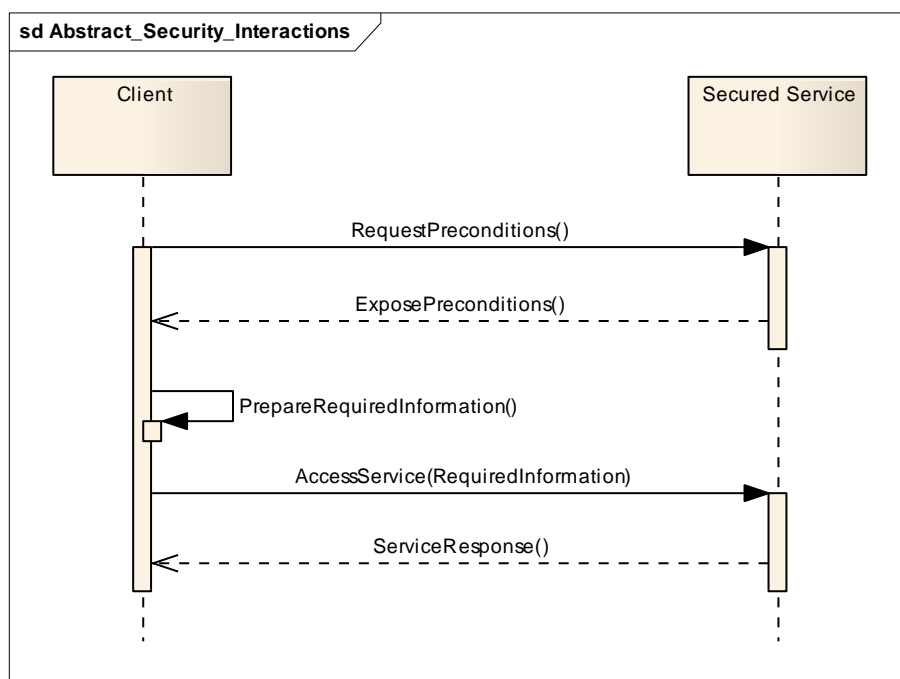
Alternatively, the client could also try to access the service without any special preparation and try to derive the security requirements from the error message. This strategy, however, may not lead to the desired result, since an error message not always indicates the real reason for the exception (see discussion in section 15.2), or the service handles a request without any security information as an anonymous request without producing an exception. In this case, the client would not be informed that a potentially richer result would be available for an authenticated request.

In this testbed, the complexity of issues surrounding the exchange of security metadata for different types of services was discovered, without being able to resolve these issues broadly. So further work on this problem seems to be necessary.

If the client is informed about the service's preconditions it can try to prepare the required information (e. g. an authentication), and submit this information together with the request.

Figure 2 does not explicitly indicate a PEP (nor a PDP or PAP), since from the client perspective there is only one service endpoint, which may be a separate PEP service, or a security-enabled service with built-in policy decision and enforcement capabilities. In addition, this diagram also abstractly represents all pre- and post-processing that may be performed by the PEP service. Such actions might include:

- Preparing an XACML authorization decision request
- Processing of the XACML authorization decision response
- Preparing a native service request based on the authorization decision response
- Processing of the native service response
- Forwarding of final result to the requestor



**Figure 2, Abstract Security Interactions**

## 10 SOAP Binding

For OGC services, SOAP is one possible binding, which must be specified for every new service specification. Moreover, SOAP is the basis for most SOA infrastructures in the mainstream IT world. OWS-6 testbed placed a strong focus on security for SOAP-based services.

### 10.1 Relevant Standards

For SOAP there are several existing security-related standards which can be applied to OWS infrastructures.

The starting point when binding to a secured service is to request the service's security requirements. These security-related metadata can be described with WS-SecurityPolicy [18] as part of a WS-Policy [16] description.

A standardized way to access the WS-Policy description of a service is defined by WS-MetadataExchange [15]. WS-MetadataExchange makes typical service-related metadata accessible, besides WS-Policy also including WSDL.

The security metadata may define requirements for communication with a secured service, such as the requirement for encrypted communication (either on message level or on transport layer level), required signatures on messages or message parts, and required security information.

An example of required security information would be authentication information. WS-SecurityPolicy is able to require certain token formats and can also refer to trusted issuers for those tokens by using WS-Addressing [14].

Security tokens can be issued by a Security Token Service (STS) defined by WS-Trust [19]. An STS is primarily designed to issue tokens. It can be used to either convert a token from a certain format into a different format, or to convert tokens from one security domain into tokens of another security domain.

Within OWS-6, an STS is used as authentication service. This STS provides Identity Assertions expressed in Security Assertion Markup Language (SAML) [12] which can be used to authenticate at a PEP. The STS also expresses its security requirements by WS-SecurityPolicy and thus can require a username token including credentials which can be used for authentication purposes.

Whenever communication has to be secured, WS-Security [17] can be applied to provide encryption on message level, using XML Encryption [20], and to provide signatures on messages or message parts following XML Signature [21]. WS-Security furthermore defines several profiles describing how to attach security tokens to SOAP messages in order to transmit them along with a request to a service.

## **10.2 Protocol**

SOAP messages consist of a header and a body element. The body is used as a container for all functional payloads, so a SOAP request contains all request information in the body, and a response carries the results in the body.

The header can be used for orthogonal information which has no directly functional relevance. Thus, the WS-Security standard [17] defines the header to be used to include security tokens such as identity information or signatures within the SOAP header.

This separation of concerns allows defining security completely independently from the business protocol, which makes it generically applicable to any SOAP service.

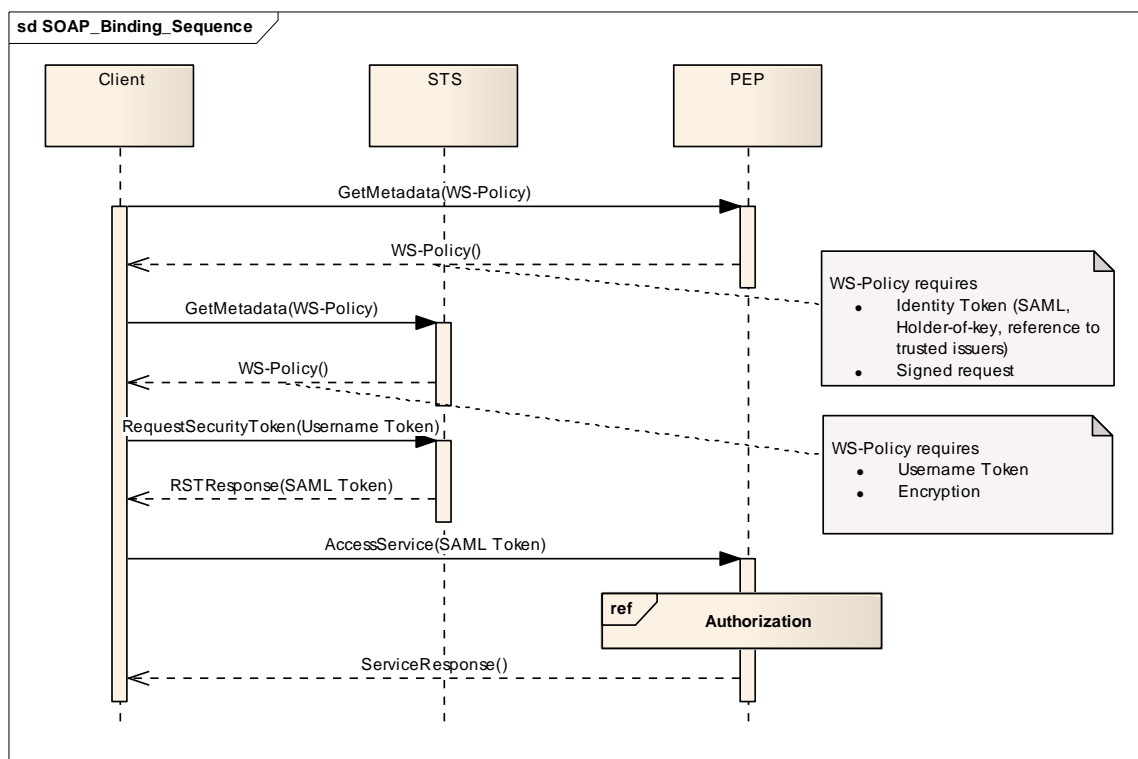
## **10.3 Interactions**

### **10.3.1 Binding to a Secured Service**

To access a service, a client must perform several steps to bind to this service. Especially for secured services several security-related requirements may have to be fulfilled.

The communication necessary for binding to a secured service in the actual testbed approach is given in Figure 3.





**Figure 3, Binding to a Secured Service**

When accessing a SOAP service, a client would usually first evaluate the WSDL description of this service (omitted in the sequence diagram). Additional to those interface description, service-specific requirements can be expressed by WS-Policy. So, additional to the WSDL description, especially for secured services a client would request the WS-Policy description as well, using the GetMetadata operation defined by WS-MetadataExchange. An example for a GetMetadata request, requesting a WS-Policy description of a service, is given in the following SOAP request:

```

<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <mex:GetMetadata
      xmlns:mex="http://schemas.xmlsoap.org/ws/2004/09/mex">
      <mex:Dialect>http://schemas.xmlsoap.org/ws/2004/09/policy</mex:Dialect>
    </mex:GetMetadata>
  </soapenv:Body>
</soapenv:Envelope>
  
```

**Figure 4, Example GetMetadata Request**

A WS-MetadataExchange response defining an identity precondition and requiring a SAML assertion from a referenced token issuer is given in Figure 5 below.

```

<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <mex:Metadata
      xmlns:mex="http://schemas.xmlsoap.org/ws/2004/09/mex"
      <mex:MetadataSection
        Dialect="http://schemas.xmlsoap.org/ws/2004/09/policy">
          <wsp:Policy
            xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"

            xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
            xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd"
            xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
              <wsp:ExactlyOne>
                <wsp:All>
                  <wsp:Policy wsu:Id="IdentityPrecondition">
                    <wsse:RelatedService
                      wsse:ServiceType="wsse:ServiceIP">
                        <wsa:EndpointReference>
                          <wsa:Address>http://v-ebiz.uni-
muenster.de/axis2/services/STS</wsa:Address>
                        </wsa:EndpointReference>
                      </wsse:RelatedService>
                      <wsse:SecurityToken wsp:Usage="wsp:Required">

<wsse:TokenType>SAMLAssertion</wsse:TokenType>
                    </wsse:SecurityToken>
                  </wsp:Policy>
                </wsp:All>
              </wsp:ExactlyOne>
            </wsp:Policy>
          </mex:MetadataSection>
        </mex:Metadata>
      </soapenv:Body>
    </soapenv:Envelope>

```

**Figure 5, Example WS-MetadataExchange Response**

Such a WS-Policy document may include a WS-SecurityPolicy element, describing requirements for security tokens and referring to one or more trusted issuers of those tokens.

In order to acquire such a token, the client would also request the WS-Policy description for the token issuing service to explore the requirements. The token issuing service, for instance a WS-Trust STS, could offer the WS-Policy description shown in Figure 6 which follows.

```

<wsp:Policy wsu:Id="SigOnly" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
  <wsp:ExactlyOne>
    <wsp:All>

```

```

        <sp:AsymmetricBinding
xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
    <wsp:Policy>
        <sp:InitiatorToken>
            <wsp:Policy>
                <sp:X509Token
sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/I
ncludeToken/AlwaysToRecipient">
                    <wsp:Policy>
                        <sp:RequireThumbprintReference/>
                        <sp:WssX509V3Token10/>
                    </wsp:Policy>
                </sp:X509Token>
            </wsp:Policy>
        </sp:InitiatorToken>
        <sp:RecipientToken>
            <wsp:Policy>
                <sp:X509Token
sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/I
ncludeToken/Never">
                    <wsp:Policy>
                        <sp:RequireThumbprintReference/>
                        <sp:WssX509V3Token10/>
                    </wsp:Policy>
                </sp:X509Token>
            </wsp:Policy>
        </sp:RecipientToken>
        <sp:AlgorithmSuite>
            <wsp:Policy>
                <sp:TripleDesRsa15/>
            </wsp:Policy>
        </sp:AlgorithmSuite>
        <sp:Layout>
            <wsp:Policy>
                <sp:Strict/>
            </wsp:Policy>
        </sp:Layout>
        <sp:IncludeTimestamp/>
        <sp:OnlySignEntireHeadersAndBody/>
    </wsp:Policy>
</sp:AsymmetricBinding>
<sp:Wss10
xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
    <wsp:Policy>
        <sp:MustSupportRefKeyIdentifier/>
        <sp:MustSupportRefIssuerSerial/>
    </wsp:Policy>
</sp:Wss10>
<sp:SignedParts
xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
    <sp:Body/>
</sp:SignedParts>
</wsp:All>
</wsp:ExactlyOne>

```

```
</wsp:Policy>
```

### Figure 6, Example WS-Policy description

This WS-Policy description requires an X.509 token within a signed request to the STS for authentication. An according STS 'RequestSecurityToken' request is given below:

```
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-
envelope">
  <soapenv:Header
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
    <wsse:Security xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
soapenv:mustUnderstand="true">
      <wsu:Timestamp xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
wsu:Id="Timestamp-19235919">
        <wsu:Created>2008-12-19T09:01:06.515Z</wsu:Created>
        <wsu:Expires>2008-12-19T09:06:06.515Z</wsu:Expires>
      </wsu:Timestamp>
      <wsse:BinarySecurityToken xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soap-message-security-1.0#Base64Binary" ValueType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v1"
wsu:Id="CertId-148082">
MIICTDCCAbUCBebJZMQwDQYJKoZIhvcNAQEEBQAwbDELMakGA1UEBhMCTEsxEDA0BgNVBAg
TBldlc3Rlcm4xEDA0BgNVBAcTB0NvbG9tYm8xDzANBgNVBAoTBkFwYWN0ZTEQMA4GA1UECx
MHUmFtcGFydDEWMBQGA1UEAxMNU2FtcGxliENsaWVudDAGFw0wNzA4MjAwOTU0MTJaGA8yM
DYyMDUyMzA5NTQxMlowbDELMakGA1UEBhMCTEsxEDA0BgNVBAgTBldlc3Rlcm4xEDA0BgNV
BAcTB0NvbG9tYm8xDzANBgNVBAoTBkFwYWN0ZTEQMA4GA1UECxMHUmFtcGFydDEWMBQGA1U
EAXMNU2FtcGxliENsaWVudDCBnzANBgkqhkiG9w0BAQEFAAOBjQAwGyKCGYEAhjQp2NJRUr
AEsPYIlg26m34016E6WkyBWMbkSvy/FJQoNg2HSotqF/DHmej7qqJCDtiHtdZqCTOo28cpy
B3XJ0g6y23ADTy1v7qUjYief4Bn3p9QFtyznUmKyZ6hK4CjGraYvcDgJrlnPkfeyVnNamkz
JB7TVRaLkumRlxHgxm0CAWEAATANBgkqhkiG9w0BAQQFAAOBgQBnLSbNEaGBj8GB0XWBndY
3JFvblPvI2mDbtZsNiggGOCezyAufGe6RnR3s5DjR5YQqPcMiDtlSkFQm4/SRN2Yh16E6l7
LfsOhGQsPiPrDrci4Tl8pzleDLsrtJiiBah1NdeISaD0kpoUiaNKiQiu16JCnxc8tGSW3nS
Pg44aLYmA==
      </wsse:BinarySecurityToken>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
Id="Signature-23930419">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
          <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
          <ds:Reference URI="#Id-16001744">
            <ds:Transforms>
              <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            </ds:Transforms>
            <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>
              uAN35WB6DXliPywz/a3RIwgJ10s=
            </ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
      </ds:Signature>
    </wsse:Security>
  </soapenv:Header>
</soapenv:Envelope>
```

```

        </ds:Reference>
        <ds:Reference URI="#Timestamp-19235919">
            <ds:Transforms>
                <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            </ds:Transforms>
            <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>
                2BtHvHoCTjJjQfZ7FxtQE1wjdT8=
            </ds:DigestValue>
        </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>
ROCzlyDrAdSTDvH86sXKBxHfvPq0U70EW0mgdtYDHUoTSMaU2+/wxwbEoxOChRjIhcPa7a5
xgsNRb4HdAxQmaiZyfxEO8alH8N4pTR4tEcEUnq4V0hkTt/kDz9CuHYFQyJcCMBghNax0B4
2S4Yc54LAB8+37cH1sEC3C6FsV7/Q=
    </ds:SignatureValue>
    <ds:KeyInfo Id="KeyId-22316618">
        <wsse:SecurityTokenReference
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd" wsu:Id="STRId-10175206">
            <wsse:Reference URI="#CertId-148082"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
x509-token-profile-1.0#X509v1"/>
        </wsse:SecurityTokenReference>
    </ds:KeyInfo>
</ds:Signature>
</wsse:Security>
<wsa:To>http://v-ebiz.uni-
muenster.de:8085/axis2/services/STS</wsa:To>
    <wsa:ReplyTo>

        <wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/a
nonymous</wsa:Address>
    </wsa:ReplyTo>

    <wsa:MessageID>urn:uuid:0DC5AD7C462BD0A5951229677264155</wsa:Message
ID>

    <wsa:Action>http://schemas.xmlsoap.org/ws/2005/02/trust/RST/Issue</w
sa:Action>
</soapenv:Header>
<soapenv:Body xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
wsu:Id="Id-16001744">
    <wst:RequestSecurityToken
xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">

        <wst:RequestType>http://schemas.xmlsoap.org/ws/2005/02/trust/Issue</
wst:RequestType>
        <wst:Lifetime>
            <wsu:Created>2008-12-19T09:01:03.625Z</wsu:Created>
            <wsu:Expires>2008-12-19T09:06:03.625Z</wsu:Expires>
        </wst:Lifetime>
        <wst:TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-
token-profile-1.1#SAMLV1.1

```

```

    </wst:TokenType>

    <wst:KeyType>http://schemas.xmlsoap.org/ws/2005/02/trust/SymmetricKey</wst:KeyType>
      <wst:KeySize>256</wst:KeySize>
      <wst:Entropy>
        <wst:BinarySecret
Type="http://schemas.xmlsoap.org/ws/2005/02/trust/Nonce">+S2DJBnOZ9U0OY
etA4cJ+rrw6kOBQmek</wst:BinarySecret>
        </wst:Entropy>

      <wst:ComputedKeyAlgorithm>http://schemas.xmlsoap.org/ws/2005/02/trust/CK/PSHA1</wst:ComputedKeyAlgorithm>
    </wst:RequestSecurityToken>
  </soapenv:Body>
</soapenv:Envelope>

```

**Figure 7, Example RequestSecurityToken request**

This request includes the required X.509 token and a signature and requires in the request body the issuance of a SAML 1.1 token, as it was required by the PEP.

The response from the STS looks as follows in Figure 8.

```

<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope" xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
  <soapenv:Header
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing">
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
soapenv:mustUnderstand="true">
      <wsu:Timestamp xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
wsu:Id="Timestamp-25209015">
        <wsu:Created>2008-12-19T09:01:14.765Z</wsu:Created>
        <wsu:Expires>2008-12-19T09:06:14.765Z</wsu:Expires>
      </wsu:Timestamp>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
Id="Signature-10194937">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
          <ds:Reference URI="#Id-29664748">
            <ds:Transforms>
              <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
            </ds:Transforms>
            <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
            <ds:DigestValue>3B19If0fGSJiH//CJSE2/zJVhCA=</ds:DigestValue>
          </ds:Reference>
          <ds:Reference URI="#Timestamp-25209015">
            <ds:Transforms>

```

```

        <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        </ds:Transforms>
        <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

        <ds:DigestValue>s6bGuBBUGjElgp4zk0cprZyWWgI=</ds:DigestValue>
        </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>
a7J1l4+ES/Jqh54kvzM+nN4ui6GXYZb+zVeM9JLv9qjVPhYpj6ACFy/6vLdbSKJwlklmqfL
qjZ3NpM7DMSytr8IHhGFMoLdsTenotxVS5nB6n//p7NkWSb8QDzu01kt5hMS99Cip5bToqS
2F3rPbL2162neio0FjjWoUosIlvuo=
    </ds:SignatureValue>
    <ds:KeyInfo Id="KeyId-17388264">
        <wsse:SecurityTokenReference
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd" wsu:Id="STRId-19008310">
            <wsse:KeyIdentifier
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soap-message-security-1.0#Base64Binary" ValueType="http://docs.oasis-
open.org/wss/oasis-wss-soap-message-security-
1.1#ThumbprintSHA1">HYL371NzoOs2+IA24VDkBGcUFQM=</wsse:KeyIdentifier>
        </wsse:SecurityTokenReference>
    </ds:KeyInfo>
    </ds:Signature>
</wsse:Security>

    <wsa:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:To>

    <wsa:MessageID>urn:uuid:A0DE17F9E6BE9DCC811229677272767</wsa:Message
ID>

    <wsa:Action>http://schemas.xmlsoap.org/ws/2005/02/trust/RSTR/Issue</
wsa:Action>

    <wsa:RelatesTo>urn:uuid:0DC5AD7C462BD0A5951229677264155</wsa:Relates
To>
</soapenv:Header>
<soapenv:Body xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
wsu:Id="Id-29664748">
    <wst:RequestSecurityTokenResponse
xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
        <wst:TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-
token-profile-1.1#SAMLV1.1</wst:TokenType>
        <wst:KeySize>256</wst:KeySize>
        <wst:RequestedAttachedReference>
            <wsse:SecurityTokenReference
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">
                <wsse:Reference
URI="#_3ff4d9e401cbf45f7b3f695f742e7bld" ValueType="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1"/>
            </wsse:SecurityTokenReference>
        </wst:RequestedAttachedReference>

```

```

    <wst:RequestedUnattachedReference>
      <wsse:SecurityTokenReference
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">
        <wsse:Reference URI="_3ff4d9e401cbf45f7b3f695f742e7b1d"
ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV1.1"/>
      </wsse:SecurityTokenReference>
    </wst:RequestedUnattachedReference>
    <wst:Lifetime>
      <wsu:Created>2008-12-19T09:01:12.890Z</wsu:Created>
      <wsu:Expires>2008-12-19T09:06:12.890Z</wsu:Expires>
    </wst:Lifetime>
    <wst:RequestedSecurityToken>
      <Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instan
f6d
ce" xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
AssertionID="_3ff4d9e401cbf45f7b3f695f742e7b1d" IssueInstant="2008-12-
19T09:01:14.578Z" Issuer="SAMPLE_STS" MajorVersion="1"
MinorVersion="1">
        <Conditions NotBefore="2008-12-19T09:01:12.890Z"
NotOnOrAfter="2008-12-19T09:06:12.890Z"/>
        <AttributeStatement>
          <Subject>
            <SubjectConfirmation>
              <ConfirmationMethod>
                urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
              </ConfirmationMethod>
              <KeyInfo
xmlns="http://www.w3.org/2000/09/xmldsig#">
                <xenc:EncryptedKey
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="EncKeyId-
urn:uuid:2DF89B47628198AFE412296772740312">
                  <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
                  <ds:KeyInfo>
                    <wsse:SecurityTokenReference
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">
                      <wsse:KeyIdentifier
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soap-message-security-1.0#Base64Binary" ValueType="http://docs.oasis-
open.org/wss/oasis-wss-soap-message-security-
1.1#ThumbprintSHA1">HYL371NzoOs2+IA24VDkBGcUFQM=</wsse:KeyIdentifier>
                    </wsse:SecurityTokenReference>
                  </ds:KeyInfo>
                  <xenc:CipherData>
                    <xenc:CipherValue>
FvUwM3pVd9uVsDhKXzWoA5tsnvel8urKKMv/Pb10j9yvhoTxvnc6IFHkvCOCNjEh9kqL7om
6lcsYOmhf9Nz4EufRIB0TzG1WilPkSQdT+iZyFiPy6XiJpuSfcfa5cMDAgsilwcjHLedD
zJcRup414fmX1CMV4CtTlinBhVjog=
                    </xenc:CipherValue>
                  </xenc:CipherData>
                </xenc:EncryptedKey>
              </KeyInfo>
            </SubjectConfirmation>
          </Subject>
        </AttributeStatement>
      </Assertion>
    </wst:RequestedSecurityToken>
  </wst:SecurityToken>

```



```

        </KeyInfo>
    </SubjectConfirmation>
</Subject>
    <Attribute AttributeName="Name"
AttributeNamespace="https://rahas.apache.org/saml/attrns">
        <AttributeValue>Colombo/Rahas</AttributeValue>
    </Attribute>
</AttributeStatement>
    <ds:Signature
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
            <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
            <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
            <ds:Reference
URI="#_3ff4d9e401cbf45f7b3f695f742e7b1d">
                <ds:Transforms>
                    <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" />
                    <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                        <ec:InclusiveNamespaces
xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="code ds
kind rw saml samlp typens #default xsd xsi" />
                    </ds:Transform>
                </ds:Transforms>
                <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />

                <ds:DigestValue>kdUKVTgTUx/uTKjVly+grMKjPk0=</ds:DigestValue>
            </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>
IVnDbj7Pjmf94aamHJpTw6ivaeFyVJIJstf+zniByMAO7MRtNXQR4PhLtMP6PCi95obrGuK
ZK1htBcsJCfy2fiFUvvn3Dpo7yYeHzzrh56Yt5Dx0lJ5B2kCvx9Si6BQ5LKJ5eTffdMNRpn
cITQ3/+pMhNUe3+x3VBQEa0K4svbU=
        </ds:SignatureValue>
        <ds:KeyInfo>
            <ds:X509Data>
                <ds:X509Certificate>
MIICTjCCAbcCBEBjZQEwDQYJKoZIhvcNAQEEBQAwbTElMAkGA1UEBhMCTEsxEDA0BgNVBAg
TBldlc3Rlcm4xEDA0BgNVBAcTB0NvbG9tYm8xZDZANBgNVBAoTBkFwYWNoZTEZQMAGAlUEC
xMHUmFtcGFydDEXMBUGAlUEAxMOU2FtcGxllFNlcnZpY2UwIBcNMDCwODIwMDk1NTEzW
hgPMjA2MjA1MjMwOTU1MTNaMG0xCzAJBgNVBAYTAkxLMRAwDgYDVQQIEwdXZXN0ZXJ
uMRAwDgYDVQHEwdDb2xvbWJvMQ8wDQYDVQQKEwZBcGFjaGUxEDA0BgNVBAcTB1Jhb
XBhcnQxZzAVBGNVBAMTDlnhbXBsZSBTZXJ2aWNlMIGfMA0GCSqGSIb3DQEBAQUAA4GN
ADCBiQKBgQCdttgg6es2lUlyOD48/iiAlWobB0WwAQtfG4bb2KyvOE9dRF7+d/aZrHti3
QWs6dtHpGkVMLgpomoq7APEqlkQnRvduk2T6ln83JwlEpPDXH/emqEC9OdNqHZj3eoyf
34JMmgShuviYDqYaK4HkRmZMiJl3aPeZzPl60yBWydAuwIDAQABMA0GCSqGSIb3DQEB
BAUAA4GBACVcoAqNbjo7+Jbm6+3pyYagQoBpdHZLnR8EU9/CRKmUGTj5qjXqYtE+Eka
6OYKBzv/dHdYlB2X3yH3YlSx10tA3+5xl4VIjYODlgh9Bs9Tbqjl1tw0G37dLrlG97
kJAVjrkfm743N9EHKftFaX4iF1tWbGxa4+vIbBV4CaUG5s5x
                </ds:X509Certificate>
            </ds:X509Data>
        </ds:KeyInfo>
    </ds:Signature>

```

```

        </Assertion>
      </wst:RequestedSecurityToken>
    <wst:RequestedProofToken>
      <wst:ComputedKey>
        http://schemas.xmlsoap.org/ws/2005/02/trust/CK/PSHA1
      </wst:ComputedKey>
    </wst:RequestedProofToken>
    <wst:Entropy>
      <wst:BinarySecret
Type="http://schemas.xmlsoap.org/ws/2005/02/trust/Nonce">MiCaTjqkF/2os4
Sk3Y+hH8WPTFbNmtQlwPCxuhgcrXo=</wst:BinarySecret>
      </wst:Entropy>
    </wst:RequestSecurityTokenResponse>
  </soapenv:Body>
</soapenv:Envelope>

```

**Figure 8, Example RequestSecurityToken response**

The message body includes a SAML 1.1 holder-of-key assertion, confirming the identity of the requesting entity. The client can now use this holder-of-key assertion within a service request using a signed WS-Security message to the PEP as evidence for successful authentication at the STS. Such a message would look as follows in Figure 9.

```

<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
  <soapenv:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <wsse:Security xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
soapenv:mustUnderstand="1">
      <wsu:Timestamp xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
wsu:Id="Timestamp-1035988">
        <wsu:Created>2008-12-19T09:01:16.218Z</wsu:Created>
        <wsu:Expires>2008-12-19T09:06:16.218Z</wsu:Expires>
      </wsu:Timestamp>
      <xenc:EncryptedKey Id="EncKeyId-
urn:uuid:DCE9843997195896E712296772763752">
        <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
          <wsse:SecurityTokenReference>
            <wsse:KeyIdentifier
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soap-message-security-1.0#Base64Binary" ValueType="http://docs.oasis-
open.org/wss/oasis-wss-soap-message-security-
1.1#ThumbprintSHA1">HYL371NzoOs2+IA24VDkBGcUFQM=</wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>
TDSd/V9JTZlP6V9MtsEWwX1hqbQkLuSQYQA5BTpXzE0cnyRf1Y/mb6YXZjOARSpGz/vVh0y
AgqUsOSzeuR/eq7u9n2XeWl16Enjymp7zGULINuoExBymR3IX1dwQsvC2js5pFMTP27qRv8
MR3v9PYaq5u/0HOBRbM08HL/si6bE=
        </xenc:CipherValue>

```

```

        </xenc:CipherData>
    </xenc:EncryptedKey>
    <Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
AssertionID="_3ff4d9e401cbf45f7b3f695f742e7b1d" IssueInstant="2008-12-
19T09:01:14.578Z" Issuer="SAMPLE_STS" MajorVersion="1"
MinorVersion="1">
        <Conditions NotBefore="2008-12-19T09:01:12.890Z"
NotOnOrAfter="2008-12-19T09:06:12.890Z"/>
        <AttributeStatement>
            <Subject>
                <SubjectConfirmation>
                    <ConfirmationMethod>
                        urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
                    </ConfirmationMethod>
                    <KeyInfo
xmlns="http://www.w3.org/2000/09/xmldsig#">
                        <xenc:EncryptedKey
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="EncKeyId-
urn:uuid:2DF89B47628198AFE412296772740312">
                            <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
                            <ds:KeyInfo>
                                <wsse:SecurityTokenReference>
                                    <wsse:KeyIdentifier
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soap-message-security-1.0#Base64Binary" ValueType="http://docs.oasis-
open.org/wss/oasis-wss-soap-message-security-
1.1#ThumbprintSHA1">HYL371NzoOs2+IA24VDkBGcUFQM=</wsse:KeyIdentifier>
                                </wsse:SecurityTokenReference>
                            </ds:KeyInfo>
                            <xenc:CipherData>
                                <xenc:CipherValue>
FvUwM3pVd9uVsDhKXzWoA5tsnvel8urKKmV/Pb10j9yvhoTxvnc6IFHkvCOCNjEh9kqL7om
6lcsYOmhf9Nz4EufRIB0TzG1WilPkSQdT+iZyFiPy6XiJpuSfcfa5cMDAgsilwcjHLED
zJcRup414fmX1CMV4CtTlinBhVjog=
                                </xenc:CipherValue>
                            </xenc:CipherData>
                        </xenc:EncryptedKey>
                    </KeyInfo>
                </SubjectConfirmation>
            </Subject>
            <Attribute AttributeName="Name"
AttributeNamespace="https://rahas.apache.org/saml/attrns">
                <AttributeValue>Colombo/Rahas</AttributeValue>
            </Attribute>
        </AttributeStatement>
        <ds:Signature
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            <ds:SignedInfo>
                <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
                <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />

```

```

        <ds:Reference
URI="#_3ff4d9e401cbf45f7b3f695f742e7b1d">
        <ds:Transforms>
            <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
            <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                <ec:InclusiveNamespaces
xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="code ds
kind rw saml samlp typens #default xsd xsi"/>
            </ds:Transform>
        </ds:Transforms>
        <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <ds:DigestValue>
            kdUKVTgTUx/uTKjVly+grMKjPk0=
        </ds:DigestValue>
        </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>
IVnDbj7Pjmf94b61aamHJpTw6ivaeFyVJIJstf+zniByMAO7MRtNXQR4PhLtMP6PCi95obr
GuKZK1htBcsJCfy2fiFUvvnv3Dpo7yYeHzzrh56Yt5Dx0lJ5B2kCvx9Si6BQ5LKJ5eTffdMN
rpncITQ3/+pMhNUe3+x3VBQEa0K4svbU=
    </ds:SignatureValue>
    <ds:KeyInfo>
        <ds:X509Data>
            <ds:X509Certificate>
MIICTjCCAbcCBEBJZQEwDQYJKoZIhvcNAQEEBQAwbTElMAkGA1UEBhMCTEsxEDA0BgNVBAg
TB1dlc3Rlcm4xEDA0BgNVBAcTB0NvbG9tYm8xDzANBgNVBAoTBkFwYWN0ZTEQMA4GA1UECx
MHUmFtcGFydDEXMBUGA1UEAxMOU2FtcGx1IFNlcnZpY2UwIBcNMDCwODIwMDk1NTEzWhgPM
jA2MjA1MjMwOTU1MTNaMG0xCzAJBgNVBAYTAkxLMRAwDgYDVQQIEwdXZXN0ZXJuMRAwDgYD
VQQHEwdDb2xvbWJvMQ8wDQYDVQQKEWZBcGFjaGUxEDA0BgNVBAStB1JhbXBhcnQxZzAVBgN
VBAMTDlNhbxBSZSBTZXJ2aWNlMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCdtgg6es
s2lUlyOD48/iiAlW0bB0WwAQtfG4bb2KyvOE9dRF7+d/aZrHti3QWs6dtHpGkVMLgpomoq7
APEq1kQnRvduk2T6ln83Jw1EpPDxH/emqeC9OdNqHZj3eoyf34JMmgShuviYDqYaK4HkRmZ
MiJ13aPeZzPl60yBWydaUwIDAQABMA0GCSqGSIb3DQEBAQUAA4GBACVcoAqNbjo7+Jbm6+3
pyYagQoBpdHZLnR8EU9/CRKmUGTj5qjXqYtE+Eka6OYKBzv/dHdYlB2X3yH3YlSx10tA3+5
xl4VIjYODlgh9Bs9Tbqj1tw0G37dLrlG97kJAVjrkfm743N9EHKfFaX4iF1tWbGxa4+vIb
bV4CaUG5s5x
            </ds:X509Certificate>
        </ds:X509Data>
    </ds:KeyInfo>
</ds:Signature>
</Assertion>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
Id="Signature-32316171">
        <ds:SignedInfo>
            <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
            <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
            <ds:Reference URI="#Id-9434319">
                <ds:Transforms>
                    <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
                </ds:Transforms>
            </ds:Reference>
        </ds:SignedInfo>
    </ds:Signature>

```

```

        <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

        <ds:DigestValue>hgsp99eAlIFLPo+qAfMBlobf9Dc=</ds:DigestValue>
        </ds:Reference>
        <ds:Reference URI="#Timestamp-1035988">
        <ds:Transforms>
        <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
        </ds:Transforms>
        <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <ds:DigestValue>
        8IJKnEs0/3FQ7a6xII4o78NKPXI=
        </ds:DigestValue>
        </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>
    Z8tpKHrVuChUwqahBmcmjTV5TC4=
    </ds:SignatureValue>
    <ds:KeyInfo Id="KeyId-25857250">
    <wsse:SecurityTokenReference
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd" wsu:Id="STRId-19368402">
    <wsse:Reference URI="#EncKeyId-
urn:uuid:DCE9843997195896E712296772763752"
ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.0#EncryptedKey"/>
    </wsse:SecurityTokenReference>
    </ds:KeyInfo>
    </ds:Signature>
</wsse:Security>
    <wsa:To>http://v-ebiz.uni-muenster.de:8085/52n-security-
gatekeeper-webapp/services/Gatekeeper</wsa:To>

    <wsa:MessageID>urn:uuid:0DC5AD7C462BD0A5951229677276208</wsa:Message
ID>
    <wsa:Action>urn:method</wsa:Action>
</soapenv:Header>
    <soapenv:Body xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
wsu:Id="Id-30801804">
    <RequestProperty
xmlns="http://gatekeeper.service.security.n52.org"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ifgi.de/kvp2xml http://v-ebiz.uni-
muenster.de:8083/OWS-5/KVP2XML.xsd">
        <property xmlns="" name="SERVICE">WMS</property>
        <property xmlns="" name="REQUEST">GetCapabilities</property>
    </RequestProperty>
    </soapenv:Body>
</soapenv:Envelope>

```

**Figure 9, GetCapabilities SOAP request with SAML holder-of-key assertion and signature (Example)**

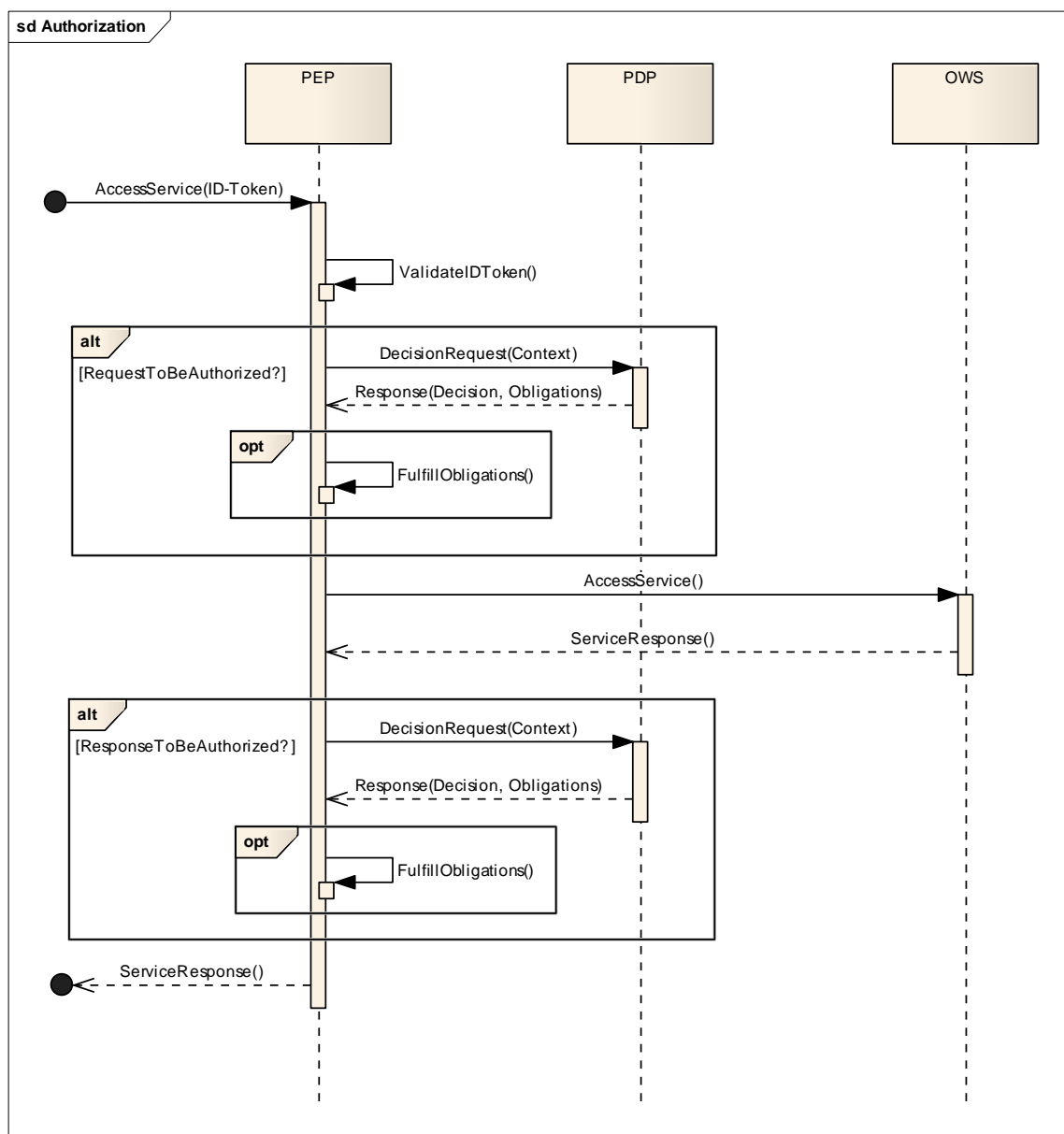
The example above shows a GetCapabilities SOAP request, including a SAML holder-of-key assertion and a signature in the WS-Security header block. By providing the signature, the message sender proves to be the holder of the key which is referenced in the SAML assertion and thus provides evidence for a successful authentication by the STS who issued this SAML token.

Now the required trust is established and the client can continue requesting the PEP, while the PEP can apply policies to the identity of the requestor.

### **10.3.2 Authorization**

Access requests can only be authorized if the identity of a requestor can be established.. Section 10.3.1 describes how to prove a requestor's identity.

The sequence diagram in Figure 10 shows the communication during the authorization process.



**Figure 10, Authorization Process**

The first step for an incoming request is the validation of the provided security tokens, especially verifying the signatures provided and ensuring that the signed elements are created by trusted parties.

Once this validation succeeded and the identity of the requestor is verified, the authorization can be performed. It seemed appropriate in this testbed to allow authorization on the request and/or on the response, although response authorization is not explicitly defined in the XACML standard. Thus, access control can be applied to requests and/or responses, which is defined by the logic of the PEP. Nevertheless, in both cases the authorization is the same. A decision request is submitted to the PDP, and depending on the response, access is permitted or denied. Since permit responses by the

PDP can optionally include obligations, it is the PEP's responsibility to fulfill those obligations. If the PEP fails in fulfilling them, access has to be denied.

When the service access is authorized, the service result can be returned to the requesting client.

## 11 RESTful Security for OGC Web Services

This section describes an entire alternative to the SOAP mechanisms discussed in the previous section, solving the same issues with a completely different approach.

The OWS-6 testbed investigated, implemented and tested the use of RESTful approaches to apply security mechanisms and technology in conjunction with the use of OGC web services. Figure 11 below shows the notional deployment of OGC web services and other capabilities for this work. The deployed environment was enabled by the cooperation and collaboration between OGC members, sponsors at NASA as well as the use of capabilities drawing on widely used authentication and authorization technologies from the open source and web community.

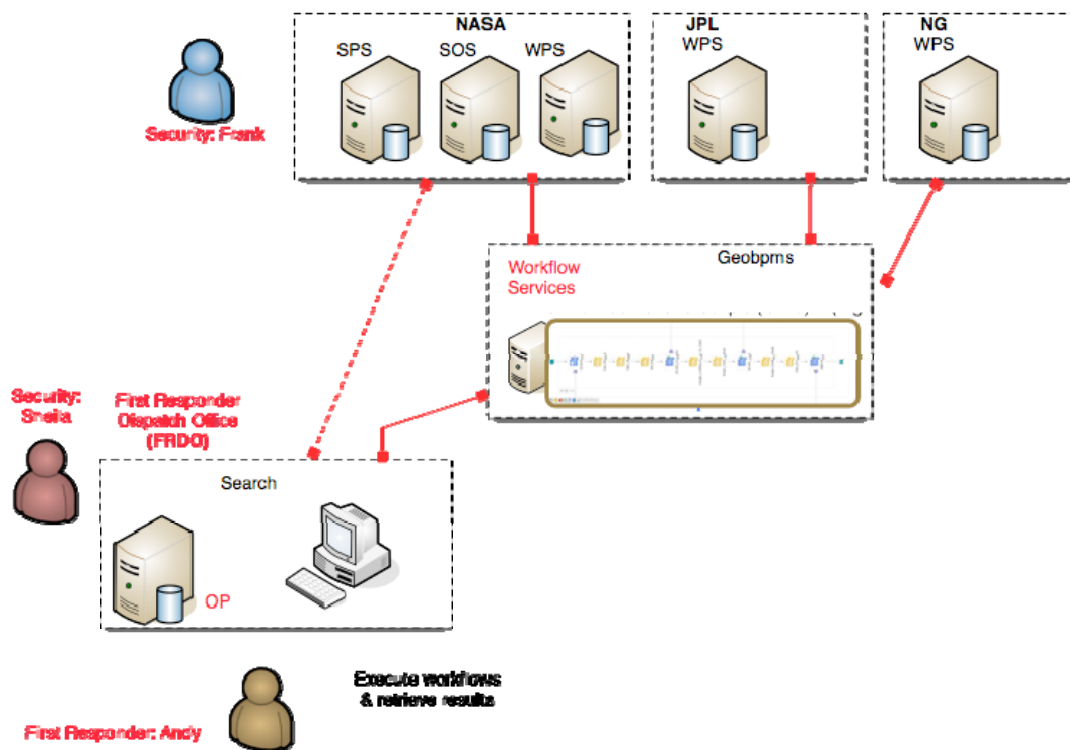


Figure 11, RESTful Workflow and Security Deployment



## 11.1 Bindings

Historically OGC web services have exercised HTTP-GET and –POST and recently SOAP bindings. During the last several years, interest has grown in OGC and elsewhere to use resource-oriented architectural style interfaces and encodings for implementation of OGC web services. In the OWS-6 testbed, a complimentary Airport Fire Scenario and demonstration was implemented using resource-oriented architectural styles for workflow and security in conjunction with use of OGC web services. The paragraphs which follow identify the standards, technologies and approaches used for these resource-oriented security protocols.

## 11.2 Relevant Standards & Specifications

The following standards were used as the basis for implementations provided in the test and demonstrations.

### OpenID Authentication 2.0

[http://openid.net/specs/openid-authentication-2\\_0.html](http://openid.net/specs/openid-authentication-2_0.html)

### Yadis Discovery Protocol 1.0

<http://yadis.org/papers/yadis-v1.0.pdf>

### OpenID Attribute Exchange 1.0

[http://openid.net/specs/openid-attribute-exchange-1\\_0.html](http://openid.net/specs/openid-attribute-exchange-1_0.html)

### OpenID Simple Registration Extension 1.0

[http://openid.net/specs/openid-simple-registration-extension-1\\_0.html](http://openid.net/specs/openid-simple-registration-extension-1_0.html)

### OAuth Core 1.0

<http://oauth.net/core/1.0/>

### XRI Resolution 2.0 (Committee Draft 03)

<http://docs.oasis-open.org/xri/2.0/specs/xri-resolution-V2.0.html>

## 11.3 Standards and Protocols

The following sections briefly describe the standards and protocols implemented to realize the RESTful security requirements for the Airport Fire Scenario.

### OpenID Authentication

OpenID is an open, decentralized, framework for user-centric digital identity. It takes advantage of already existing internet technology (URI, HTTP, SSL, Diffie-Hellman key exchange protocol) and realizes that users are already creating identities for blogs, photostreams, profile pages, etc. OpenID provides a protocol and service to easily transform existing URIs into an account which can be used to support OpenID logins.

In this testbed, OpenID was deployed using the open source Ruby library maintained by the JanRain project (<http://openidenabled.com/ruby-openid/>). This OpenID library provides the following capabilities:

- API for verifying OpenID identities (OpenID::Consumer)
- API for serving OpenID identities (OpenID::Server)
- Consumer and server support for extensions, including simple registration
- Yadis 1.0 and OpenID 1.0 service discovery, including server fallback
- Does not depend on underlying web framework
- Multiple storage implementations (Filesystem, SQL)
- Comprehensive test suite

Some experimental extensions to the current released libraries (v2.1.6) were implemented in this testbed to address security features required in the OWS-6 RFQ and Sponsors' requirements described in the Airport Fire Scenario. OpenID extensions implemented in this testbed included:

- Ability to dynamically add new domains when required to establish temporary trust between different security domains
- Ability to register web applications with an OpenID provider
- Extended OpenID provider user interface to manage user profile and authorization grants.

Source code for these extensions to the current released library version of OpenID is available upon request from [www.geobliki.com](http://www.geobliki.com). These extensions are being considered as proposed enhancements to the OpenID project by JanRain, Inc. who maintains the open source libraries and their distribution.

### **Open Authorization (OAuth)**

The OAuth protocol, defined by the OAuth Core specification, enables websites or applications (Consumers) to access Protected Resources from a web service (Service Provider) via an API, without requiring Users to disclose their Service Provider credentials to the Consumers. More generally, OAuth creates a freely-implementable and generic methodology for API authentication.

OAuth does not require a specific user interface or interaction pattern, nor does it specify how Service Providers authenticate Users, making the protocol ideally suited for cases where authentication credentials, such as with OpenID, are unavailable to the Consumer.

OAuth aims to unify the experience and implementation of delegated web service authentication into a single, community-driven protocol. OAuth builds on existing protocols and best practices that have been independently implemented by various websites. An open standard, supported by large and small providers alike, promotes a

consistent and trusted experience for both application developers and the users of those applications.

## **Yadis Discovery Protocol**

The Yadis specification defines a service discovery protocol allowing relying parties (aka identity consumers or member sites) to determine automatically, without end-user intervention, the most appropriate authentication protocol to use.

The Yadis specification provides:

- A general purpose identifier for persons and any other entities, which can be used in a variety of services.
- Syntax for a resource description document identifying services available using that identifier and an interpretation of the elements of that document.
- A protocol for obtaining that resource description document, given that identifier.

The purpose of the Yadis protocol is to enable a Relying Party to obtain a Yadis Resource Descriptor that describes the services available using a Yadis ID. A Yadis ID is an identifier used by one or more Yadis Services. A Yadis ID may be a URL; but it must be an identifier that is resolvable to a URL.

Together these enable coexistence and interoperation of a rich variety of services using a single identifier. The identifier uses a standard syntax and a well-established namespace; it requires no additional namespace administration infrastructure.

When a User offers a Yadis ID to a Relying Party, that Relying Party needs to discover which services are available using that Yadis ID. For example:

- Is it an OpenID URL, an XRI, a Lightweight Identity (LID) or a Simple Extensible Identity Protocol (SXIP) ID?
- What authentication methods are available?
- What other services?

To discover which services is available using a Yadis ID, the Relying Party Agent makes an HTTP request. This request may take any one of several forms, specified in the Yadis Specification.

In response to the request, the Relying Party Agent obtains either:

1. A Yadis document.
2. A URL that locates a Yadis document.

The Yadis document contains a Yadis Resource Descriptor, which identifies the services available using that Yadis ID, including services that can authenticate the User.

The Yadis document is based on a simple, extensible XML document called an Extensible Resource Descriptor (XRD). The format of XRD documents is being specified

by the XRI Technical Committee of OASIS (see the XRI Resolution 2.0 specification.) The XML schemas for the Yadis document are specified in the Yadis Specification.

The Yadis document contains a Yadis Resource Descriptor, which provides a list of identifiers of services. These are the services that know the User identified by the Yadis ID used to obtain the Yadis document. The Yadis Resource Descriptor also enables the User to specify which services it prefers be used.

### **OpenID Attribute Exchange (AX)**

Attribute Exchange is a service extension to OpenID that defines two message types for transferring attributes:

- fetch
- store

Fetch retrieves attribute information from an OpenID Provider, while store saves or updates attribute information on the OpenID Provider. Both messages originate from the Relying Party and are passed to the OpenID Provider via the user agent in accordance with the OpenID Authentication protocol specification.

An attribute is a unit of personal identity information that is identified by a unique URI. It may refer to any kind of information. A reference example of defining attribute types is provided by [[OpenID.axschema](#)] (Hardt, D., “Schema for OpenID Attribute Exchange,” May 2007.).

Attributes are defined in the AX schema for the following attribute types:

- |                 |                                      |
|-----------------|--------------------------------------|
| • Name          | • Instant Messaging                  |
| • Work          | • Web Sites                          |
| • Date of Birth | • Audio/Video Greetings              |
| • Telephone     | • Images                             |
| • Address       | • Other personal details/preferences |
| • Email         |                                      |

### **OpenID Simple Registration Extension**

OpenID Simple Registration is an extension to the OpenID Authentication protocol that allows for very light-weight profile exchange. It is designed to pass commonly requested pieces of information when an End User registers a new account with a web service. The OpenID schema defines attributes which includes the following for the Simple Registration extension:

Type URI	Label	SREG Property
<a href="http://axschema.org/namePerson/friendly">http://axschema.org/namePerson/friendly</a>	Alias/Username	openid.sreg.nickname
<a href="http://axschema.org/contact/email">http://axschema.org/contact/email</a>	Email	openid.sreg.email
<a href="http://axschema.org/namePerson">http://axschema.org/namePerson</a>	Full name	openid.sreg.fullname
<a href="http://axschema.org/birthDate">http://axschema.org/birthDate</a>	Birth date	openid.sreg.dob
<a href="http://axschema.org/person/gender">http://axschema.org/person/gender</a>	Gender	openid.sreg.gender
<a href="http://axschema.org/contact/postalCode/home">http://axschema.org/contact/postalCode/home</a>	Postal code	openid.sreg.postcode
<a href="http://axschema.org/contact/country/home">http://axschema.org/contact/country/home</a>	Country	openid.sreg.country
<a href="http://axschema.org/pref/language">http://axschema.org/pref/language</a>	Language	openid.sreg.language
<a href="http://axschema.org/pref/timezone">http://axschema.org/pref/timezone</a>	Time zone	openid.sreg.timezone

### eXtensible Resource Descriptor Sequence (XRDS)

XRDS is an XML format for discovery of metadata about a resource – in particular discovery of services associated with the resource, a process known as service discovery. For example, a website offering OpenID login can resolve a user's OpenID identifier to an XRDS document to discover the location of the user's OpenID service provider.

The XML format used by XRDS was originally developed in 2004 by the OASIS XRI (Extensible Resource Identifier) Technical Committee as the resolution format for XRIs. The acronym XRDS was coined during subsequent discussions between XRI TC members and OpenID developers at first Internet Identity Workshop held in Berkeley, CA in October 2005.

The protocol for discovering an XRDS document from a URL was formalized as the Yadis specification published by [Yadis.org](http://Yadis.org) in March 2006. Yadis became the service discovery format for OpenID 1.1.

A common discovery service for both URLs and XRIs proved very useful. In November 2007 the XRI Resolution 2.0 specification in OASIS formally added the URL-based method of XRDS discovery (Section 6). This format and discovery protocol subsequently became part of OpenID Authentication 2.0

### XRI Resolution 2.0

XRI Resolution provides a simple generic format for resource description (XRDS documents), a protocol for obtaining XRDS documents from HTTP(S) URIs, and generic and trusted protocols for resolving Extensible Resource Identifiers (XRIs) using XRDS documents and HTTP(S) URIs. These protocols are intended for use with both HTTP(S) URIs as defined in RFC2616 and with XRIs as defined by Extensible Resource Identifier (XRI) Syntax Version 2.0 or higher.

## 11.4 Interactions

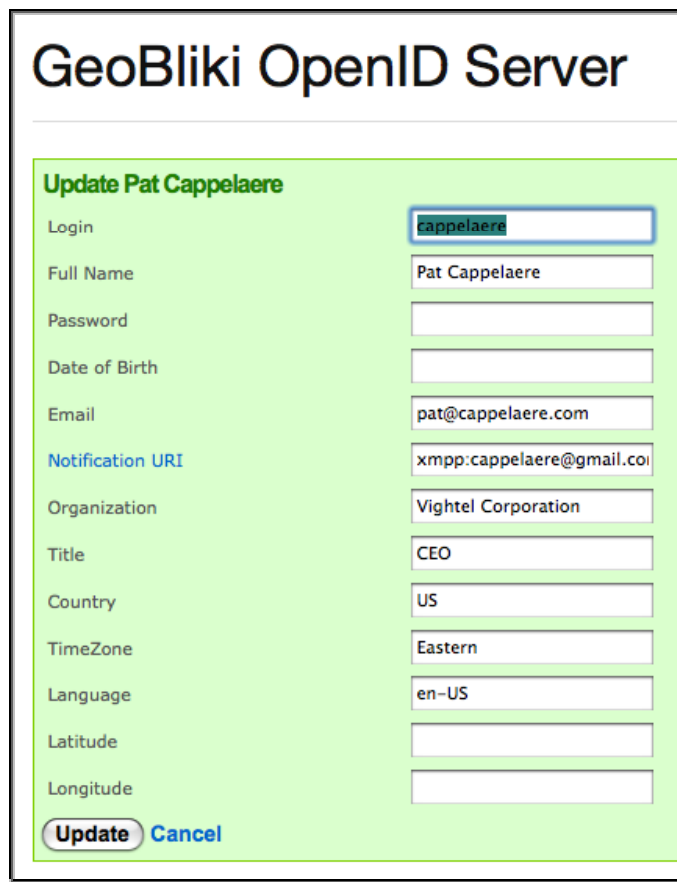
The interaction diagrams which follow depict the sequence of interactions that were exercised to test and demonstrate the use of resource-oriented security technologies described in scenario. The sequence of interactions is shown below:

1. Establish Trust (across security domains)
2. Register Application with OpenID
3. Delegate Authority and execute tasks
4. Revoke grants

Localized user identity management for this testbed scenario was enabled using an OpenID provider (OP). The OP manages user authentication (and authorization) when a user presents his OpenID to a trusted site. User profile information is obtained from the OP using an extension of the OpenID protocol called Attribute Exchange or AX.

### 11.4.1 Register User with OpenID

The User must first register and create a profile with the local OpenID provider as shown in Figure 12 below.



The screenshot shows a web form titled "GeoBliki OpenID Server" with a subtitle "Update Pat Cappelaere". The form contains the following fields and values:

Field	Value
Login	cappelaere
Full Name	Pat Cappelaere
Password	
Date of Birth	
Email	pat@cappelaere.com
Notification URI	xmpp:cappelaere@gmail.co
Organization	Vightel Corporation
Title	CEO
Country	US
TimeZone	Eastern
Language	en-US
Latitude	
Longitude	

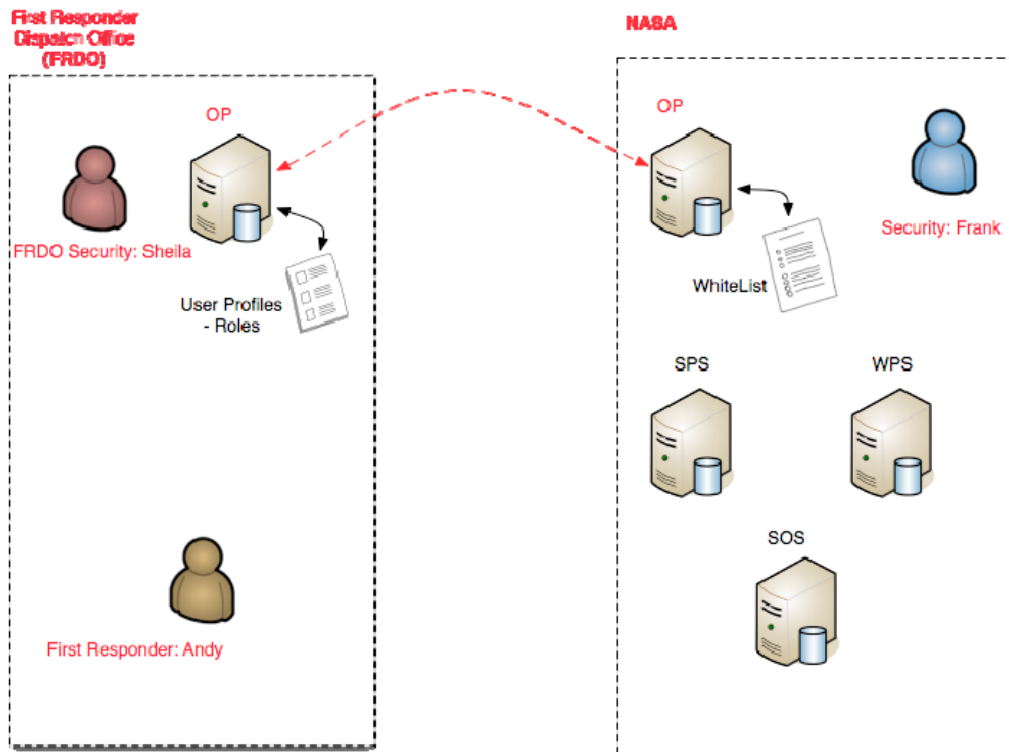
At the bottom of the form are two buttons: "Update" and "Cancel".

**Figure 12, OpenID Server User Profile Entry/Update**

This User Profile information also includes a Notification URI and associated protocol, which the user has chosen to receive communication and requests from the OpenID server. The Notification URI and protocol will be used to issue authorization requests to be shown later.

### 11.4.2 Establish Trust

In order to allow access to and for tasking resources in another domain, the user in the First Responder Dispatch Office domain initiates a request to add an identity and role to the NASA domain as shown in Figure 13 below. This capability to establish trust was accomplished in this testbed using experimental extensions added to the OpenID server libraries to be described in the following paragraphs.



**Figure 13, Establish Cross-Domain Trust**

To provide a means to establish trust across security domains in this testbed, several experimental extensions were added to the OpenID server capabilities as described in the following paragraphs.

1. The OpenID server was modified to maintain a list of trusted domains. Domains can be added on the fly to allow temporary trust.
2. The current trusted domain list can be accessed via any one of the following:

```
GET https://op.geobliki.com/trusted_domains # for HTML output format
GET https://op.geobliki.com/trusted_domains.xml # for XML output format
GET https://op.geobliki.com/trusted_domains.json # for JSON format
```

The JSON output result appears as follows:

```
[{"domains":"geobpms.geobliki.com","id":1},
{"domains":"op.geobliki.com","id":2},
{"domains":"redcross.org","id":3},
{"domains":"servir.geobliki.com","id":4}]
```

3. The server database was extended. A new Ruby controller class and database tables were added or modified to provide persistent store for trusted identities as follows:



### Trusted Domain Controller Class (in Ruby):

```
class TrustedDomainsController < ApplicationController
  layout "op_layout"
  before_filter :login_required
  active_scaffold :trusted_domain do |config|
    config.label = "OP Trusted Domains"
    config.list.columns = [ :domains]
    config.list.sorting = { :domains => 'ASC' }
  end
end
```

### Trusted Domain database table:

```
CREATE TABLE `trusted_domains` (
  `id` int(11) NOT NULL auto_increment,
  `domains` text NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=5 DEFAULT CHARSET=latin1;
```

In addition, the OpenID user database table was extended to include the following fields:

- Company Name
- User Title
- Permissions
- User Location Latitude
- User Location Longitude

4. The OpenID server database was extended to include a database table to register web applications. Registered applications (such as 'xyz') would have an OpenID of the form shown in the following example.

Example: <https://op.geobliki.com/webapp/xyz>

A new webapps table was added to the OpenID server database. The purpose of this database table was to store registration information for web applications provided by a registered user. The 'cert' attribute contains the application public key to be fetched by the data provider and used to verify the digital signature of the message.

```
CREATE TABLE `webapps` (
  `id` int(11) NOT NULL auto_increment,
  `user_id` int(11) default NULL,
  `name` varchar(32) default NULL,
  `url` varchar(256) default NULL,
  `description` text,
  `cert` text,
  `created_at` datetime default NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=14 DEFAULT CHARSET=latin1;
```

The data provider can access the web application information profile by using the OpenID of the web application and the following AX schema references:

```
http://axschema.org/x/webapp/name
http://axschema.org/x/webapp/url
http://axschema.org/x/webapp/description
http://axschema.org/x/webapp/cert
```

### 11.4.3 Registering Application with OpenID

Before the application consumer can obtain access to an application in another domain, it must register its identity with the local OpenID provider. This registration request includes a public and private key used to authenticate the application during exchange of identity and during request for authorization to access another service (or application).

The asymmetrical public-key cryptography approach uses a pair of keys for each user: one public and one private. Requests and data that are encrypted with the private key can only be decrypted with the public key, and vice versa. The private key is closely guarded and never shared; while the public key is distributed so the recipient can use it to decrypt the incoming request and data.

By encrypting a message with the public key, it can be ensured that the message is only readable by the holder of the corresponding private key. Sender authentication was accomplished by creating a digital signature with an algorithm that combines the sender's private key with the content of the message. By evaluating this signature with the sender's public key, the recipient can determine that the message actually came from the sender, and if it has been altered.

To establish that a particular key pair actually belongs to a particular entity, certificates were created by a trusted third party, known as a certificate authority (CA). These certificates, signed with the CA's public key, attest that entity X has public key Y. For this testbed, a local OS server provided the local CA to create the keys. For production deployments, an existing and trusted CA should be used.

### 11.4.4 Creating a Certificate Authority (CA)

A local CA was created on the local OS to create a self signed Certificate Authority. The relevant files and directories were created in a directory on the local server.

An example session used to create a Certificate Authority is shown below where bold text represents user input and <enter> representing the <enter> or <return> key.

```
$ /System/Library/OpenSSL/misc/CA.pl -newca <enter>
CA certificate filename (or enter to create)<enter>

Making CA certificate...
Generating a 1024 bit RSA private key
.....++++++
.....
.....++++++
```

```
writing new private key to './demoCA/private/cakey.pem'
Enter PEM pass phrase: capassword <enter>
Verifying - Enter PEM pass phrase: capassword <enter>
-----
You are about to be asked to enter information that will be
incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name or a
DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]: <enter>
State or Province Name (full name) [Maryland]: <enter>
Locality Name (eg, city) [Greenbelt]: <enter>
Organization Name (eg, company) [NASA]: <enter>
Organizational Unit Name (eg, section) [Goddard Space Flight Center]:
<enter>
Common Name (eg, YOUR name) []: Troy Ames <enter>
Email Address []: Troy.J.Ames@nasa.gov <enter>
$
```

#### 11.4.5 Create a Certificate Request

Next, a new certificate request was created using the CA function as shown below. The private key is written to "newkey.pem" and the request is written to the file "newreq.pem".

```
$ /System/Library/OpenSSL/misc/CA.pl -newreq <enter>
Generating a 1024 bit RSA private key
.....++++++
.....++++++
writing new private key to 'newkey.pem'
Enter PEM pass phrase: certpassword <enter>
Verifying - Enter PEM pass phrase: certpassword <enter>
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a
DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]: <enter>
State or Province Name (full name) [Maryland]: <enter>
Locality Name (eg, city) [Greenbelt]: <enter>
Organization Name (eg, company) [NASA]: <enter>
Organizational Unit Name (eg, section) [Goddard Space Flight Center]:
<enter>
Common Name (eg, YOUR name) []: Troy Ames <enter>
Email Address []: Troy.J.Ames@nasa.gov <enter>
```

```
Please enter the following 'extra' attributes to be sent with your
certificate request
A challenge password []: <enter>
An optional company name []: <enter>
Request is in newreq.pem, private key is in newkey.pem
$
```

#### 11.4.6 Extract Public and Private Keys

The public RSA key was then extracted from the certificate using the “openssl rsa” which writes it to the “pubkey.pem” file as shown below. The content of this file was used later to register the web application.


```
$ openssl rsa -in newkey.pem -pubout -out pubkey.pem <enter>
Enter pass phrase for newkey.pem: certpassword <enter>
writing RSA key
$
```

The private key was extracted from the certificate into the PKCS#8 format required by the Java Security API using the “openssl pkcs8” command as shown below. The private key was written to the “privkey.pem” file which was used by the web application to sign future requests.

```
$ openssl pkcs8 -topk8 -nocrypt -in newkey.pem -out privkey.pem <enter>
Enter pass phrase for newkey.pem: certpassword <enter>
$
```

#### 11.4.7 Create a Web Application Entry on the OpenID Server

The public key generated in the previous steps was then used to create a new Web Application Entry for the application with the OpenID server. For this testbed the Geobliki OpenID server displays a user interface form that was used to enter required information including the content of the “pubkey.pem” file created earlier to complete the application entries as shown in Figure 14.



## GeoBliki OpenID Server


Your Web Apps [Search](#) [Create New](#)

Name	Openid	Url	Description	date
<b>Update JavaTest1</b>				
App Name	<input type="text" value="JavaTest1"/>			
App URL	<input type="text"/>			
App Description	<input type="text" value="A test using Java to create a campaign and targets"/>			
Public Key	<pre> -----BEGIN PUBLIC KEY----- MIGMA0GCsGqSib3DQEBAAUAA4GNADCBiQKBgQCoB+O6VJl7Uk wVIGJ79DD9aFIZ Ua84mMzHlWDFla+4r01ctONnTX7clV0Sp0DoZ/9ijU+ccL9HJmQsTW 0QxaZz7Hy n/ITI53o7WA+i3HuRogM6LJ1NGoqUphKVwbc2CwNlVevGNWy69Z3w fpaU+VFRMge tgvybAjl3imgUBeYwkQIDAQAB -----END PUBLIC KEY----- </pre>			
<input type="button" value="Update"/> <input type="button" value="Cancel"/>				

1 Found

**Figure 14, Create a new Web Application Entry on the OpenID Server**

The new generated OpenID (<https://op.geobliki.com/app/JavaTest1>) shown in Figure 15 will be used as the OAuth consumer key for future requests by this application.



## GeoBliki OpenID Server

Your Web Apps [Search](#) [Create New](#)

Name	Openid	Url	Description	date	
JavaTest1	<a href="https://op.geobliki.com/app/JavaTest1">https://op.geobliki.com/app/JavaTest1</a>	..	A test using Java to create a campaign and targets	03/19/2009 08:35 PM	<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Show</a>

1 Found

[Home](#) | [Help](#) | [Feedback](#) | [Blog](#) | [GeoBliki](#)

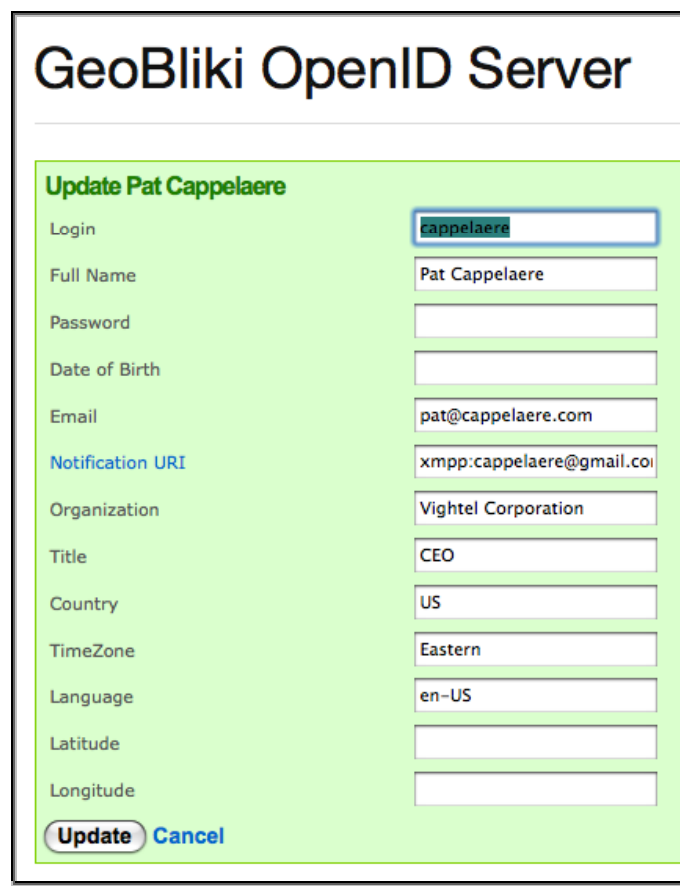
NOTE: This is an EXPERIMENTAL Prototype OpenID Server NOT To Be Used For Operations

**Figure 15, OpenID Application Consumer Key**

### 11.4.8 Cross-Domain Authority Delegation

OAuth was originally designed independently of OpenID to allow users to provide access to their resources to other services without having to relinquish their usernames and passwords to those services.

In this testbed the OpenID server and the Service Provider were not in the same security domain. To access resources in another domain, permission must be obtained by the consumer to access resources published by a provider in a different security domain on the user's behalf. The user must be registered to a trusted community of OpenID providers and have a notification capability, such as XMPP or SMS, shown as Notification URI below to support authorizations in real-time. The OpenID server interface used in this testbed to enter the user's identity profile is shown in Figure 16 below.



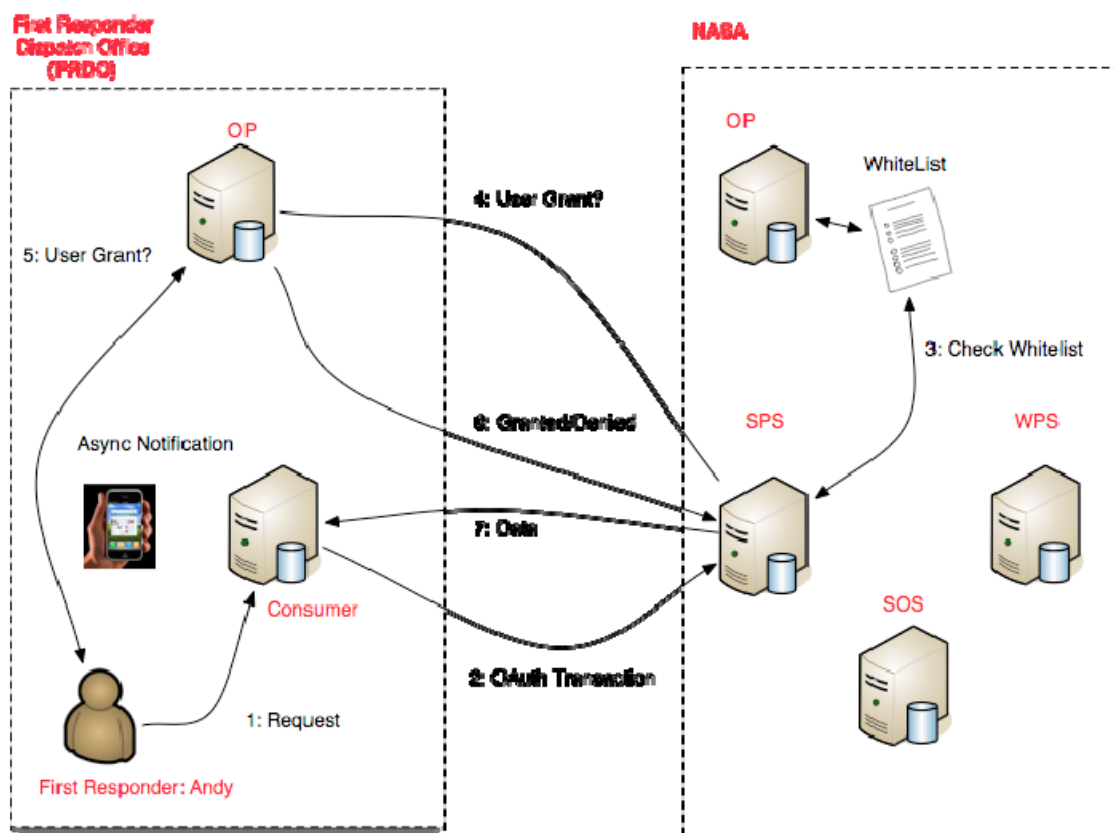
The screenshot displays the 'GeoBliki OpenID Server' interface. The main heading is 'GeoBliki OpenID Server'. Below it, a green-bordered box contains the title 'Update Pat Cappelaere'. The form includes the following fields and values:

Field	Value
Login	cappelaere
Full Name	Pat Cappelaere
Password	
Date of Birth	
Email	pat@cappelaere.com
Notification URI	xmpp:cappelaere@gmail.com
Organization	Vightel Corporation
Title	CEO
Country	US
TimeZone	Eastern
Language	en-US
Latitude	
Longitude	

At the bottom of the form are two buttons: 'Update' and 'Cancel'.

Figure 16, OpenID Server Interface to enter User's Identity Profile

Given the OpenID profile defined above, the transaction can be achieved using OAuth and OpenID as shown in Figure 17 and described in the paragraphs that follow.



**Figure 17, Delegation of Authority using OAuth and OpenID**

The Application Consumer uses the user's OpenID as the OAuth token. The Service Provider (SPS) then uses the OpenID to retrieve the user profile that has been established across the secure domains. Notification is provided to the user's OpenID provider that a request is being made to access resources on his behalf. The user then accepts or rejects the request. The grant will remain permanent until explicitly revoked by the user. The following steps describe the sequence of interactions depicted in Figure 17 shown above.

#### 11.4.9 Application Consumer OAuth request

The application configuration properties contained in the `oauth.properties` file is shown below:

```
# User (Resource Owner) OpenID
oauth_token=https://op.geobliki.com/user/joeuser
# Client (Web Application) OpenID
oauth_consumer_key=https://op.geobliki.com/app/JavaTest1
# Client RSA private key file used to sign request
oauth_signature_private_key_file=examples/privkey.pem
# Client RSA public key file used to validate RSA private key signing
oauth_signature_public_key_file=examples/pubkey.pem
# Request realm, not included in signature of request but needed by
server
realm=/wfcs
```

- The “oauth\_token” property is the user OpenID created in Clause 11.4.1.
- The “oauth\_consumer\_key” property is the OpenID assigned to the Web Application in Clause 11.4.7.
- The “oauth\_signature\_private\_key\_file” property is the relative path to the “privkey.pem” file created in Clause 11.4.6.
- The “oauth\_signature\_public\_key\_file” property is the relative path to the “pubkey.pem” file created in Clause 11.4.6.

The public key is used by the application to validate the OAuth message signature ensuring that the private and public keys are consistent with each other. The “realm” property is needed by the Service Provider.

#### 11.4.10 GeoBPMS Data Provider (SPS) Response

The following paragraph describes the actions taken upon receipt of the OAuth request by the GeoBPMS. It:

- extracts the User OpenID and the Application OpenID from the OAuth request.
- requests the OpenID Server (<https://op.geobliki.com/>) to authenticate the Application Consumer OpenID and return the public key provided when the application entry was created
- validates the request signature using the public key
- requests the OpenID Server to authenticate the User OpenID
- requests the OpenID Server to authorize request based on user profile and user granted authorizations

If the OpenID Server cannot authorize the request based on its own user grants then it will delegate the authorization by issuing an authorization request to the Notification URI in the user profile provided in Clause 11.4.1. Based on the response to the authorization request it will respond to the application with an accepted or denied status message. The Application Consumer creates and sends an OAuth message to the Service Provider using http protocol as shown below.

```
POST /scenarios HTTP/1.1
Accept: */*
```



```
Connection: close
Content-Type: application/x-www-form-urlencoded
Authorization: OAuth realm="/wfcs",
oauth_consumer_key="https%3A%2F%2Fop.geobliki.com%2Fapp%2FJavaTest1",
oauth_nonce="1238620576008125000", oauth_timestamp="1238620576",
oauth_signature="aTV%2FcnahX7MnXKEQ5fZS%2BogVLTpaYXEe8uyygk72QhEafTMAVG
WjCD%2Fka4NB7tUQYVZH4norXE6JGbpA%2B1WHNptjRRp8G3HHZ5Trna9xQ2IRp%2FkDGXq
LLmMmahixQEi2fv6oGjTpImPrMRCNcQoTYcY5MT%2B4z6pUU4j34lDcB80%3D",
oauth_signature_method="RSA-SHA1",
oauth_token="https%3A%2F%2Fop.geobliki.com%2Fuser%2Ftjames",
oauth_version="1.0"
User-Agent: JavaOpenIDTest/0.01
Content-length: 521
Cache-Control: no-cache
Pragma: no-cache
Host: http://geobpms.geobliki.com

raw_post_data=%3Centry%20xmlns%3Ageobpms%3D%22http%3A%2F%2Fgeobpms.geob
liki.com%2F1.0%22%3E%0A%20%20%20%20%3Ctitle%3ETestCampaign1%3C%2Ftitle%
3E%0A%20%20%20%20%3Ccategory%20scheme%3D%22http%3A%2F%2Fgeobpms%2F1.0%2
2%20term%3D%22flood%22%2F%3E%0A%20%20%20%20%3Cauthor%3E%0A%20%20%20%
20%20%20%20%3Cname%3ETroy%3C%2Fname%3E%0A%20%20%20%20%3C%2Fauthor%3E%0A
%20%20%20%20%3Ccontent%3EThis%20is%20a%20Test%20Campaign%3C%2Fcontent%3
E%0A%20%20%20%20%3Cgeobpms%3Aitem_type%3Escenarios%3C%2Fgeobpms%3Aitem_
type%3E%0A%3C%2Fentry%3E
```

The Application Consumer then waits for a response from the Service Provider. The Service Provider and the OpenID Server communicate to perform an authorization exchange in order to accept or deny the request. The Service Provider sends a request to the User's Notification URI requesting authorization. Once the authorization is accepted the Service Provider replies with a status of "201 Created". The response to the authorization request is shown below.

```
201 Created
Status=[201 Created]
Set-Cookie=
[_geobpms2_session_id=b76776bd892de248988c9075764d8b8e; path=/]
Served-By=[Joyent]
Date=[Fri, 03 Apr 2009 13:50:36 GMT]
Content-Type=[application/xml; charset=utf-8]
Via=[1.1 geobpms.geobliki.com]
Cache-Control=[no-cache, max-age=1800]
Connection=[close]
Content-Length=[1]
Expires=[Fri, 03 Apr 2009 14:20:36 GMT]
Server=[Mongrel 1.1.5]
null=[HTTP/1.1 201 Created]
Location=[http://geobpms.geobliki.com/scenarios/98]
Vary=[Accept-Encoding]
```

In this testbed and application, the authorization grant will remain in effect until revoked so requests will be authorized without subsequent notifications.

### 11.4.11 Target Service Request to Service Provider

Having received authorization to submit requests to the Service Provider, the Application Consumer can now send a request to the Service Provider as shown below

```
<entry xmlns:geobpms="http://geobpms.geobliki.com/1.0">
  <title>TestRequest1</title>
  <category scheme="http://geobpms/1.0" term="flood"/>
  <author>
    <name>Troy</name>
  </author>
  <content>This is a Test Request - Troy</content>
  <geobpms:item_type>scenario_requests</geobpms:item_type>
  <geobpms:latitude>38.8583</geobpms:latitude>
  <geobpms:longitude>100.4103</geobpms:longitude>
</entry>
```

The Service Provider responds to the Application Consumer as shown below.

```
200 OK
Status=[200 OK]
Set-Cookie=[_geobpms2_session_id=2755a03c0c69d627f5fb01fa87e4e88d;
path=/]
Served-By=[Joyent]
Date=[Fri, 03 Apr 2009 17:37:25 GMT]
Content-Type=[application/xml; charset=utf-8]
Via=[1.1 geobpms.geobliki.com]
Cache-Control=[no-cache, max-age=1800]
Connection=[close]
Content-Length=[78]
Expires=[Fri, 03 Apr 2009 18:07:25 GMT]
null=[HTTP/1.1 200 OK]
Server=[Mongrel 1.1.5]
Vary=[Accept-Encoding]
```

## 12 XACML Policy Models

XACML [7]and [8] defines a language for creating policies. Policies are composed of one or more Policy and PolicySet. A PolicySet may contain one or more Policy and references to one or more external Policy. A Policy is composed of the following elements:

- target
- rule-combining algorithm
- rules
- obligations

The XACML specifications define several combining algorithms for determining a single decision from the results of multiple rules.

## 12.1 XACML Policy Language Model

The XACML Policy Language Model defines an XML encoding for expressing general purpose access restrictions and extension points. The entire set of access restrictions defines an XACML Policy. The Policy is structured as shown in Figure 18 below. The top level element is the <PolicySet>. It can host zero or more <PolicySet> elements, which can be included inline or by reference. This feature allows the reuse of pre-defined policy segments as well as the integration of multiple policies. Each <PolicySet> element can host one or more <Policy> elements, which is the container for a set of <Rule> elements. Inside the <Rule> element, conditions can be formed to express complex access restrictions, using the <Condition> element. Each <PolicySet>, <Policy> and <Rule> element have a <Target> element, which can be used to define simple matching conditions for the Subject, Action, Resource and Environment. This allows the effective structuring of a policy into sub-trees, which eases the maintenance of rights defined in a policy. On the other hand, the simple matching in a <Target> element ensures fast decision making, when it comes to deriving an authorization decision.

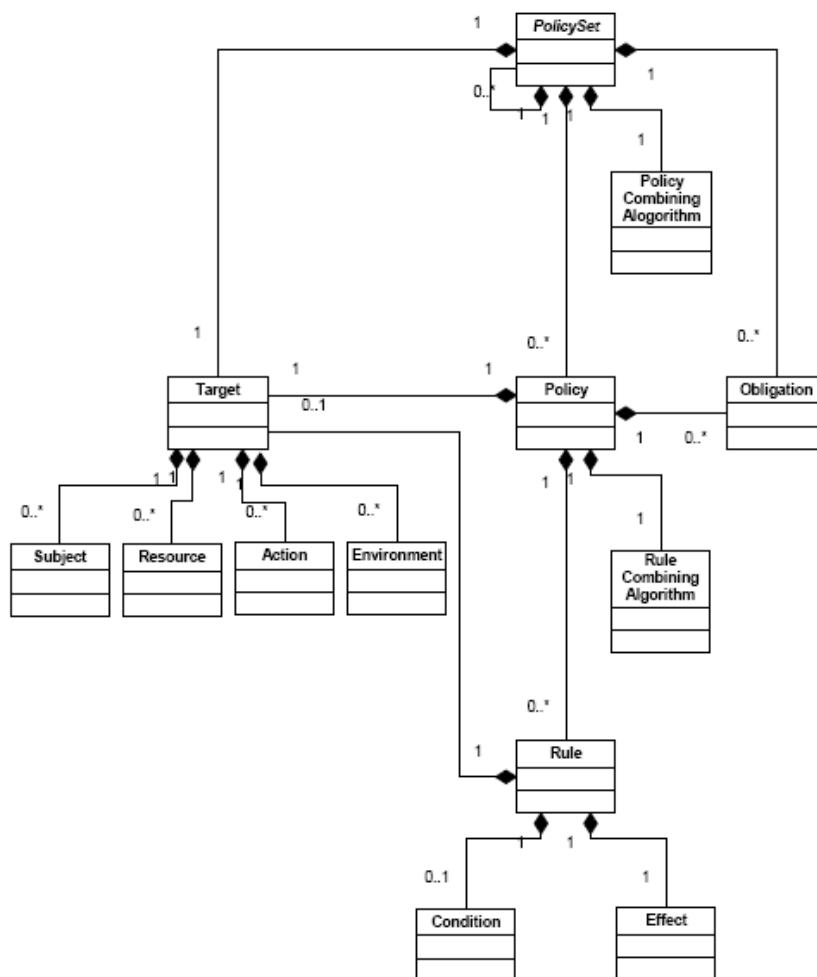


Figure 18, XACML Language Policy Model

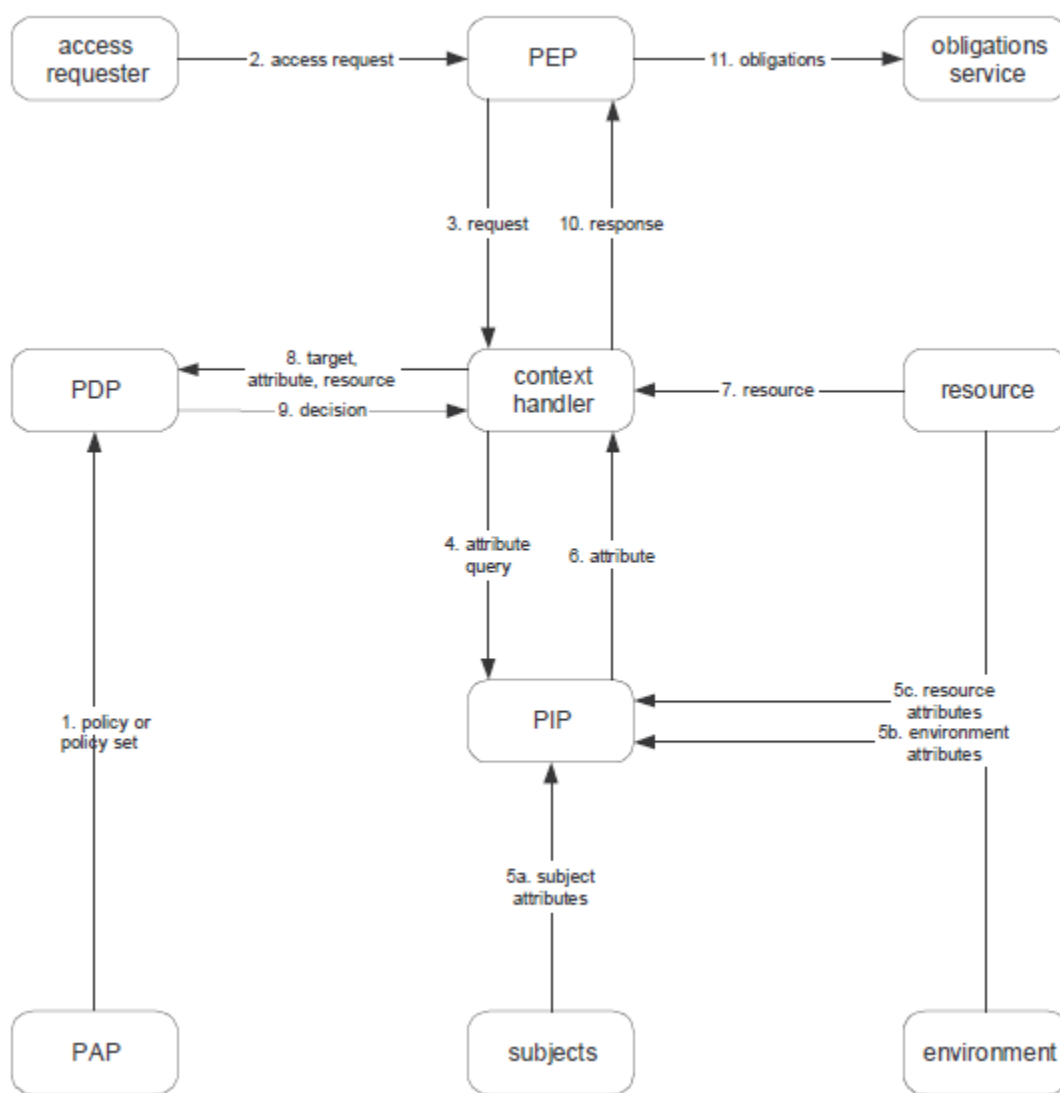
The flexible matching of Subjects in the <Target> element supports direct association of access rights to subjects or roles, as defined in the RBAC profile of XACML [9]. In order to derive an authorization decision for a given request, the XACML policy is traversed from the top (i.e. <PolicySet> element) to the leaves (i.e. <Rule> elements). For all matching <Rule> elements, their Effect (i.e. Permit, Deny, etc.) is taken as the most basic driver for the authorization decision. By traversing up the policy the effects of all Rules – associated to a <Policy> element – are combined using the RuleCombiningAlgorithm. The resulting effects of all <Policy> elements are matched on the next highest level, until reaching the top <PolicySet> element; the PolicyCombiningAlgorithm creates the final effect of the entire policy, which represents the authorization decision.

The XACML Policy Language defines four different results for the authorization decision: (i) Permit, (ii) Deny, (iii) Indeterminate and (iv) NotApplicable. Finally, the process of deriving an authorization decision can result in an error, which is documented as additional information in the <Decision> element. In addition, the decision can be “Permit with Obligation”, which can be expressed in the <Obligation> element, attached to the <Policy> or <PolicySet> element

Policies define permissions for subjects. Subjects are typically requestors and may be defined as individuals, user groups, applications, etc. Permission allows a subject to perform a certain action on a certain resource.

## **12.2 XACML Data Flow Policy Model**

OWS-6 used the XACML [8] policy models. The XACML specification allows defining conditions and obligations in addition to (subject, resource, and action) triples.



**Figure 19, XACML v1.0 Data Flow Model**

A resource may be a service, a layer of a WMS, a feature type of a WFS, a single XML node of the service request or response, etc. Actions can be simply general access to a resource, service operations such as GetMap or GetFeature, etc. It is part of the policy editor's responsibility to express subjects, actions and resources for a given use case.

A policy is evaluated within the Policy Decision Point (PDP). A Policy Enforcement Point (PEP) submits an XACML request context to the PDP, including information about subject, resource, and action. Furthermore this XACML request context may include environment attributes, which may be evaluated by conditions.

It is necessary that the XACML request context corresponds to the policy model which shall be applied. If an XACML request context does not match to the subject, resource, and action policies defined, a policy evaluation will fail. Therefore there is a logical

dependency between access decision request and the stored policy documents on the PDP.

If all applicable rules within the policy or the policy set evaluate to ‘permit’, then the PDP replies with a response context including a “Permit” statement. If the policy evaluation leads to a denial, the PDP responds with a “Deny”. The message “NotApplicable” is returned if there is no policy which fits the XACML request context, and errors during the policy evaluation lead to an “Indeterminate” response.

A PEP may only allow access as requested by the subject, if the PDP responds with a “Permit” which may include access restrictions using filtering capabilities of XACML based on resource-id contained in the access decision response. All other responses have to result in a denial of access, due to the use of deny biased PEPs within this testbed.

### 12.3 Obligations

Obligations are part of the XACML standard and are defined within policies, but they have no direct effect on the policy evaluation. Whenever a policy permitting a request context contains one or more obligations, those obligations are included in the XACML response context.

Obligations express actions that must be fulfilled by the PEP before executing the PDP decision. If a PEP fails in fulfilling the obligation, either because the action required by the obligation cannot be performed or because the PEP does not understand the obligation, the effect has to be a denial of access.

Whenever obligations are used, there is a dependency between policies and PEP. If a PEP shall fulfill obligations defined in a certain policy, it has to know how this can be achieved. If a PEP is not able to do so, the authorization is invalid and the requested action has to be denied. Thus, obligations are only useful as long as a PEP is able to interpret and fulfill them.

### 12.4 Spatial Authorization

Spatial authorization is a special requirement for geo services. If spatial authorization is required, resources can no longer be regarded as atomic. Policies no longer grant access to a resource in general, but they have to define policies on geometries as subsets of this resource. This is not directly addressed in the XACML standard, but there are different approaches to fulfill this requirement based on XACML. The following paragraphs describe two different but related approaches used in this testbed:

- GeoXACML
- XACML using Spatial Obligations

Although these approaches are significantly different, they were both found appropriate to fulfill the actual requirements on spatial authorization defined for this testbed.

### 12.4.1 GeoXACML

The Geospatial eXtensible Access Control Markup Language (GeoXACML) defines a geo-specific extension to the XACML [7].

**For additional details, which describe the implementation of GeoXACML in the context of OWS-6 and other architectural and implementation issues, the reader should refer to OGC 09-036r1 “OWS-6 GeoXACML Engineering Report”.**

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.

#### 12.4.1.1 GeoXACML’s Geometry Model and 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 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 1: Spatial functions provided by GeoXACML**

#### 12.4.1.2 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:

- Definition of fine grained, positive and negative access control rules (i.e. the atomic access control object is an arbitrary XML node).
- Authorization decisions based 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.
- Definition of content dependant access control rules.
- Definition of spatial access control rules.
- Definition of context dependant access control rules.
- Flexible combination of all kind of rule types.
- Easy implementation of a spatial access control system based on standardized and expressive policies

#### 12.4.2 XACML with Spatial Obligations

A PDP may not always be able to achieve the desired spatial authorization for a given service type, or it may require many authorization decisions, which could reduce



performance of a security system. Examples are described in the following paragraphs to illustrate the possible advantages that obligations may offer in these cases.

**WMS GetMap.** Consider a WMS GetMap request, resulting in an image, can only be handled as a single information item, which only allows a PDP to grant access to the complete image, or deny access completely. Using an obligation, the access decision response could permit access to the image, but require the PEP to remove all areas of the image which are unauthorized. Conversely, without using obligations, where a GetMap bounding box intersects an unauthorized area, the access decision response would result in denying access to the allowed area.

**WFS GetFeature.** Consider a WFS GetFeature request, which may return megabytes of GML data. If certain feature attributes have restrictions that must be authorized, a PDP could do so without obligations by authorizing each feature attribute in the GML response. This approach is possible but may be inefficient. By contrast, using obligations could permit access by requiring the PEP to add an OGC filter to the GetFeature request, resulting in a WFS response where the unauthorized elements are excluded from the GetFeature result. This would reduce the efforts for authorization to only one decision and the fulfillment of the obligation in the PEP.

In this manner, a PDP can be used for handling standard XACML with any kind of obligations as an alternative to use of geospatial extensions within the PDP.

Currently, there is no standardized way of defining spatial obligations, although this would be desirable to ensure interoperability between security systems and to enable exchange of policies with obligations from one security system to another.

Since no standard exists to define how to describe obligations the approach taken in this part of OWS-6 was to use an OGC filter expression within a WFS GetFeature request to express spatial restrictions. This approach allows one to define a buffer around features, to define if coordinate transformations of this geometry shall be allowed or not (transformation may lead to inexact results), and to define if features may intersect or if they must be completely within the authorized geometry in order to be permitted.

Another alternative would be to use encodings for geometric data types, such as defined in GML or GeoXACML, directly in those obligations instead of using reference to certain features within a WFS. However, this might result in geometries to be encoded redundantly within different policies or on different PDPs, while the policies refer to the same features types. Alternatively, a PDP could dynamically include geometry encodings in obligations of an authorization decision response based on policy, with accompanying adaptations for the PDP implementation.

#### 12.4.2.1 WMS Example

For a WMS, the supported operations GetCapabilities, GetMap and GetFeatureInfo can be regarded as actions in terms of the XACML terminology. XACML resources can be mapped to the layers offered by the WMS. Thus, simple access control without spatial

restrictions could authorize certain operations for certain layers for certain users or user groups. It should be noted that the policies shall be consistent among the different operations, especially the GetCapabilities response should be filtered according to the policies. Whenever a certain operation is not allowed, then the corresponding GetCapabilities response should not include this operation, and if the policies deny certain layers the capabilities should not advertise those forbidden layers. Of course this cannot be a general rule, since there may be application specific requirements making other behavior necessary, but a consistent behavior of a secured service should be regarded as a general guideline when creating policies.

A policy including a spatial obligation is shown below in Figure 20. This example shows a policy for a WMS. In the <Subject> element the subject is identified by the <AttributeValue> 'guest', which is a role, following the URN in the <SubjectAttributeDesignator> element.

The resource is identified within the <Resource> element. In this example the value there is 'service.wms#layer::http://wms-url.com/wms#layername', indicating that this is a WMS service and adding the layer name to the service URL.

The allowed actions are defined in the <Action> elements, in this example carrying the values 'service.wms::GetMap' and 'service.wms::GetCapabilities'.

The above mentioned part of the policy allows a user being assigned to the role 'guest' to perform GetMap and GetCapabilities operations on the layer 'layername' of the WMS '://wms-url.com/wms'.

```
<Policy PolicyId="guest_spatially_authorized_SE"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:first-applicable">
  <Description>Description</Description>
  <Target>
    <Subjects>
      <Subject>
        <SubjectMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">guest</AttributeValu
e>
          <SubjectAttributeDesignator
AttributeId="urn:conterra:names:sdi-suite:policy:attribute:role"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="http://www.sdi-
suite.de/securitymanager/xacml/names/function#string-equals-ignore-
case">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">service.wms#layer::h
ttp://wms-url.com/wms#layername</AttributeValue>
```

```

        <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </ResourceMatch>
</Resource>
</Resources>
<Actions>
    <Action>
        <ActionMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">service.wms::GetMap<
/AttributeValue>
                <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </ActionMatch>
        </Action>
    </Action>
        <ActionMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">service.wms::GetCapa
bilities</AttributeValue>
                <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </ActionMatch>
        </Action>
    </Actions>
</Target>
<Rule RuleId="Access_granted" Effect="Permit">
    <Description>Description</Description>
</Rule>
<Obligations>
    <Obligation ObligationId="urn:conterra:names:sdi-
suite:policy:obligation:spatialauthz::Spatial_Obligation"
FulfillOn="Permit">
        <AttributeAssignment
DataType="http://www.w3.org/2001/XMLSchema#boolean"
AttributeId="spatial.authz.transformation.allowed">true</AttributeAssig
nment>
        <AttributeAssignment
DataType="http://www.w3.org/2001/XMLSchema#double"
AttributeId="spatial.authz.buffer.size">0.0</AttributeAssignment>
        <AttributeAssignment
DataType="http://www.w3.org/2001/XMLSchema#string"
AttributeId="spatial.authz.feature.srs">EPSG:4326</AttributeAssignment>
        <AttributeAssignment
DataType="http://www.conterra.de/xsd/spatialauthzfilter"
AttributeId="spatial.authz.spatialfilter">
            <spat:SpatialFilter
xmlns:spat="http://www.conterra.de/xsd/spatialauthzfilter">
                <spat:GeometrySource Location="http://wfs-url.com/wfs"
Type="wfs" Schema="Demo_European_Countries:European_Countries">
                    <spat:GeometryRef GeomField="Shape">
                        <ogc:Filter xmlns:ogc="http://www.opengis.net/ogc">

```

```

        <ogc:FeatureId fid="F3__105"/>
        <ogc:FeatureId fid="F3__103"/>
    </ogc:Filter>
</spat:GeometryRef>
</spat:GeometrySource>
</spat:SpatialFilter>
</AttributeAssignment>
</Obligation>
</Obligations>
</Policy>

```

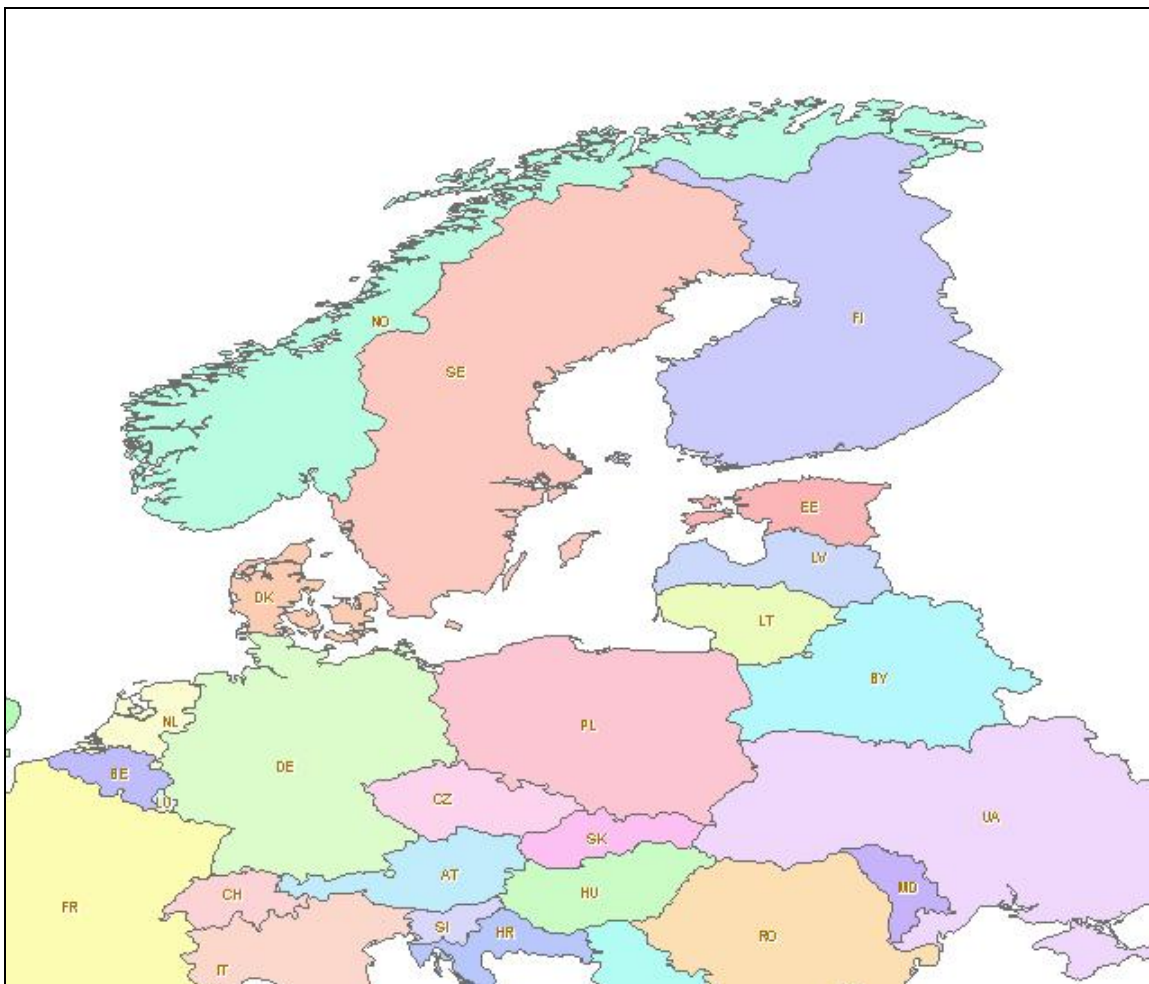
**Figure 20, Policy for WMS with Spatial Obligations**

Whenever such permission is granted, all obligations mentioned in the `<Obligations>` element and carrying the value 'permit' in the 'FulfillOn' parameter are added. In this example this is a spatial obligation. This obligation indicates that coordinate transformations are allowed ('spatial.authz.transformation.allowed' is true), that there is no buffer around the authorized geometry ('spatial.authz.buffer.size' is 0.0), the spatial reference system of the feature used for authorization ('spatial.authz.feature.srs': EPSG:4326), and it refers to the geometries representing the authorized area (WFS URL = 'http://wfs-url.com/wfs ', attribute name = 'Shape', feature IDs = 'F3\_\_105' and 'F3\_\_103').

The corresponding PEP shall now fulfill this obligation by restricting a WMS response to the geometry of the referenced features. As long as a WMS GetMap response lies completely within those feature geometries, no further action has to be performed. Once the GetMap response includes geometries outside the referenced feature geometries, all image content representing unauthorized content has to be erased.

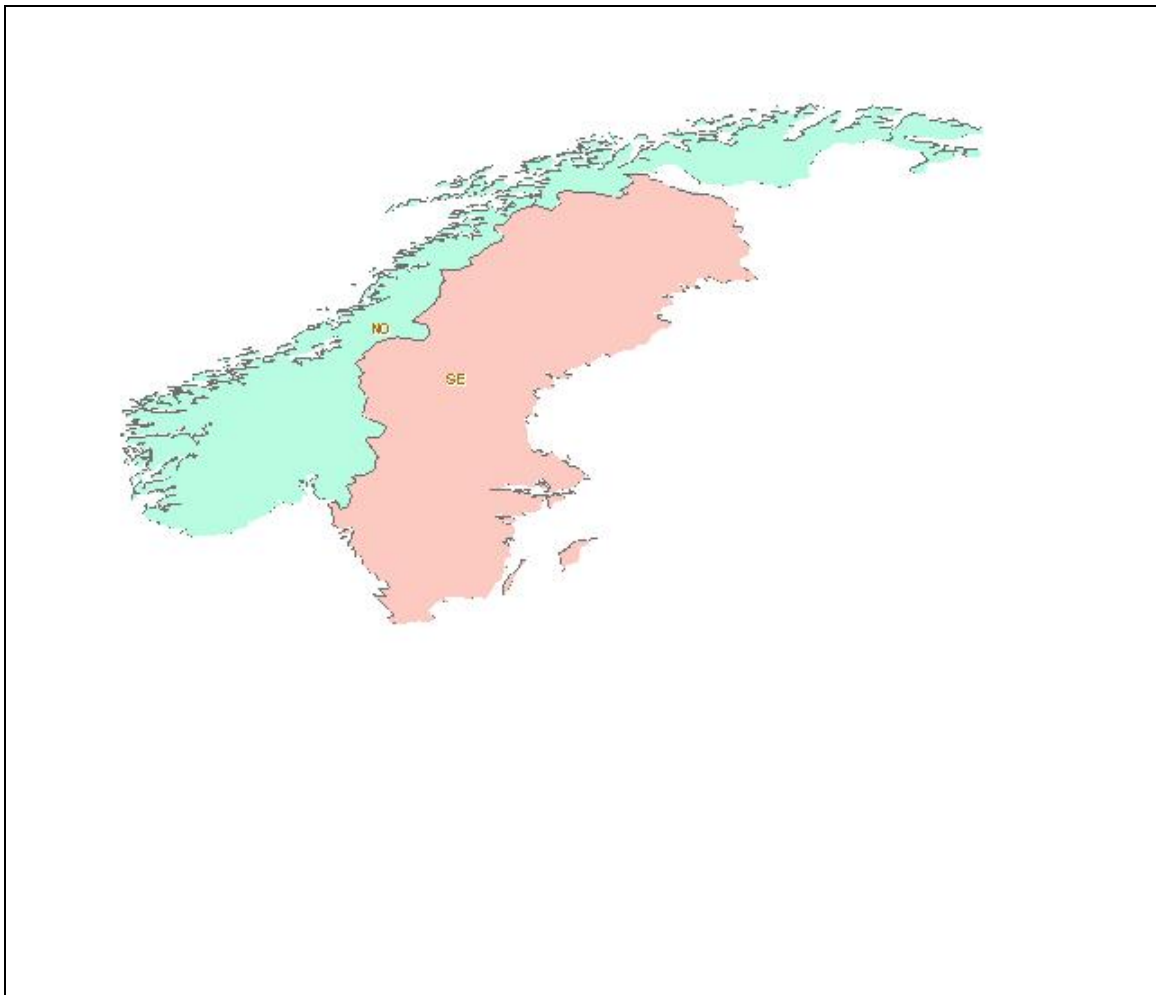
For GetFeatureInfo requests the effect is to block all GetFeatureInfo requests for a unauthorized coordinate.

The effect of a spatially restricted WMS GetMap request is visualized in Figure 21 and Figure 22 which follow



**Figure 21, WMS GetMap Response**

Figure 21 shows a GetMap response, showing European countries, as it was created by the service. Given that there are spatial restriction, allowing only access to Norway and Sweden, the PEP would have to erase all other information, resulting in the image shown in Figure 22.



**Figure 22, Restricted GetMap Response**

#### **12.4.2.2 WFS Example**

For WFS the operations can be regarded as XACML actions, and the different feature types offered by a WFS can be mapped to resources. Of course it is possible to choose another granularity of resources, such as for individual features, but a granularity on feature type level was chosen for this testbed.

Instead of a more fine-grained resource model, obligations can be used with WFS to enforce spatial restrictions as well as restrictions on feature- or even on attribute level.

The obligation approach for WFS used within OWS-6 adds OGC Filter statements to a WFS request, extending the initial query received by the WFS client. Of course, this approach to rights enforcement assumes that the corresponding WFS applies the filter expression correctly. If the WFS cannot be trusted to do so, it might be desirable to authorize the response by evaluating if there are any unauthorized features included. This could be done either by the PEP itself or in a more generic way by a GeoXACML-

capable PDP. However, it heavily depends on security and performance requirements if this additional check is needed or not.

An example of a WFS GetFeature request submitted to a PEP is shown below in Figure 23. The SOAP header of this request contains a SAML holder-of-key assertion, identifying the requestor and being signed by the issuing STS. This assertion includes the requestor's public key, and the requestor used its corresponding private key to apply a signature to the message body and a timestamp in the message header (referenced by the 'Reference' element within the 'SignedInfo' element of the signature). Referencing the timestamp makes the signature unique, since even for two identical requests the timestamp always will differ. This is important to prevent replay attacks, since even two identical consecutive requests will have different signatures due to the different timestamps.

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
  <soapenv:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <wsse:Security xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
soapenv:mustUnderstand="1">
      <wsu:Timestamp xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
wsu:Id="Timestamp-11605653">
        <wsu:Created>2009-04-01T11:23:18.078Z</wsu:Created>
        <wsu:Expires>2009-04-01T11:28:18.078Z</wsu:Expires>
      </wsu:Timestamp>
      <xenc:EncryptedKey Id="EncKeyId-
urn:uuid:E7186978362EF90E6A12385849980784">
        <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p"/>
        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
          <wsse:SecurityTokenReference>
            <wsse:KeyIdentifier
EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soap-message-security-1.0#Base64Binary" ValueType="http://docs.oasis-
open.org/wss/oasis-wss-soap-message-security-
1.1#ThumbprintSHA1">HYL371NzoOs2+IA24VDkBGcUFQM=</wsse:KeyIdentifier>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
        <xenc:CipherData>
          <xenc:CipherValue>Pw4nZOpoOwvgjdXGuFzIypNPCwjDc3tjoKV+H2nYfuKDjwADCV
sw4802/vqG0D3wIgrlC9b5UM087parJnLaFAVATNGvQIqmi job9jdI9220qIlsiyjlcNbya
nE5UsCR24xuPTALu64LyRLCVhyfYT8RBobB1kiJv7PnXB3CqSw=</xenc:CipherValue>
        </xenc:CipherData>
      </xenc:EncryptedKey>
      <Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
AssertionID="_027784803c4a81a5334231fecc827636" IssueInstant="2009-04-
01T11:22:00.359Z" Issuer="STS" MajorVersion="1" MinorVersion="1">
```



```

        <Conditions NotBefore="2009-04-01T11:21:58.046Z"
NotOnOrAfter="2009-04-01T11:26:58.046Z"/>
        <AuthenticationStatement AuthenticationInstant="2009-04-
01T11:21:58.046Z"
AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
            <Subject>
                <NameIdentifier
Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress">alice</NameIdentifier>
                <SubjectConfirmation>
                    <ConfirmationMethod>
                        urn:oasis:names:tc:SAML:1.0:cm:holder-of-key
                    </ConfirmationMethod>
                    <KeyInfo
xmlns="http://www.w3.org/2000/09/xmldsig#">
                        <X509Data
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                            <X509Certificate>
MIICKjCCAZMCBEmmTwwwDQYJKoZIhvcNAQEEBQAwxDELMAkGA1UEBhMCZGUxDDAKBgNVBAG
TA25ydzERMA8GA1UEBxMIbXVlbnN0ZXIxDTALBgNVBAoTBGhvbWUxDTALBgNVBAstBGhvbW
UxDjAMBgNVBAMTBWFSaWNlMB4XDTA5MDIyNjA4MTMwMFoXDTE5MDEwNTA4MTMwMFowXDELMA
kGA1UEBhMCZGUxDDAKBgNVBAGTA25ydzERMA8GA1UEBxMIbXVlbnN0ZXIxDTALBgNVBAoT
BGhvbWUxDTALBgNVBAstBGhvbWUxDjAMBgNVBAMTBWFSaWNlMIGfMA0GCSqGSIb3DQEBAQU
AA4GNADCBiQKBgQC017qyAHUe/SXiPBG1IrwLEmdnPCQuXJYU11oiWMOcKl04kaWiKhj+IX
OhvQLz55StQwS6CRAjw0kWHXM/wyM3OnnWtDHhzFDirBcQkz6sqwYgTFOqxYK4xQVhTQj5i
xd/64qLqndfZ56Q/Tj2OaKLXY4X4YXGxYYQmY31KnOE4wIDAQABMA0GCSqGSIb3DQEBAUA
A4GBABadBiu923rPhklwmuXLJBgbgobfb6E12OLBly+JjVbpyaTAZK4urKBV1NktaLJxtFHT
UyrbIzs7ZFczwv214SR+ARTl2AtMACnfBlwS4dR0E+ZDSybhQ6qh/UAl1XdQBnzNnLVbiLE
bw0WWxCgpYxH4YLCpljq12udhggqmRrvNE</X509Certificate>
                            </X509Data>
                        </KeyInfo>
                    </SubjectConfirmation>
                </Subject>
            </AuthenticationStatement>
            <AttributeStatement>
                <Subject>
                    <NameIdentifier
Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress">alice</NameIdentifier>
                </Subject>
                <Attribute
AttributeName="urn:n52:authentication:subject:principal:role"
AttributeNamespace="def">
                    <AttributeValue>alice</AttributeValue>
                    <AttributeValue>admin</AttributeValue>
                </Attribute>
            </AttributeStatement>
            <ds:Signature
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                <ds:SignedInfo>
                    <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                    <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1">
                    <ds:Reference
URI="#_027784803c4a81a5334231fecc827636">
                    <ds:Transforms>

```



```

        <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
        <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
            <ec:InclusiveNamespaces
xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#" PrefixList="code ds
kind rw saml samlp typens #default xsd xsi"/>
            </ds:Transform>
        </ds:Transforms>
        <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

        <ds:DigestValue>/AsUaldT6xUBKs1Sy798eJRLJTI=</ds:DigestValue>
        </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>
em1j9/Tc4hWoZXfnJlhHajFABIXyQlhqhvnNWUmaOvcZlYiLvNDXjjQaKTBXn0Jyle6NvYyR
UgXsGJsrfzkWBV5aSfHuzJ0SG8ssnaBvfyrJg3q42CCiS8ss3Nf8DewRNNqcZ4iQea9Bz2
xL23KW/2pMkSkxfTZ4J0lttfDf9II=
    </ds:SignatureValue>
    <ds:KeyInfo>
        <ds:X509Data>
            <ds:X509Certificate>
MIICTjCCAbcCBEBjZQEwDQYJKoZIhvcNAQEEBQAwbTElMAkGA1UEBhMCTEsxEDAObGNVBAG
TB1dlc3Rlcm4xEDAObGNVBACTB0NvbG9tYm8xDzANBgNVBAoTBkFwYWN0ZTEQMA4GA1UECx
MHUmFtcGFydDEXMBUGA1UEAxMOU2FtcGx1IFNlcnZpY2UwIBcNMDCwODIwMDk1NTEzWhgPM
jA2MjA1MjMwOTU1MTNaMG0xCzAJBgNVBAYTAkxLMRAwDgYDVQQIEwdXZXN0ZXJumRAwDgYD
VQQHEwdDb2xvbWJvMQ8wDQYDVQQKEWZBcGFjaGUxZDAObGNVBAsTB1JhbXBhcnQxZzAVBgN
VBAMTDlnbXhBbSBZSBTZXJ2aWNlMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCDtg6es
s2lUlyOD48/iiAlWobB0WwAQtfG4bb2KyvOE9dRF7+d/aZrHti3QWs6dtHpGkVMLgpomoq7
APEq1kQnRvduk2T6ln83JwlEpPDxH/emqeC9OdNqHZj3eoyf34JMmgShuviYDqYaK4HkRmZ
MiJl3aPeZzPl60yBWydAuwIDAQABMA0GCSqGSIb3DQEBAQUAA4GBACVcoAqNbj07+Jbm6+3
pyYagQoBpdHZLnR8EU9/CRKmUGTj5qjXqYtE+Eka6OYKBzv/dHdYlB2X3yH3YlSx10tA3+5
xl4VIjYODlgh9Bs9Tbqjl1tw0G37dLrlG97kJAVjrkfm743N9EHKftFaX4iF1tWbGxa4+vIb
bv4CaUG5s5x
            </ds:X509Certificate>
        </ds:X509Data>
    </ds:KeyInfo>
</ds:Signature>
</Assertion>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
Id="Signature-16457145">
        <ds:SignedInfo>
            <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
            <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
            <ds:Reference URI="#Id-4299997">
                <ds:Transforms>
                    <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
                </ds:Transforms>
                <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />

                <ds:DigestValue>TRY0xDKan20ArSed5LJWxarCs1I=</ds:DigestValue>
            </ds:Reference>

```

```

        <ds:Reference URI="#Timestamp-11605653">
            <ds:Transforms>
                <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            </ds:Transforms>
            <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

        <ds:DigestValue>8irDDMDaOc9U3jOylobPjxeIFac=</ds:DigestValue>
        </ds:Reference>
    </ds:SignedInfo>

    <ds:SignatureValue>08YVPZlAByo0NFB9mOLm6Vld4Fs=</ds:SignatureValue>
        <ds:KeyInfo Id="KeyId-6065773">
            <wsse:SecurityTokenReference
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd" wsu:Id="STRId-9083230">
                <wsse:Reference URI="#EncKeyId-
urn:uuid:E7186978362EF90E6A12385849980784"
ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.0#EncryptedKey"/>
            </wsse:SecurityTokenReference>
        </ds:KeyInfo>
    </ds:Signature>
</wsse:Security>
<ows6:OriginalBinding
xmlns:ows6="http://gatekeeper.service.security.n52.org"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://gatekeeper.service.security.n52.org
http://52north.org/schema/security/gatekeeper/KVP2XML.xsd">Post/XML</ow
s6:OriginalBinding>
    <wsa:To>http://localhost:8090/52n-security-gatekeeper-webapp-1.0-
SNAPSHOT/services/GK</wsa:To>

    <wsa:MessageID>urn:uuid:FDF380AB985F6CEA8D1238584998109</wsa:Message
ID>
    <wsa:Action>urn:method</wsa:Action>
</soapenv:Header>
    <soapenv:Body xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
wsu:Id="Id-4299997">
        <GetFeature xmlns="http://www.opengis.net/wfs"
xmlns:icism="urn:us:gov:ic:ism:v2"
xmlns:gml="http://www.opengis.net/gml"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:utds="http://www.opengis.net/ows-6/utds/0.3"
xmlns:build="http://www.opengis.net/citygml/building/1.0"
outputFormat="gml/3.1.1" service="WFS" version="1.1.0">
            <Query typeName="utds:Building">
        </Query>
    </GetFeature>
</soapenv:Body>
</soapenv:Envelope>

```

**Figure 23, WFS GetFeature Request submitted to PEP**

The PEP will then analyze the request. Since feature types are regarded as resources in this example, they are discovered within the request.

The request above asks for the feature type 'udts.building', which results in the authorization decision request shown in Figure 24 below. This request seeks an authorization for the action 'GetFeature' on the resource 'udts:building' by a subject with the roles 'alice' and 'admin'

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <Request xmlns="urn:oasis:names:tc:xacml:1.0:context">
      <Subject xmlns="urn:oasis:names:tc:xacml:1.0:context">
        <Attribute
AttributeId="urn:n52:authentication:subject:principal:role"
DataType="http://www.w3.org/2001/XMLSchema#string"
xmlns="urn:oasis:names:tc:xacml:1.0:context">
          <AttributeValue
xmlns="urn:oasis:names:tc:xacml:1.0:context">alice</AttributeValue>
        </Attribute>
        <Attribute
AttributeId="urn:n52:authentication:subject:principal:role"
DataType="http://www.w3.org/2001/XMLSchema#string"
xmlns="urn:oasis:names:tc:xacml:1.0:context">
          <AttributeValue
xmlns="urn:oasis:names:tc:xacml:1.0:context">admin</AttributeValue>
        </Attribute>
      </Subject>
      <Resource xmlns="urn:oasis:names:tc:xacml:1.0:context">
        <Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string"
xmlns="urn:oasis:names:tc:xacml:1.0:context">
          <AttributeValue
xmlns="urn:oasis:names:tc:xacml:1.0:context">udts:Building</AttributeVa
lue>
        </Attribute>
      </Resource>
      <Action xmlns="urn:oasis:names:tc:xacml:1.0:context">
        <Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"
xmlns="urn:oasis:names:tc:xacml:1.0:context">
          <AttributeValue
xmlns="urn:oasis:names:tc:xacml:1.0:context">GetFeature</AttributeValue>
        </Attribute>
      </Action>
    </Request>
  </soapenv:Body>
</soapenv:Envelope>
```

**Figure 24, PDP Authorization Decision Request**

The PDP will match this request against the available policies. The policy set used for the authorization is shown in Figure 25 below.

First, the policy set defines in its target on which combination of subjects, resources and actions it should be invoked. In this example the policy set is relevant for any subject accessing the resource identified by its service URL and the resource 'udts:building' with the actions 'DescribeFeatureType', 'GetCapabilities', 'GetFeature', and 'GetGmlObject'. Thus it is applicable for the given combination of the roles 'alice' and 'admin', the resource 'udts:building', and the action 'GetFeature'.

Now this policy set contains a policy which again defines its own target, which is as subject the role 'alice', any resource, and the action 'GetFeature'. If this combination is requested, the effect is defined as 'Permit'.

```
<?xml version="1.0" encoding="UTF-8"?>
<PolicySet PolicySetId="urn:conterra:names:sdi-
suite:policy:interceptor:wfs::OWS-6_Airport-WFS"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-
algorithm:first-applicable" xmlns="urn:oasis:names:tc:xacml:1.0:policy"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:policy
C:\DOKUME~1\Ifgi\Desktop\cs-xacml-schema-policy-01.xsd">
  <Description>OWS-6_Demo-PolicySet</Description>
  <Target>
    <Subjects>
      <AnySubject/>
    </Subjects>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="http://www.sdi-
suite.de/securitymanager/xacml/names/function#string-equals-ignore-
case">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">http://services.inte
ractive-instruments.de/ows6/cgi-bin/ows6airport-
wfs.exe</AttributeValue>
          <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </ResourceMatch>
      </Resource>
      <Resource>
        <ResourceMatch MatchId="http://www.sdi-
suite.de/securitymanager/xacml/names/function#string-equals-ignore-
case">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">utds:Building</Attri
buteValue>
          <ResourceAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </ResourceMatch>
      </Resource>
    </Resources>
```

```

    <Actions>
      <Action>
        <ActionMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">DescribeFeatureType<
/AttributeValue>
          <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </ActionMatch>
        </Action>
      <Action>
        <ActionMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">GetCapabilities</Att
ributeValue>
          <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </ActionMatch>
        </Action>
      <Action>
        <ActionMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">GetFeature</Attribut
eValue>
          <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </ActionMatch>
        </Action>
      <Action>
        <ActionMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">GetGmlObject</Attrib
uteValue>
          <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </ActionMatch>
        </Action>
    </Actions>
  </Target>
  <Policy PolicyId="WFS_Right"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-
algorithm:first-applicable">
    <Description>Spatial Obligation for OWS-6</Description>
    <Target>
      <Subjects>
        <Subject>
          <SubjectMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">

```

```

        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">alice</AttributeValu
e>
        <SubjectAttributeDesignator
AttributeId="urn:n52:authentication:subject:principal:role"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </SubjectMatch>
    </Subject>
</Subjects>
<Resources>
    <AnyResource/>
</Resources>
<Actions>
    <Action>
        <ActionMatch
MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">GetFeature</Attribut
eValue>
            <ActionAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
            </ActionMatch>
        </Action>
    </Actions>
</Target>
<Rule RuleId="allow_access" Effect="Permit">
    <Description>allow access</Description>
</Rule>
<Obligations>
    <Obligation ObligationId="SpatialObligation01"
FulfillOn="Permit">
        <AttributeAssignment DataType="http://www.opengis.net/ogc"
AttributeId="propertyfilter">
            <ogc:Filter xmlns:ogc="http://www.opengis.net/ogc"
xmlns:gml="http://www.opengis.net/gml"
xmlns:build="http://www.opengis.net/citygml/building/1.0">
                <ogc:Or>
                    <ogc:DWithin>

                        <ogc:PropertyName>build:lod1Solid</ogc:PropertyName>
                        <gml:Point
srsName="urn:ogc:def:crs:EPSG::4326">
                            <gml:coordinates>29.963745015416, -
90.029951432619</gml:coordinates>
                        </gml:Point>
                        <Distance uom="m">1150</Distance>
                    </ogc:DWithin>
                    <ogc:And>
                        <ogc:DWithin>

                            <ogc:PropertyName>build:lod1Solid</ogc:PropertyName>
                            <gml:Point
srsName="urn:ogc:def:crs:EPSG::4326">
                                <gml:coordinates>29.963745015416, -
90.029951432619</gml:coordinates>
                            </gml:Point>

```

```

        <Distance uom="m">2500</Distance>
      </ogc:DWithin>
    </ogc:PropertyIsEqualTo>

    <ogc:PropertyName>@icism:classification</ogc:PropertyName>
    <ogc:Literal>U</ogc:Literal>
    </ogc:PropertyIsEqualTo>
  </ogc:And>
</ogc:Or>
</ogc:Filter>
</AttributeAssignment>
</Obligation>
</Obligations>
</Policy>
</PolicySet>

```

**Figure 25, PolicySet used for authorization decision**

Finally, this policy includes an obligation which is applied whenever the policy leads to a 'Permit' decision (see obligation attribute 'FullfillOn="Permit"'). This obligation then has to be appended to the decision response to the PEP, resulting in the response shown in Figure 26 below.

```

<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <Response xmlns="urn:oasis:names:tc:xacml:1.0:context">
      <Result ResourceId="utds:Building">
        <Decision>Permit</Decision>
        <Status>
          <StatusCode
Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
        </Status>
        <Obligations xmlns="urn:oasis:names:tc:xacml:1.0:policy">
          <Obligation FulfillOn="Permit"
ObligationId="SpatialObligation01">
            <AttributeAssignment AttributeId="propertyfilter"
DataType="http://www.conterra.de/xsd/spatialauthzfilter">
              <ogc:Filter
xmlns:build="http://www.opengis.net/citygml/building/1.0"
xmlns:gml="http://www.opengis.net/gml"
xmlns:ogc="http://www.opengis.net/ogc">
                <ogc:Or>
                  <ogc:DWithin>

                    <ogc:PropertyName>build:lod1Solid</ogc:PropertyName>
                    <gml:Point
srsName="urn:ogc:def:crs:EPSG::4326">
                      <gml:coordinates>29.963745015416, -
90.029951432619</gml:coordinates>
                    </gml:Point>
                    <Distance uom="m"
xmlns="urn:oasis:names:tc:xacml:1.0:policy"
xmlns:ns1="urn:oasis:names:tc:xacml:1.0:policy">1150</Distance>

```

```

        </ogc:DWithin>
        <ogc:And>
            <ogc:DWithin>

                <ogc:PropertyName>build:lod1Solid</ogc:PropertyName>
                <gml:Point
srsName="urn:ogc:def:crs:EPSG::4326">

                    <gml:coordinates>29.963745015416, -90.029951432619</gml:coordinates>
                    </gml:Point>
                    <Distance uom="m"
xmlns="urn:oasis:names:tc:xacml:1.0:policy"
xmlns:ns2="urn:oasis:names:tc:xacml:1.0:policy">2500</Distance>
                </ogc:DWithin>
                <ogc:PropertyIsEqualTo>

                    <ogc:PropertyName>@icism:classification</ogc:PropertyName>
                    <ogc:Literal>U</ogc:Literal>
                    </ogc:PropertyIsEqualTo>
                </ogc:And>
            </ogc:Or>
        </ogc:Filter>
    </AttributeAssignment>
</Obligation>
</Obligations>
</Result>
</Response>
</soapenv:Body>
</soapenv:Envelope>

```

**Figure 26, Authorization Decision Response with Obligation (OGC Filter)**

The decision is ‘Permit’, but the obligation is included, forcing the PEP to fulfill this obligation in order to perform the permit decision. If the PEP is unable to fulfill the obligation, it is not allowed to permit the request.

The obligation in this example is of the type ‘spatialauthzfilter’. The PEP then needs the knowledge how to fulfill obligations of this type. In this case this obligation consists of a filter encoding statement, shown in Figure 26, which is then included in the service request by the PEP, resulting in the GetFeature request from the PEP to the WFS as shown in Figure 27 below.

```

<GetFeature xmlns="http://www.opengis.net/wfs"
xmlns:build="http://www.opengis.net/citygml/building/1.0"
xmlns:gml="http://www.opengis.net/gml"
xmlns:icism="urn:us:gov:ic:ism:v2"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:utds="http://www.opengis.net/ows-6/utds/0.3"
outputFormat="gml/3.1.1" service="WFS" version="1.1.0">
    <Query typeName="utds:Building">
        <ogc:Filter
xmlns:build="http://www.opengis.net/citygml/building/1.0"
xmlns:gml="http://www.opengis.net/gml"
xmlns:ogc="http://www.opengis.net/ogc">
            <ogc:Or>

```



```

        <ogc:DWithin>
          <ogc:PropertyName>build:lod1Solid</ogc:PropertyName>
          <gml:Point srsName="urn:ogc:def:crs:EPSG::4326">
            <gml:coordinates>29.963745015416, -
90.029951432619</gml:coordinates>
          </gml:Point>
          <ns1:Distance
xmlns:ns1="urn:oasis:names:tc:xacml:1.0:policy"
xmlns="urn:oasis:names:tc:xacml:1.0:policy" uom="m">1150</ns1:Distance>
          </ogc:DWithin>
        <ogc:And>
          <ogc:DWithin>

            <ogc:PropertyName>build:lod1Solid</ogc:PropertyName>
            <gml:Point srsName="urn:ogc:def:crs:EPSG::4326">
              <gml:coordinates>29.963745015416, -
90.029951432619</gml:coordinates>
            </gml:Point>
            <ns2:Distance
xmlns:ns2="urn:oasis:names:tc:xacml:1.0:policy"
xmlns="urn:oasis:names:tc:xacml:1.0:policy" uom="m">2500</ns2:Distance>
            </ogc:DWithin>
            <ogc:PropertyIsEqualTo>

              <ogc:PropertyName>@icism:classification</ogc:PropertyName>
              <ogc:Literal>U</ogc:Literal>
            </ogc:PropertyIsEqualTo>
          </ogc:And>
        </ogc:Or>
      </ogc:Filter>
    </Query>
  </GetFeature>

```

**Figure 27, WFS GetFeature Request with Filter Encoding**

This request now includes a filter statement with spatial and attribute restrictions. Thus, the WFS will only submit authorized features.

## 13 OWS-6 Use Cases

Within OWS-6, a strong focus was put on security aspects between different security domains. The OWS-6 RFQ provided three distinct use cases as a basis for development and testing of security architectures and deployment for OGC web services. These three use cases are described in the following three clauses.

### 13.1 OWS-6 WS-Security Deployment

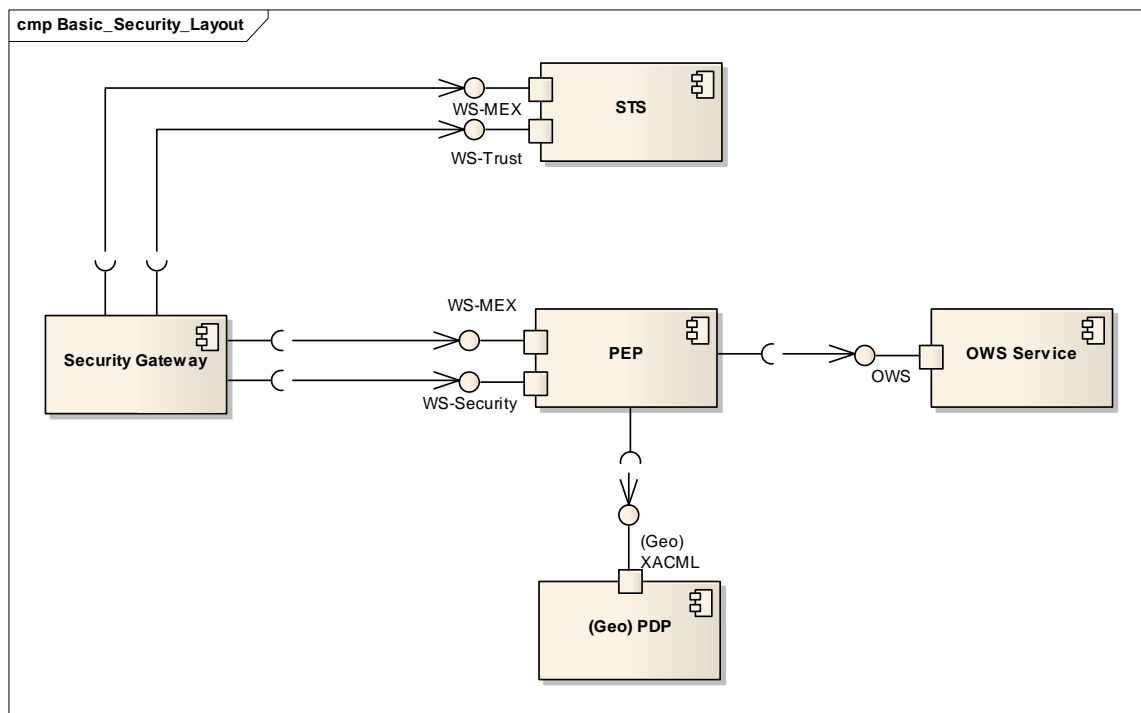
The OWS-6 SOAP use cases consist of a PEP which is used by a client as service endpoint. This PEP exposes a WS-MEX [15] interface, providing a WS-Policy [16] document that informs about the security preconditions, expressed in WS-SecurityPolicy [18]. These preconditions indicate the requirement for a SAML identity token, issued by

a Security Token Service (STS) following the WS-Trust standard [19], to be submitted within the request.

The client reacts by requesting the required token at the STS. Therefore, it first uses the WS-MEX interface to request the WS-Policy description of the STS, in order to discover the STS' preconditions for issuing the required token. In the OWS-6 use cases, this WS-Policy document requires the user's credentials to be submitted using the WS-Security Username Token profile.

Consequently, the client submits a "RequestSecurityTokenRequest" to the STS, which is answered with the according SAML token. Once the client receives the token, it can be submitted within the service request as part of the request header, using the WS-Security SAML profile.[17].

The PEP receives the message, checks if all requirement expressed by the WS-Policy document are fulfilled, delegates the access decision to the PDP, and enforces this decision. In case of a 'permit' decision the request is passed on to the OWS. This is visualized in Figure 28.

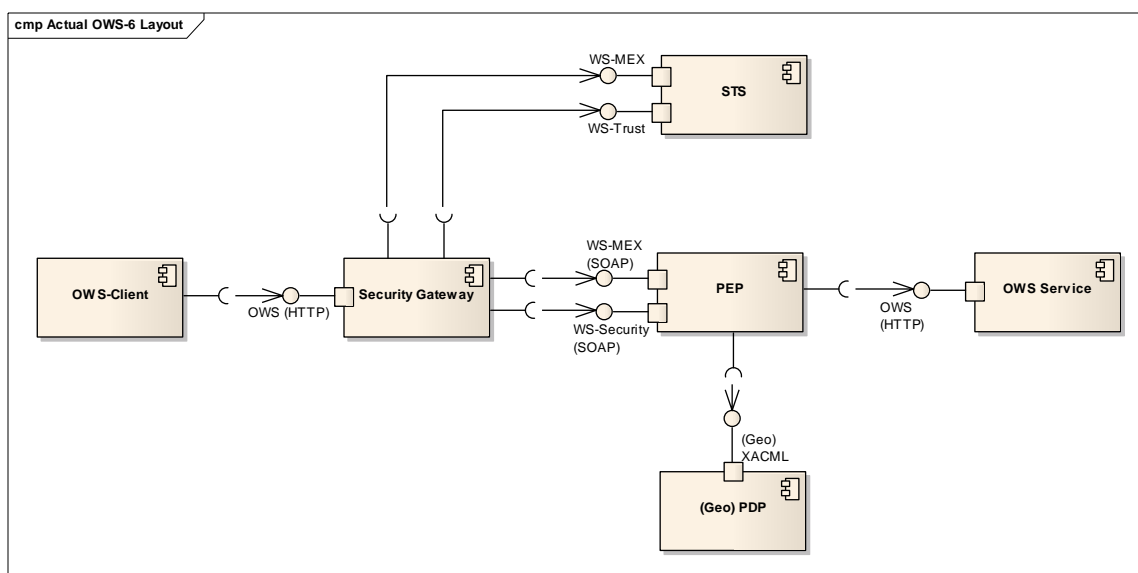


**Figure 28: Basic Security Deployment**

This figure assumes the existence of SOAP OWS services and OWS clients capable of supporting WS-MEX, WS-Trust and WS-Security. All those assumptions are not fulfilled in the OWS-6 use cases. In contrast, the OWS services used do not support SOAP rather than HTTP-GET/KVP and HTTP-POST/XML. Thus, the PEP additionally acted as a SOAP wrapper for the OWS, following the SOAP wrapping approach as described in OGC document 07-158 [5].

As shown in Figure 29, the OWS client uses a security gateway component which exposes a standard OWS interface to the client based on HTTP-GET and –POST. The gateway has a GUI allowing a user to specify a URL of a secured service (the PEP URL). Once the URL is specified, the gateway requests the WS-Policy document and realizes the need for an authentication. Thus, it provides a login dialog on the GUI, requesting the user credentials. These credentials are used to request a SAML token at the STS.

After receiving the SAML token, the gateway receives OWS HTTP-GET and –POST requests, transforms them into SOAP including the SAML token in the SOAP header, and submits them to the PEP. Responses from the PEP, which are also SOAP responses, are then transformed back into HTTP and forwarded to the client.



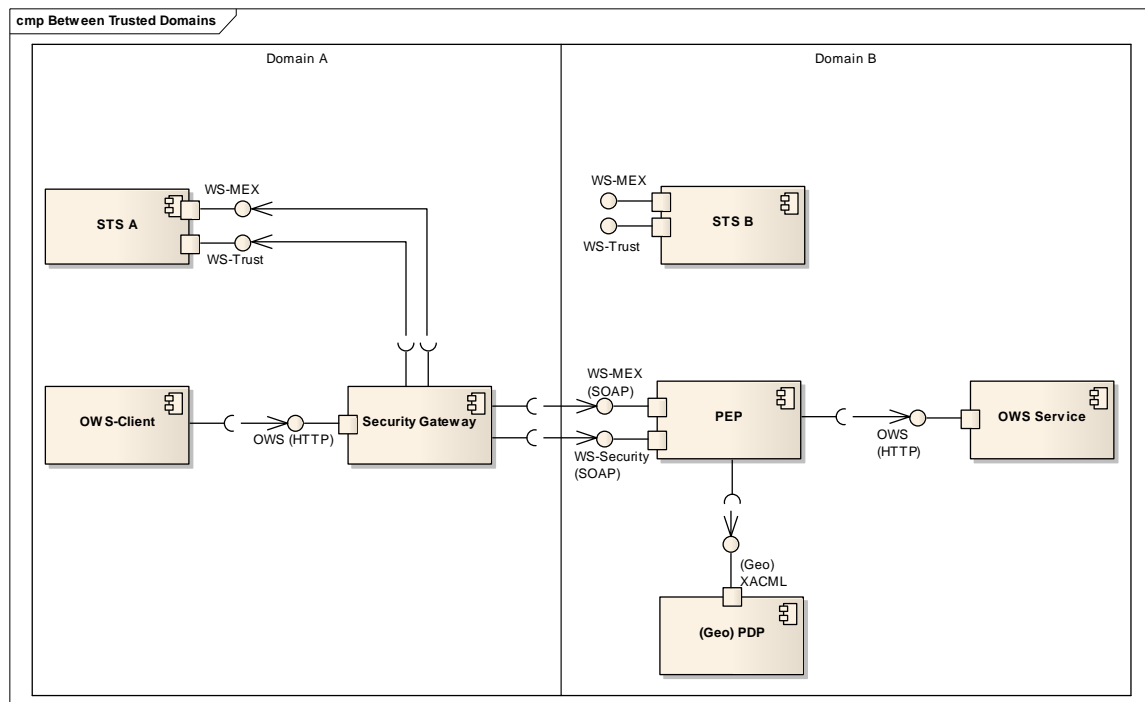
**Figure 29: Actual OWS-6 Deployment**

### 13.2 Within One Security Domain

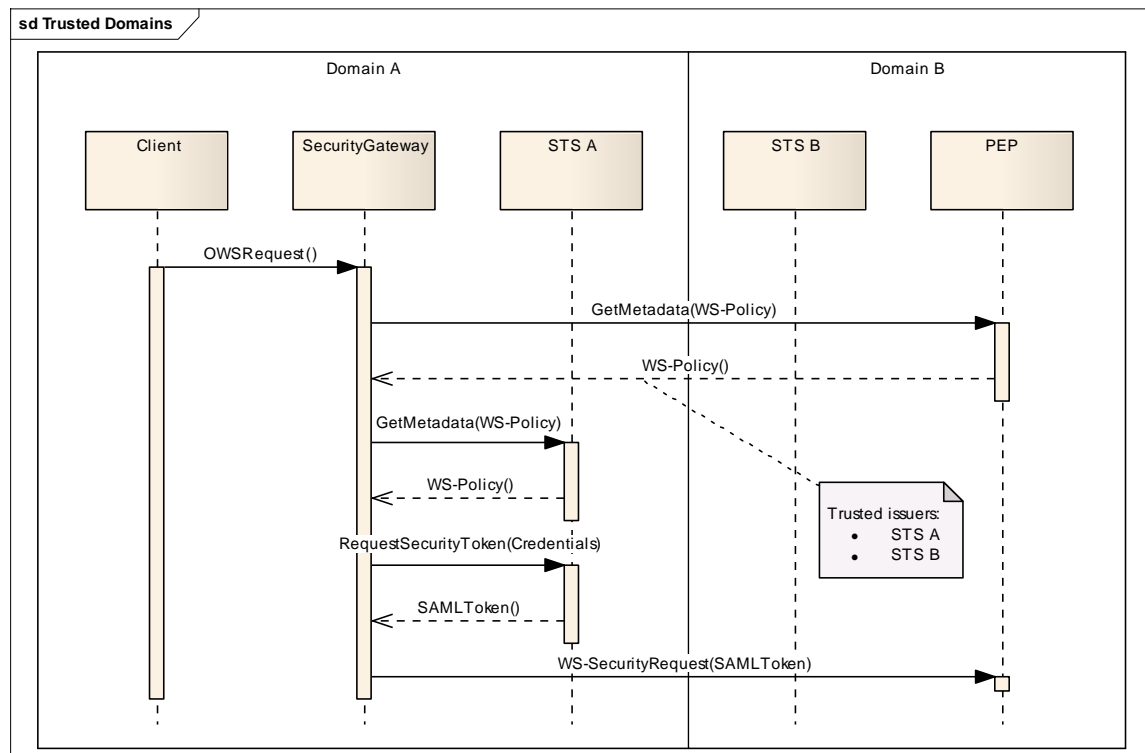
As long as the OWS and the OWS client are both within the same security domain, there are no special requirements beyond the architecture described in the previous section. The PEP (on behalf of the OWS) communicates with the STS from its own security domain as token issuer. The STS is trusted by the client since it is in the same security domain, so it can directly request the required token from this STS and use it for authentication.

### 13.3 Between Trusted Security Domains

If the communication partners are from different security domains, a trust relationship between these domains is necessary. If those domains trust each other directly, the secured service refers in its WS-Policy document to both, its own and the other domain's STS, accepting identity tokens from both issuers. See Figure 30 and Figure 31 below.



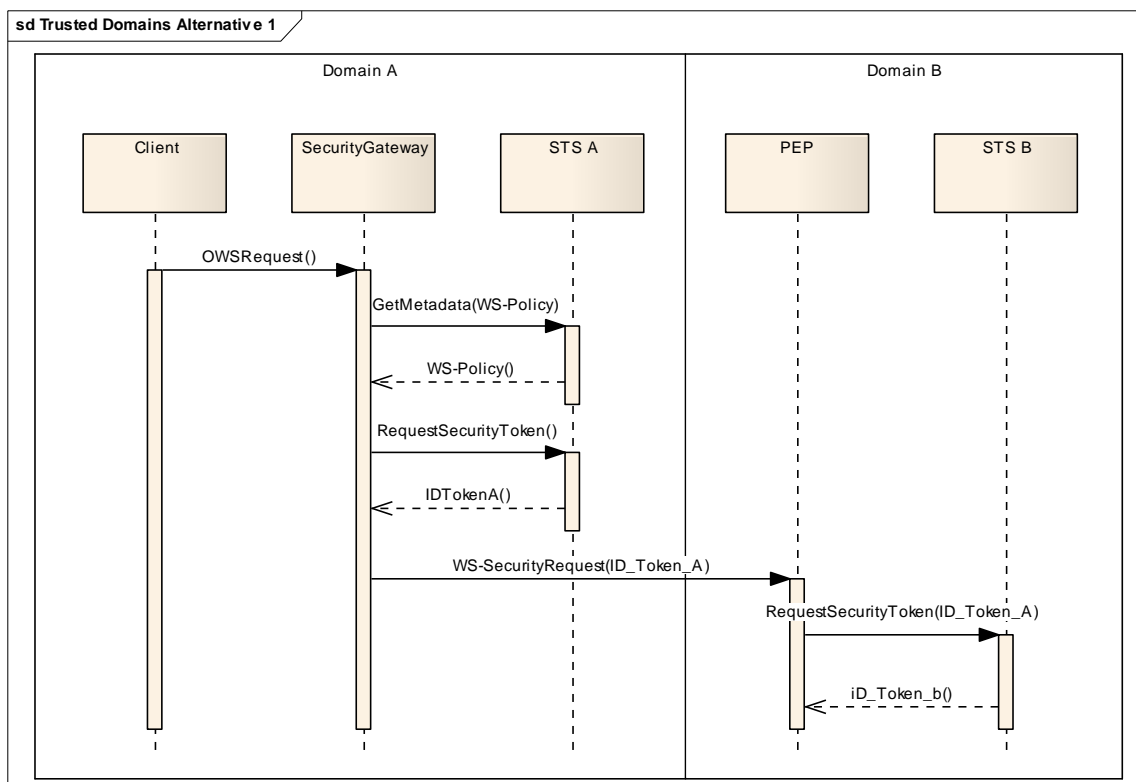
**Figure 30: Security between Trusted Security Domains**



**Figure 31: Interactions between Trusted Domains**

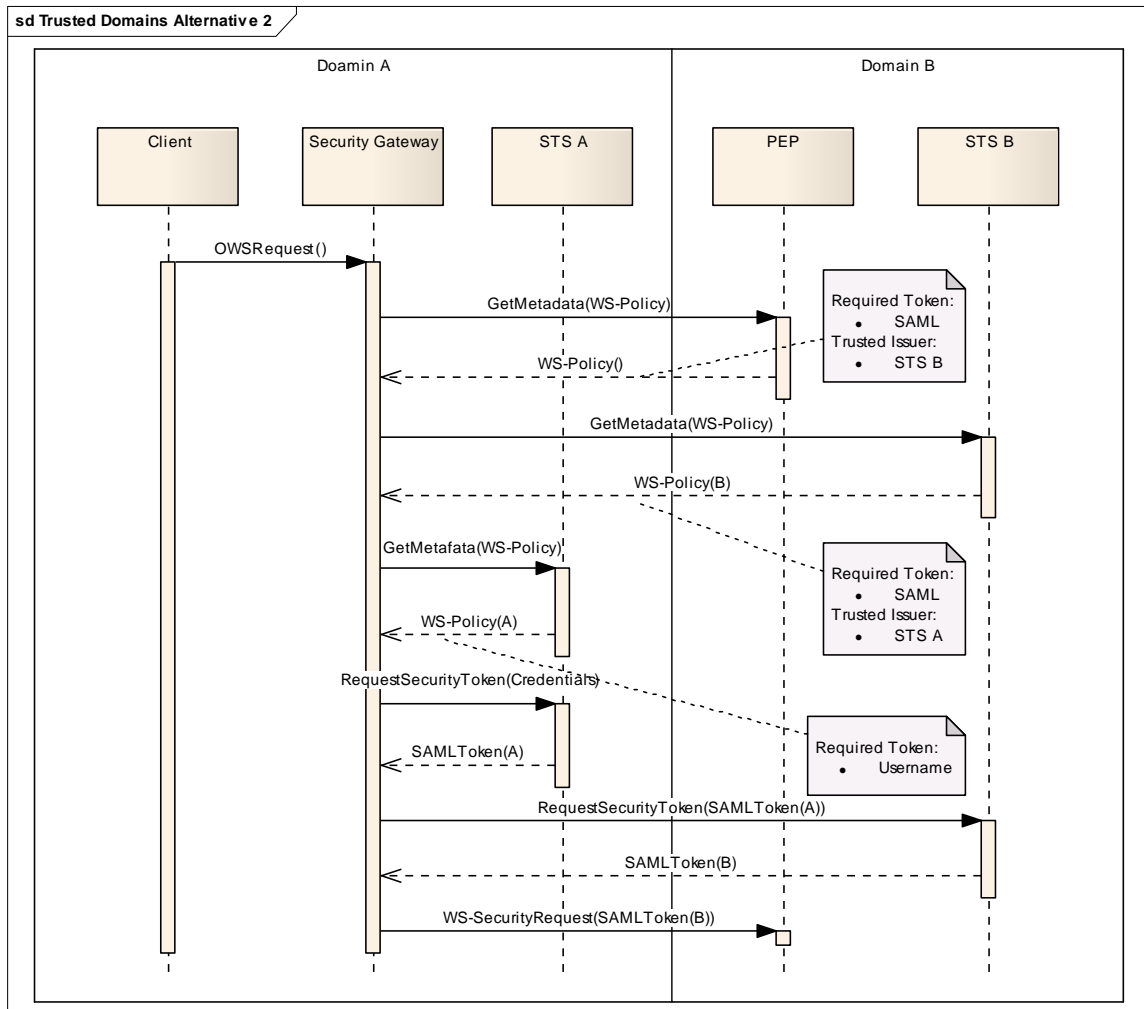
Alternatively, the service would not even have to trust the remote STS directly. There are two other options:

1. The service uses its own STS to convert the remote identity token into a token issued by the STS in Domain B, as shown in Figure 32.



**Figure 32: Trusted Domains, alternative 1**

2. The PEP only refers to its own STS as trusted token issuer. The client accesses the STS from the provider's domain, which accepts in its WS-Policy document identity tokens from the client's STS. That would force the client to first authenticate at the own STS, let this identity token be converted by the provider's STS, and use the converted token to access the secured service, as shown in Figure 33.

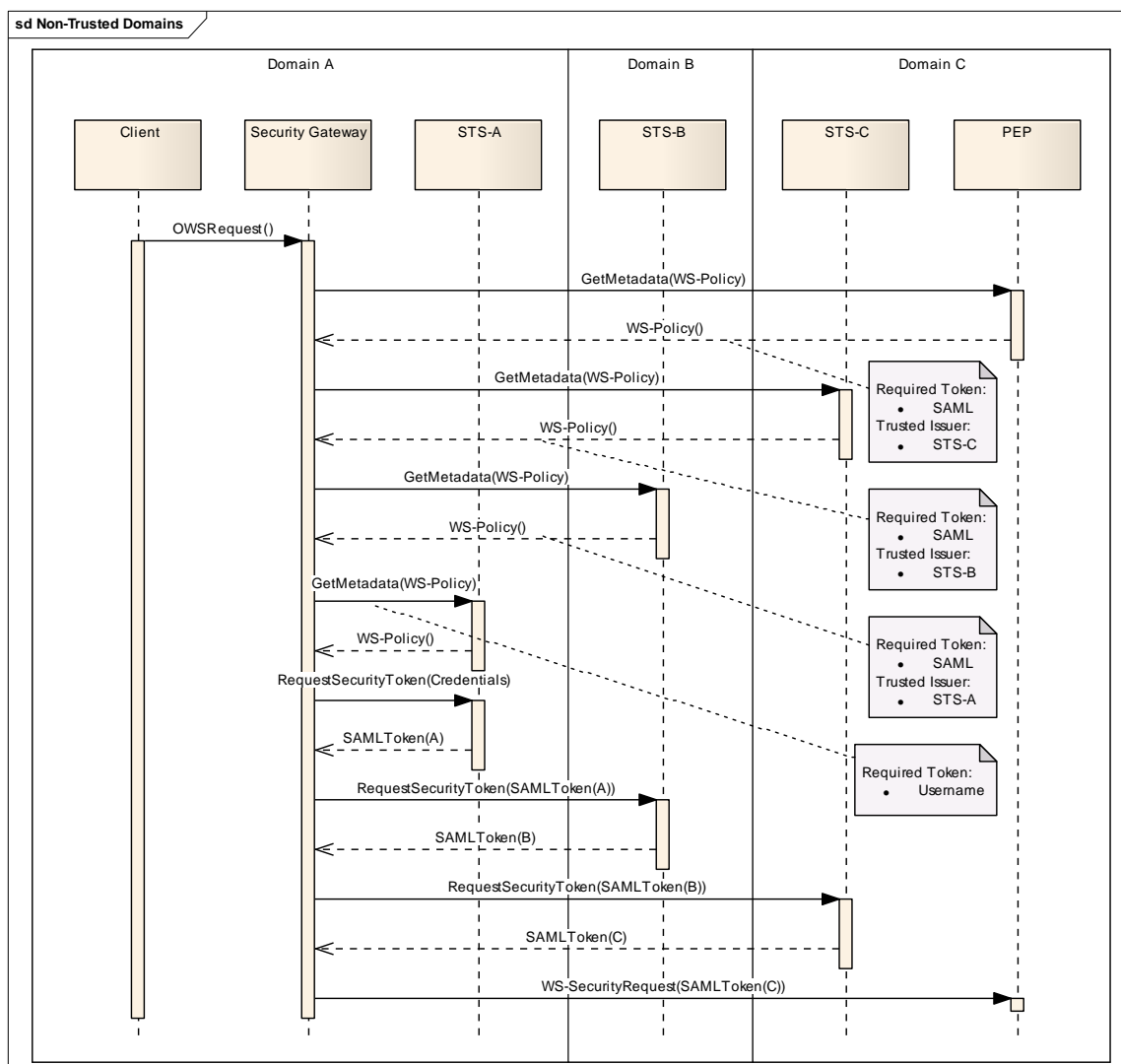


**Figure 33: Trusted Domains, alternative 2**

#### 13.4 Between Un-trusted Security Domains (Trust Establishment)

Secure communication between domains not being within a trust relationship requires trust establishment. Thus, a chain of trust relationships has to be created which provides a transitive trust relationship between the communication partners.

Figure 34 shows three security domains, with a trust relationships between Domain A and Domain B as well as between Domain B and Domain C, but no direct trust relationship between Domain A and Domain C.

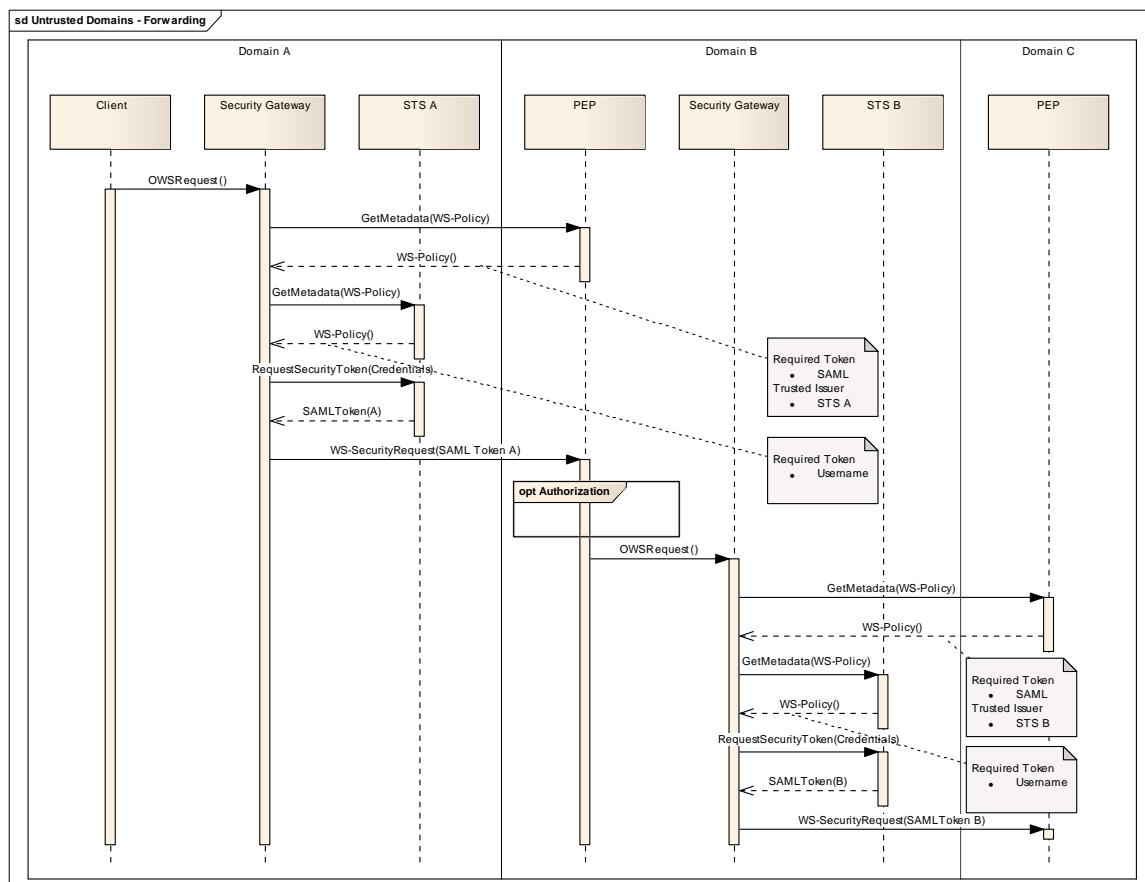


**Figure 34: Trust Establishment between Non-Trusted Domains**

### 13.5 Between Un-trusted Security Domains (Forwarding)

Certain circumstances concerning network security aspects of the sponsors, the use case 'Between Un-trusted Security Domains' could not be realized as described in section 13.4, since no direct access between un-trusted networks can be accepted. Thus, Figure 35 shows an alternative which – strictly speaking – is not real communication between un-trusted domains but rather is a cascaded communication between trusted domains, based on the assumption that there are trust relationships between Domain A and B and between Domain B and C, but not between Domain A and C.

Since the Security Gateway offers a pure OWS interface, it is possible to protect this Gateway in Domain B with a PEP once more. Thus, Domain C only has to allow access to its OWS to Domain B, acting on behalf of Domain A.



**Figure 35: Un-trusted Domains - Forwarding**

## 14 RFQ Use Cases

The sections that follow provide the use cases contained in the Request for Quotation (RFQ) that served as the basis for developing the actual security architecture and implementations produced in the GPW thread as part of the OWS-6 testbed.



## 14.1 Within One Security Domain

This use case describes the requirements to be exercised within a single security domain.

### Use Case #1: Single Trusted Domain Security

Use Case Id:	GPW #1	Use Case Name:	Processing IC-ISM classified data within a single trusted domain
Use Case Domain:	OWS-6 GPW for Security Domains		Status: Draft (RFQ)
Use Case Description:	A source in a trusted domain maintains and serves IC-ISM classified data to authorized users within the same trusted domain		
Actors (Initiators):	User1: owner/custodian of IC-ISM classified data User2: trusted user requesting data	Actors (Receivers)	User2: trusted user receiving data
Pre-Conditions: - User1 manages secured data for publication and access. - IC-ISM classified data is available for query and retrieval by authorized users - User2 has appropriate credentials to request Desired Data.		Post-Conditions: - User2 retrieves and displays Requested Data	
System Components - Authentication Service (STS): authenticate users who wish to retrieve data from a trusted source - Authorization Service (PDP): Grant permission for requesting user to access resources based upon access rights - Policy Store (PAP): maintains policies to determine who may retrieve data based on identity and IC-ISM classification of data. - Gatekeeper (PEP): provides interface between the client and the OGC web service to control access to data based on authentication and authorization results - WFS: serves feature data with controlled access - CS-W: repository of service offerings, data and metadata for IC-ISM classified data			

Use Case Id:	GPW #1	Use Case Name:	Processing IC-ISM classified data within a single trusted domain
<p>Basic Course of Action</p> <ol style="list-style-type: none"> <li>1. User2 enters his credentials to the STS, which verifies his identity and returns an Identity Token to User2 to certify his identity</li> <li>2. User2 queries CS/W for Desired Data using the Identity Token provided by the STS.</li> <li>3. CS/W returns Query Result describing data available to User2.</li> <li>4. User2 examines the Query Result to determine that the Desired Data exists on WFS. The Query Result also indicates that Requested Data is IC-ISM coded which requires that User2 have appropriate authorization to retrieve the Requested Data.</li> <li>5. User2 prepares and issues request to retrieve the data from WFS via Gatekeeper (PEP) using his Identity Token.</li> <li>6. PEP contacts PDP providing User2 Identity Token and metadata about the Requested Data to obtain authorization for User2 to retrieve Requested Data.</li> <li>7. PDP contacts PAP to determine policies for retrieval of the Requested Data by User2</li> <li>8. PAP returns result to PDP to determine is User2 is authorized to retrieve Requested Data.</li> <li>9. PDP notifies PEP that User2 is authorized to retrieve the Requested Data.</li> <li>10. PEP submits request to WFS for the Requested Data.</li> <li>11. WFS returns Requested Data to User2 via the PEP.</li> </ol>			

## 14.2 Between Trusted Security Domains

This use case describes the requirements to be exercised between two security domains with an established trust relationship.

### Use Case #2: Trusted-to-Trusted Domain Security

Use Case Id:	GPW #2	Use Case Name:	Processing IC-ISM classified data across two trusted domains
Use Case Domain:	OWS-6 GPW for Security Domains	Status:	Draft 2008-07-18
Use Case Description:	A source in a trusted domain maintains and serves IC-ISM classified data to authorized users in a separate trusted domain		
Actors (Initiators):	User1(domain1): owner/custodian of IC-ISM classified data  User2 (domain2): trusted user requesting data	Actors (Receivers)	User2 (domain2): trusted user receiving data
Pre-Conditions:  - User1 manages secured data for publication and access.  - IC-ISM classified data is available for query and retrieval by authorized users  - User2 has appropriate credentials to request needed data.		Post-Conditions:  - User2 retrieves and displays requested data from Domain1	
System Components  - Authentication Service (Domain1) (STS1): authenticate users for Domain1 trusted sources  - Authentication Service (Domain2) (STS2): authenticate users who wish to retrieve data from a Domain2 trusted source  - Authorization Service (Domain1) (PDP1): Grant permission for requesting user to access resources based upon access rights  - Authorization Service (Domain2) (PDP2): Grant permission for requesting user to access resources based upon access rights  - Policy Store (Domain1) (PAP1): maintains policies to determine who may retrieve data based on identity and IC-ISM classification of data.  - Gatekeeper (Domain1) (PEP1): provides interface between the client and the OGC web service to control access to data based on authentication and authorization results for Domain1  - WFS: serves feature data with controlled access (Domain1)  - CS-W: repository of service offerings, data and metadata for IC-ISM classified data (Domain1)			

Use Case Id:	GPW #2	Use Case Name:	Processing IC-ISM classified data across two trusted domains
<ol style="list-style-type: none"> <li>1. User2 enters his credentials to the STS2, which verifies his identity and returns an Identity Token to User2 to certify his identity</li> <li>2. User2 queries CS/W for Desired Data using the Identity Token provided by the STS2.</li> <li>3. CS/W returns Query Result describing data available to User2.</li> <li>4. User2 examines the Query Result to determine that the Desired Data exists on WFS. The Query Result also indicates that Requested Data is IC-ISM coded which requires that User2 have appropriate authorization to retrieve the Requested Data.</li> <li>5. User2 prepares and issues request to retrieve the data from WFS via Gatekeeper (PEP1) using his Identity Token.</li> <li>6. PEP1 contacts PDP1 providing User2 Identity Token and metadata about the Requested Data to obtain authorization for User2 to retrieve Requested Data.</li> <li>7. PDP1 contacts PAP1 to determine policies for retrieval of the Requested Data by User2</li> <li>8. PAP1 returns result to PDP1 to determine is User2 is authorized to retrieve Requested Data.</li> <li>9. PDP1 notifies PEP1 that User2 is authorized to retrieve the Requested Data.</li> <li>10. PEP1 submits request to WFS for the Requested Data.</li> <li>11. WFS returns Requested Data to User2 via the PEP1.</li> </ol>			

### 14.3 Between Un-trusted Security Domains (Trust Establishment)

This use case describes the requirements to establish and exercise trust between two security domains.

#### Use Case #3: Trusted to Temporarily-Trusted Security Domain

Use Case Id:	GPW #3	Use Case Name:	Processing IC-ISM classified data between a trusted domain and a temporarily trusted domain
Use Case Domain:	OWS-6 GPW for Secure Domains	Status:	Draft 2008-07-18
Use Case Description:	A source in a trusted domain (domain1) maintains and serves IC-ISM classified data to authorized users in a temporarily trusted domain (domain3)		
Actors (Initiators):	User1(domain1): owner/custodian of IC-ISM classified data  User3 (domain3): temporarily-trusted user requesting data	Actors (Receivers)	User3 retrieves and displays requested data from Domain1
Pre-Conditions:	- User1 manages secured data for publication and access. - IC-ISM classified data is available for query and retrieval by authorized users - User3 has appropriate credentials to request needed data.		
	Post-Conditions: - User3 retrieves and displays requested data		

Use Case Id:	GPW #3	Use Case Name:	Processing IC-ISM classified data between a trusted domain and a temporarily trusted domain
<p>System Components</p> <ul style="list-style-type: none"> <li>- Authentication Service (Domain1) (STS1): authenticate users for Domain1 trusted sources</li> <li>- Authentication Service (Domain2) (STS3): authenticate users who wish to retrieve data from a Domain3 trusted source</li> <li>- Authorization Service (Domain1) (PDP1): Grant permission for requesting user to access resources based upon access rights</li> <li>- Authorization Service (Domain2) (PDP3): Grant permission for requesting user to access resources from Domain3 based upon access rights</li> <li>- Policy Store (Domain1) (PAP1): maintains policies to determine who may retrieve data based on identity and IC-ISM classification of data.</li> <li>- Gatekeeper (Domain1) (PEP1): provides interface between the client and the OGC web service to control access to data based on authentication and authorization results for Domain1</li> <li>- WFS: serves feature data with controlled access (Domain1)</li> <li>CS-W: repository of service offerings, data and metadata for IC-ISM classified data (Domain1)</li> </ul>			
<p>Basic Course of Action</p> <ol style="list-style-type: none"> <li>1. User3 enters his credentials to the STS3, which verifies his identity and returns an Identity Token to User3 to certify his identity</li> <li>2. User3 queries CS/W for Desired Data using the Identity Token provided by the STS3.</li> <li>3. CS/W returns Query Result describing data available to User3.</li> <li>4. User3 examines the Query Result to determine that the Desired Data exists on WFS. The Query Result also indicates that Requested Data is IC-ISM coded which requires that User3 have appropriate authorization to retrieve the Requested Data.</li> <li>5. User3 prepares and issues request to retrieve the data from WFS via Gatekeeper (PEP1) using his Identity Token.</li> <li>6. PEP1 contacts PDP1 providing User2 Identity Token and metadata about the Requested Data to obtain authorization for User3 to retrieve Requested Data.</li> <li>7. PDP1 contacts PAP1 to determine policies for retrieval of the Requested Data by User3</li> <li>8. PAP1 returns result to PDP1 to determine is User3 is authorized to retrieve Requested Data.</li> <li>9. PDP1 notifies PEP1 that User3 is authorized to retrieve the Requested Data.</li> <li>10. PEP1 submits request to WFS for the Requested Data.</li> <li>11. WFS returns Requested Data to User3 via the PEP1.</li> </ol>			

## 15 Future Work and Unsolved Issues

During the work on OWS-6 various issues have been recognized but could not be solved during this testbed. These issues are listed within this section, and future work items are

described. Some of those tie into well known issues at the OGC, such as the problematic nature of the capabilities document.

## **15.1 Security Metadata**

OGC services are typically described by metadata, indicating, among others, the type of a service, its content and its URL. The information about the service type and its URL enable a client to access this service.

### **15.1.1 Technical Metadata**

The technical part of service metadata includes binding information. If this service is protected, then the PEP URL should be published not the URL of the service itself. But publishing only the URL is not sufficient in this case. Metadata should indicate that this service is protected, and which security requirements are imposed by this service or at least where information on those security requirements can be found. This is not yet covered in the current metadata standards, breaking the publish–find–bind principle due to the gap between find and bind.

An approach could be to include a typed link to an information resource in an ‘access constraints’ element of the service metadata, such as a WS-Policy document. An alternative would be to indicate within the metadata, that the service supports WS-MetadataExchange (WS-MEX) [15]. Together with the service URL, WS-MEX specifies how a WS-Policy document can be requested from the service.

### **15.1.2 Content Metadata**

Besides technical information, service metadata also provides information on the content the service provides. If this service is access controlled, different users could potentially be authorized for different content. Thus, the service metadata should be able to reflect this distinction. It would be desirable that a catalog service only provides information according to the actual user’s access rights. But this would imply that the catalog service itself is also access controlled, and that it is informed about all access rights defined for the described service. This seems unrealistic, unless the catalog would be able to access the service’s security system remotely and request the access rights for the actual user directly. Since a catalog provides metadata for a large number of services, requesting access rights for each secured service during a search would be practically impossible.

Moreover, any of those approaches would break the principle of uniform metadata and lead to user-specific metadata, corresponding to a user’s access rights.

An alternative would be to only provide a minimum set of information which is permitted for each user. This set may potentially be empty. This approach would prevent users with extended permissions from finding content in the catalog to which they are not authorized.

There seems to be no obvious solution to that problem, so further investigation on this issue is recommended.

### 15.1.3 Capabilities

Service capabilities also provide binding- and content-related information similar to metadata provided by a catalog. Thus, similar problems as mentioned in the previous two sections may also occur.

The main difference is that these capabilities are directly requested from the PEP, and thus the content of the capabilities can be controlled by the service's security system.

A description or a reference to a description of security requirements should be included with the binding information. Binding information with security requirements could be provided in the same way as described in section 15.1.1. According to the current standards, the process of requesting capabilities in order to get information on binding preconditions seems to be in the wrong order, since a GetCapabilities request already presumes a binding to the service.

This exposes a security weakness in the OGC service model, having the capabilities document include content-related information as well as binding-related information simultaneously. Perhaps a solution would be to separate this information into different documents, such as providing a WSDL description [13] for binding information, a WS-Policy description [16] for security requirements, and a capabilities document for content-related information only. All of this information could be provided via WS-MetadataExchange [15], which provides both SOAP and RESTful interfaces to request metadata documents and thus would be widely applicable.

Meantime, a compromise for legacy services would be to allow a GetCapabilities request independently from any security preconditions, but in case of unfulfilled security preconditions answer the request as an empty capabilities document except for the access constraints information.

## 15.2 Security Error Messages

Applying access control to a service may result in additional exceptions, compared to an unprotected service. These exceptions may be classified as follows:

- Security requirements are not fulfilled
- Security information (such as authentication token) is invalid
- Access is denied

Usually exceptions are handled by error messages, potentially indicating the reason for this exception. However, from a security point of view error messages carry the risk of information leakage. This is especially relevant for an 'access denied' response, since such a response already reveals the existence of the requested resource.

No single answer exists to address this issue; it should be decided on a case-by-case basis as part of the security analysis for a particular access control system. Accepted behavior may stretch from 'access denied' messages to generating artificial HTTP error responses,



or even pretending that the service URL does not exist. Any of those behaviors may be possible.

### 15.3 Blocking or Filtering

Due to the non-atomic nature of geospatial data, where subsets of datasets are often meaningful (such as spatially restricted subsets of the data being served by an OGC service), and access control may lead to a situation where parts of the requested data are authorized for a certain user, while other parts are not.

This situation could be handled in one of two different ways:

- Block the request
- Filter the result

Both options have relevance. In the filter case, the system should decide whether the user shall be informed that a filtering has occurred and thus the result is incomplete, or if this fact shall be hidden. Once again the problem of information leakage is relevant, since the information about the incompleteness of the result implies the existence of additional data.

Another question is whether a user should be able to define if data filtering is acceptable within a request or if the whole request should be blocked in case a subset of the requested data is unauthorized. Information leakage can occur here as well. For example, suppose a user first accepts filtering and then repeats the same request without filtering. If the second request results in an ‘access denied’ response the user learns that more data matching the first request is available but not provided.

### 15.4 Standardization of Obligations

While the use of obligations as described in section 12.4.2 is consistent with the XACML standard, the content of obligations is not yet standardized. This results in a dependency between the definition of obligations in the policies and their enforcement in the PEP. As long as there is no common agreement on how to formulate obligations for OGC services and how to interpret them, there will be a tight coupling between PEP and PDP implementations and associated policies.

As long as there is no exchange of policies among authorization systems, the tight coupling between PEP, PDP and their policies should not pose a problem. However, in the future it would be desirable to provide interoperability for the use of obligations.

The obligation approach followed in this testbed uses filter statements according to the OGC Filter Encoding standard [11], which can be a valid approach for all service types accepting filter encoding in their requests, such as WFS. For service types where filter encoding is not applicable, such as WMS, more service-specific obligation formats would be needed.

## 15.5 Architecture of the Access Control System

The XACML specification version 2 presents an abstract architecture for the access control system which differs significantly from that presented in version 1 of the specification. The difference in between these abstract architectures has significant impacts on any concrete application of the XACML access control system to OGC web services, including:

- the potential of OGC implementors developing separate but interoperable components of an access control system,
- the message exchange within a modular implementation of the abstract architecture.

These differences potentially impact on the design, efficiency, and scalability of the resulting system and therefore seem worthy of further study.

## 15.6 Altering Results

The security approach taken within this testbed handles client requests for unauthorized elements by silently altering the result. Such an approach is strictly prohibited by OGC Web Services Common Standard which states, in section 11.7 of version 1.1.0c1, *“Upon receiving a valid operation request, the service shall send a response corresponding exactly to the request...”*

To resolve this conflict, it might become necessary to introduce a separate request parameter, indicating if the client prefers a ‘best effort’ response, including only those elements being permitted, or an ‘all or nothing’ approach, resulting in either complete responses or a denial message if the request cannot be completely fulfilled.

Similarly, it would be worth looking into out of band comments on the return value, to be able to add the comment that the result has possibly been altered.

## 15.7 Performance

Different approaches to the same security goals, using GeoXACML, XACML with obligations, or other technologies, have different impacts on the system performance. Even different formulations of policies resulting in the same behavior of an access control system can have performance impacts. During this testbed performance evaluation was out of scope, and due to the small data volumes such an analysis would not have yielded meaningful results.

Since performance is an important factor for all service providers, future work to evaluate performance aspects for different approaches, indicating critical factors influencing the performance for each approach would be meaningful.

## **15.8 OGC Service Types**

The access control issues probably differ for different kinds of OGC web services and therefore it would be worth investigating all these issues, case by case, for the different service types.