Open Geospatial Consortium

Approved Date: 2013-09-25

Posted Date: 2013-11-06

Reference number of this OGC® document: 13-099

External identifier of this OGC® document: http://www.opengis.net/doc/dp/geoxacml-xacml-paws

Category: OGC® Publicly Available Discussion Paper

Editors: Jan Herrmann
Andreas Matheus

OGC GeoXACML and XACML Policy Administration Web Service (PAWS)

Attention

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

But it is important to notice that this OGC publication represents the opinions of the authors regarding the functionality and interfaces of a PAWS.

Warning

This document is not an OGC Standard. This document is an OGC Discussion Paper and is therefore not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, an OGC Discussion Paper should not be referenced as required or mandatory technology in procurements.

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

OGC 13-099

License Agreement

Contents

Page

Figures

Page

Tables

Page

Abstract

This specification defines the interfaces of the OGC (Geo)XACML Policy Administration Web Service (OGC (Geo)XACML PAWS or simply PAWS in the following) that supports the creation, modification, exchange, analysis, testing, transformation, encrypting and signing of XACML and GeoXACML encoded access control policies.

This draft specification was prepared as a deliverable for the OGC Web Services, Phase 9 (OWS-9) initiative of the OGC Interoperability Program. This document presents the results of the work within add-on project of the OWS-9 Security and Services Interoperability (SSI) thread.

Please note that currently the document only contains the definition of the mandatory operations i.e. the basic conformance class. The writing of the sections describing the optional operations is still a to do. These sections need to define the following operations:

- AnalyzePolicyElement operation
- OptimizePolicyElement operation
- TransformPolicyElement operation
- TestPolicyElement operation
- EncryptPolicy operation
- SignPolicy operation

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

Keywords

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

ogcdoc, OGC document, standard, GeoXACML, XACML, access control, Policy Administration, Web Service

Submitters

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

Name	Affiliation	Contact
Jan Herrmann	Secure Dimensions	jh@secure-dimensions.de
Andreas Matheus	University of the Bundeswehr	andreas.matheus@unibw.de

Introduction

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

The challenge that arises when using GeoXACML and XACML – further (Geo)XACML – based access control systems is to provide suitable administration systems that support the sound administration of the emerging complex access control policies. The introduction of (Geo)XACML based access control systems can only be successful if the (Geo)XACML encoded policies do describe the to be enforced access rights. To achieve this objective a powerful policy administration service is required.

This specification defines the interfaces of the OGC (Geo)XACML Policy Administration Web Service (OGC (Geo)XACML PAWS or simply PAWS in the following) that supports the creation, modification, exchange, analysis, testing, transformation, encrypting and signing of XACML and GeoXACML encoded access control policies.

Note: The choice of defining a dedicated web service interface is motivated by the special characteristics of the operations on hierarchical XACML policy tree elements. These e.g. do not allow leveraging the OGC Web Feature service specification. Further using the generic OGC Web Processing Service (WPS) interface as a already standardized wrapper is considered the wrong approach as the design and features of the WPS interface do not bring benefits to the XACML policy administration scenario but do add significant implementation complexity for developers.

Figure 1 gives an example how a PAWS instance could be deployed in a (Geo)XACML based access control system.

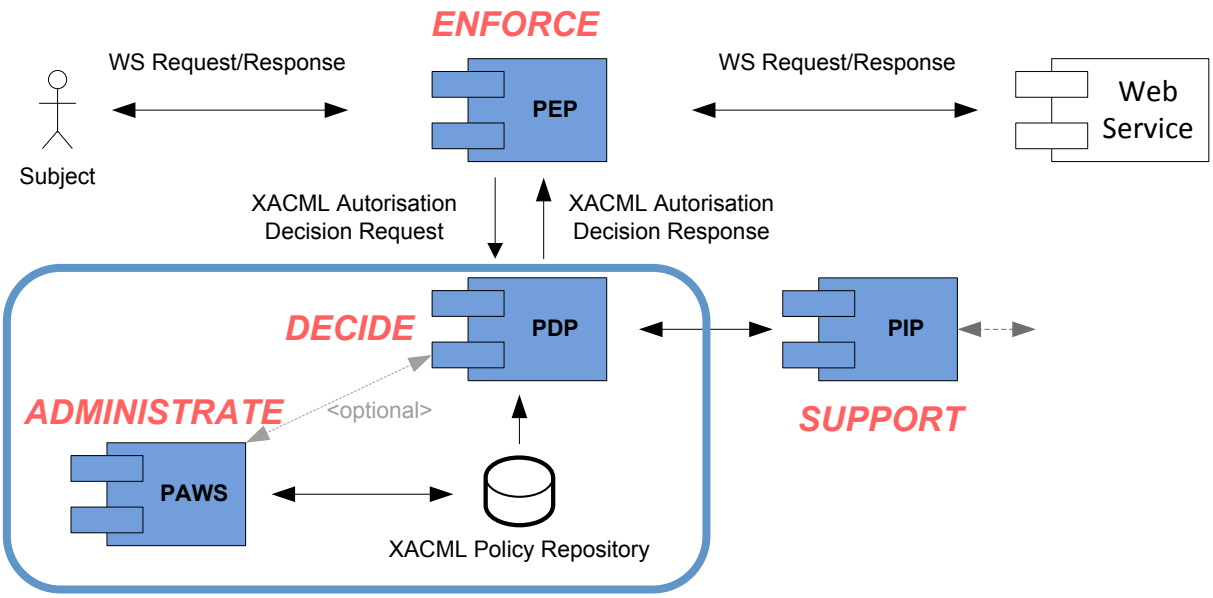

Figure 1: PAWS deployment within an XACML based access control system.

OGC (Geo)XACML Policy Administration Web Service (PAWS)

1 Scope

This document specifies the behavior of a service that provides access to, analysis of and transactions on XACML v3.0 as well as on GeoXACML v3.0[1] encoded access control policies. The service interface is independent of the underlying storage model and therefore constitutes an abstraction layer towards the storage layer.

The design of the PAWS interface has been arranged to support sufficiently fine-grained read, write, delete, etc. access to (Geo)XACML policy trees. On the other side, the interface was designed in a way to be not too fine-grained. Opposed to operations on XML node level (which would make the definition of rich administrative rights unmanageable) it was decided to define slightly more coarse-grained operations on (Geo)XACML policy trees that take its logical structure and associated semantics into account. This design choice is crucial in order to allow for a manageable definition of rich administrative rights, controlling the allowed interactions on (Geo)XACML policy trees through PAWS instances.

This specification defines the operations listed in table 1.

Mandatory	Optional
GetCapabilities	AnalyzePolicyElement
CreatePolicyContainer	OptimizePolicyElement
ListPolicyContainers	TransformPolicyElement
Copy/Delete/MovePolicyContainer	TestPolicyElement
SelectPolicyElement	SignPolicy
Insert/Update/DeletePolicyElement	EncryptPolicy
CreateLock	
ShowLocks	
ReleaseLock	

Table 1: Listing of mandatory and optional PAWS operations

Below we give a high-level description of the PAWS operations to get an overview on the features of PAWS instances. For details, see chapter 9 to 21.

[1] Latest draft version see here: https://portal.opengeospatial.org/files/?artifact_id=55231

Mandatory Operations

GetCapabilities

The GetCapabilities operation allows a client to query service metadata (e.g. service version, supported operations and offered policy stores – repositories for policy containers; see section 10 for details) in form of Capabilities documents that describe the abilities of the specific PAWS instances.

ListPolicyContainers

The ListPolicyContainers operation allows a client to list all policy containers within one of the supported policy stores (see section 10 for details).

CreatePolicyContainer

The CreatePolicyContainer operation allows a client to create a policy container within one of the supported policy stores. Policy Containers can e.g. be used to group different versions of the same policy or semantically related policies or parts of policies (e.g. PermissionPolicySets as defined in [XACML-RBAC-Profile]).

CopyPolicyContainer

The CopyPolicyContainer operation allows a client to copy (and optionally to rename) a policy container from one of the supported policy stores to another.

MovePolicyContainer

The MovePolicyContainer operation allows a client to move (and optionally to rename) a policy container from one of the supported policy stores to another.

DeletePolicyContainer

The DeletePolicyContainer operation allows a client to delete an existing policy container including all its contained policies.

InsertPolicyElement

The InsertPolicyElement operation allows policy elements of type xacml:PolicySet, xacml:Policy, xacml:PolicySetIdReference, xacml:PolicyIdReference and xacml:Rule to be inserted in the specified policy container.

UpdatePolicyElement

The UpdatePolicyElement operation allows to update selected policy elements of type xacml:PolicySet, xacml:Policy, xacml:PolicySetIdReference, xacml:PolicyIdReference and xacml:Rule.

DeletePolicyElement

The DeletePolicyElement operation allows to delete selected policy elements of type xacml:PolicySet, xacml:Policy, xacml:PolicySetIdReference, xacml:PolicyIdReference, xacml:PolicySetIdReference, xacml:PolicyIdReference and xacml:Rule.

SelectPolicyElement

The SelectPolicyElement operation allows to select policy elements of type xacml:PolicySet, xacml:Policy, xacml:PolicySetIdReference, xacml:PolicyIdReference and xacml:Rule based on client specified query constraints.

CreateLock

The CreateLock operation allows locking policy stores, policy containers and complete policy trees for exclusive access in future operations. Complete means that the policy is locked from its root element down to all direct ancestor elements (for details see section 15).

ShowLocks

The ShowLocks operation allows a user to list active locks.

ReleaseLock

The ReleaseLock operation allows a user to release an active lock.

Optional Operations[2]

AnalyzePolicyElement

The generic AnalyzePolicyElement operation is an open hook for implementers to offer formal policy verification operations (e.g. well-known static policy analysis operations like is-satisfyable) on specific parts of XACML policy trees through PAWS instances.

OptimizePolicyElement

The generic OptimizePolicyElement operation is an open hook for implementers to offer specific syntactical or semantically optimization operations (e.g. simplify or evaluation-performance-optimized-restructuring).

TransformPolicyElement

The generic TransformPolicyElement operation is an open hook for implementers to offer specific syntactical or semantically transformation operations (e.g. transform an XACML v2.0 encoded policy into an XACML v3.0 encoded policy).

TestPolicyElement

---

[2] Note that chapters describing these operations still have to be written

The TestPolicyElement operation allows for testing certain parts of (Geo)XACML policies against self-defined XACML authorization decision requests by retrieving rich details of the policy evaluation steps performed by the underlying PDP.

EncryptPolicy

The EncryptPolicy operation allows to encrypt a (Geo)XACML policy available through the PAWS.

SignPolicy

The SignPolicy operation allows to sign a (Geo)XACML policy available through the PAWS.

2 Conformance

Table 1 specifies the conformance classes defined by this specification and the tests specified in Annex A that shall be satisfied in order to comply with each class.

Table 2 — PAWS Conformance Classes

Conformance class name	Conformance Class URN	Operation or behaviour
Basic PAWS	urn:ogc:conf-class:paws:1.0.0:basic	Any compliant PAWS instance shall implement the following operations: GetCapabilities, ListPolicyContainers, CreatePolicyContainer, CopyPolicyContainer, MovePolicyContainer, DeletePolicyContainer, InsertPolicyElement, UpdatePolicyElement, DeletePolicyElement, SelectPolicyElement, CreateLock, ShowLocks, and ReleaseLock.
Analyze PAWS	urn:ogc:conf-class:paws:1.0.0:analyze	Any compliant PAWS instance shall implement the Basic PAWS conformance class and shall additionally implement the AnalyzePolicyElement operation.
Optimize PAWS	urn:ogc:conf-class:paws:1.0.0:optimize	Any compliant PAWS instance shall implement the Analyze PAWS conformance class and shall additionally implement the OptimizePolicyElement operation.
Transform PAWS	urn:ogc:conf-class:paws:1.0.0:transform	Any compliant PAWS instance shall implement the Basic PAWS conformance class and shall additionally implement the TransformPolicyElement operation.
Test PAWS	urn:ogc:conf-class:paws:1.0.0:test	Any compliant PAWS instance shall implement the Basic PAWS conformance class and shall additionally implement the TestPolicyElement operation.
Trust PAWS	urn:ogc:conf-class:paws:1.0.0:trust	Any compliant PAWS instance shall implement the Basic PAWS conformance class and shall additionally implement the EncryptPolicy and SignPolicy operation.

3 Normative references

The following normative documents contain provisions, which, through reference in this text, constitute provisions of this document. For dated references, subsequence 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.

NAMESPACE, Namespaces in XML, World Wide Web Consortium Recommendation, Bray, T., Hollander, D., Layman, A., eds., available at http://www.w3.org/TR/1999/REC-xml-names-19990114

XML 1.0, *Extensible Markup Language (XML) 1.0 (Third Edition*), World Wide Web Consortium Recommendation, Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E., Yergeau, F., eds., available at http://www.w3c.org/TR/REC-xml

XPATH, *XML Path Language (XPath) 2.0,* World Wide Web Consortium Recommendation, Berglund, A., Boag, S., Chamberlin, D., Fernandez, M., Kay, M., Robie, J., Simeon, J., eds., available at http://www.w3.org/TR/xpath20/

XMLSCHEMA1, *XML Schema Part 1: Structures*, World Wide Web Consortium Recommendation, Thompson, H.S., Beech, D., Maloney, M., Mendelsohn, N., eds., available at http://www.w3.org/TR/2004/REC-xmlschema-1-20041028

XMLSCHEMA2, *XML Schema Part 2: Datatypes*, World Wide Web Consortium Recommendation, Biron, P.V., Malhotra, A., eds, available at http://www.w3.org/TR/2004/REC-xmlschema-2-20041028

OGC 06-121r9*OGC Web Services Common Specification 1.1*, OGC Document #06-121r9, Whiteside, A., ed., available at https://portal.opengeospatial.org/files/?artifact_id=20040

WSDL 1.1, Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/2001/NOTE-wsdl-20010315

XACML v3.0, Rissanen, Erik: eXtensible Access Control Markup Language (XACML) Version 3.0 / Organization for the Advancement of Structured Information Standards (OASIS). 2013. OASIS Committee Specification. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.pdf.

XACML v2.0, Rissanen, Erik: eXtensible Access Control Markup Language (XACML) Version 2.0 / Organization for the Advancement of Structured Information Standards (OASIS). 2013. OASIS Committee Specification. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.

GeoXACML v3.0, Matheus, Andreas: OGC Geospatial eXensible Access Control Markup Language (GeoXACML) 3.0 Core, OGC Document http://www.opengis.net/doc/DP/GEOXACML.

IETF RFC 4646, A. Phillips, M. Davis: Tags for Identifying Languages, http://www.ietf.org/rfc/rfc4646.txt

RFC 2141, R. Moats: URN Syntax, http://www.ietf.org/rfc/rfc2141.txt

ISO 8601, http://www.ietf.org/rfc/rfc3339.txt.

4 Terms and definitions

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

Policy Store

The abstract term XACML policy store describes a repository like entity that contains XACML policy containers.

Policy Container

The abstract term XACML policy containers describes an entity that can hold one or multiple XACML policy trees having either <PolicySet> or <Policy> elements as root elements.

Policy Tree

The term Policy tree refers to an XACML or GeoXACML encoded XML document.

Policy Element

A policy element is of one of the following types as defined in the XACML v2.0 or v3.0 specification:

- <xacml:PolicySet>...
- <xacml:Policy>...
- <xacml:PolicySetIdReference>...
- <xacml:PolicyIdReference>...
- <xacml:Rule>...

SHA-256

Secure Hash Algorithm with 256 bit digests.

XML tree

XML documents form a XML tree structure that starts at "the root" and branches to "the leaves".

State machine[3]

A state machine (also: finite-state machine (FSM) or finite-state automaton), is a mathematical model of computation used to design both computer programs. It is conceived as an abstract machine that can be in one of a finite number of states. The

[3] Definition taken from: https://en.wikipedia.org/wiki/Finite-state_machine

machine is in only one state at a time; the state it is in at any given time is called the current state. It can change from one state to another when initiated by a triggering event or condition; this is called a transition. A particular FSM is defined by a list of its states, and the triggering condition for each transition.

5 Conventions

5.1 Abbreviated terms

HTTP	Hypertext Transfer Protocol
HTTPS	Secure Hypertext Transfer Protocol
OGC	Open Geospatial Consortium
OWS	OGC Web Service
PAWS	Policy Administration Web Service
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
WSDL	Web Services Description Language
XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language
XPATH	XML Path Language
GeoXACML.	Geospatial eXensible Access Control Markup Language
(Geo)XACML	GeoXACML or XACML

5.2 Use of examples

This specification makes extensive use of informative XML examples. They are meant to illustrate the various aspects of PAWS operations. The bulk of the examples can be found in Annex C with some examples embedded in the body of the specification.

5.3 XML schemas

Throughout this specification, normative XML Schema fragments are used to define the XML encoding of PAWS operations. These fragments are gathered into a single validated schema file in Annex B.

6 Basic service aspects

6.1 Introduction

This section describes aspects of a PAWS behavior that are independent of particular operations or are common to several operations.

6.2 Request and response encoding and transport protocol

This specification defines an XML based request and response encoding only. It is assumed that the XML encoded PAWS requests and responses are always transmitted via HTTP POST – i.e. inside the HTTP body.

The root element of all PAWS requests encode the name of the service operation being invoked. The content of the resulting responses are in many cases copies of the original requests and express an acknowledgment that the operation has been successfully processed. Exceptions to this generic concept are responses to ShowLocks (see section 16), SelectPolicyElement (see section 21), AnalyzePolicyElement (see section ·), OptimizePolicyElement (see section ·), TransormPolicyElement (see section ·) and TestPolicyElement (see section ·) requests, as they need to contain richer response content.

6.3 Policy encoding

PAWS instances conformant to this specification shall operate upon access control policies encoded using XACML v2.0, v3.0 or GeoXACML v3.0. (see [XACML v2.0], [XACML v3.0] and [GeoXACML v3.0]).

It is important to note that the XACML encoding of the existing, to be edited policy is the “baseline schema”. If an e.g. to be inserted policy element is not valid according to the baseline schema or if the resulting modified policy is not valid according to the baseline schema, an exception shall be generated (see section 6.6 and chapter 9 to 21).

Further it is important to note that every (Geo)XACML policy can either be represented through a single XML tree with a unique root element (a <xacml:PolicySet> or <xacml:Policy> element) or is a set of interrelated XML trees. The latter case occurs in cases where policy authors use <xacml:PolicySetIdReference> or <xacml:PolicyIdReference> elements that point to other XACML trees. If an XACML policy is represented through a set of XML trees there must be a unique root element of type <xacml:PolicySet> that constitutes the entry point for PDP evaluation.

Note that this specification does not change or concretize the syntax and semantics of XACML PolicySetIdReference and PolicyIdReference elements. It is highly recommended to define or point to a XACML reference element profile and implement the PAWS reference handling accordingly.

In order to allow for simple selection of XACML policy elements that shall be operated upon this specification highly recommends that XACML policy element identifiers (i.e. xacml:PolicySetId, xacml:PolicyId and xacml:RuleId XML attributes) have unique names within the same policy container (see section 10 and 18).

6.4 XPath expression encoding

A number of operations defined in this specification may contain XPath encoded paramters that identify a subset of policies to be operated upon. XPath encoded parameters shall be encoded as described in the XML Path Language (XPath) 2.0 (Second Edition) specification (cp. [XPath2]).

6.5 URN encoding

A number of operations defined in this specification may contain urn encoded parameters that identify policy stores, policy containers or policy elements to be operated upon. All URN encoded parameters shall be be of data type paws:urnType as defined below.

```
<xsd:simpleType name="urnType">
  <xsd:annotation>
    <xsd:documentation>
      Regular expression to validate RFC 2141 URN syntax.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:restriction base="xsd:anyURI">
    <xsd:pattern value="urn:[a-zA-Z0-9][a-zA-Z0-9-]{1,31}:([a-zA-Z0-9()+,.:=@;$_!*'-]|%[0-9A-Fa-
       f]{2})+"/>
  </xsd:restriction>
</xsd:simpleType>
```

6.6 Exception reporting

In the event that a PAWS instance encounters one or more errors while processing a request or receives an invalid request, it shall generate an XML document indicating that an error has occurred. The format of the error(s) response is specified by, and shall validate against, the exception response schema defined in chapter 8 of the OGC Web Service Common Implementation Specification (see [OGC 06-121r9]).

An ows:ExceptionReport element may contain one or more PAWS processing exceptions specified using the ows:Exception element. The mandatory version attribute is used to indicate the version of the service exception report schema. For this version of the specification, this value shall be 1.0.0. The optional language attribute may be used to indicate the language used. The code list for the language parameter is defined in [IETF RFC 4646].

Individual exception messages are contained within the ows:ExceptionText element. The mandatory code attribute shall be used to associate an exception code with the accompanying message.

Table 3 lists exception codes defined or used by this specification.

Table 3 — PAWS exception codes

exceptionCode values	Meaning of code	Locator value
OperationParsingFailed	The request is badly formed and failed to be parsed by the server.	None, omit locator parameter.
OperationProcessingFailed	An error was encountered while processing the operation.	None, omit locator

		parameter.
MissingParameterValue[4]	Operation request does not include a parameter value.	Name of missing parameter.
InvalidParameterValue[4]	Operation request contains an invalid parameter value.	Name of parameter with invalid value.
VersionNegotiationFailed[4]	List of versions in "AcceptVersions" parameter value, in GetCapabilities operation request, did not include any version supported by this server.	None, omit locator parameter.
InvalidUpdateSequence[4]	Value of (optional) updateSequence parameter, inGetCapabilities operation request, is greater than current value of service metadata updateSequence number.	None, omit locator parameter.
NoApplicableCode[4]	No other exceptionCode specified by this service and server applies to this exception.	None, omit locator parameter.
PolicyStoreUnknown	The PolicyStoreId parameter of a PAWS request points to a policy store unknown by the PAWS instance.	Name of parameter with invalid value.
PolicyContainerUnknown	An error occured while processing a CopyPolicyContainer or MovePolicyContainer operation because the source policy container does not exisit.	Name of parameter with invalid value.
PolicyElementUnknown	The Query parameter of a PAWS request points to a policy element unknown by the PAWS instance.	Name of parameter with invalid value.
PolicyRootUnknown	The Query parameter of a PAWS request points to a policy element that is either unknown to the PAWS instance or does not represent a XACML policy root element (i.e. a PolicySet or Policy element without ancestors).	Name of parameter with invalid value.
EntityLocked	The request failed because an entity involved in the operation was locked by another user.	Identity of the locked entitiy
SourcePolicyStoreUnknown	The PolicyStoreId parameter of a PAWS request specifiying the source policy store points to a policy store unknown by the PAWS instance.	Name of parameter with invalid value.
DestinationPolicyStoreUnknown	The PolicyStoreId parameter of a PAWS request specifiying the destination policy store points to a policy store unknown by the PAWS instance.	Name of parameter with invalid value.
SourcePolicyContainerUnknown	An error occured while processing a CopyPolicyContainer or MovePolicyContainer operation because the source policy container does not exisit.	Name of parameter with invalid value.
PolicyContainerAlreadyExists	An error occured while processing a CreatePolicyContainer, CopyPolicyContainer or MovePolicyContainer operation because a policy container with an equal name already exists in the target policy store.	Name of parameter with invalid value.
PolicyContainerNameInvalid	An error occured while processing a CreatePolicyContainer, CopyPolicyContainer or MovePolicyContainer operation because the target policy container name is not a valid URN.	Name of parameter with invalid value.
LockIdParameterValueUnknown	An error occured while processing a CreateLock or RealeaseLock operation because the targeted lockId is unknown.	Name of parameter with invalid value.
LockExpiryUpdateFailed	An error occured while processing a lock expiry reset through a CreateLock operation.	None, omit locator parameter.
SubjectIdUnknown	An error occured while processing a ShowLocks operation because the specified SubjectId is unknown.	Name of parameter with invalid value.
PolicyElementIdentifierExists	An error occurred because a PAWS instance was requested to create a policy element with an already used XACML policy element identifier within that container.	Name of parameter with invalid value.
PolicyElementInsertReferenceNotDefined	An error occurred because a PAWS instance was requested to insert a policy element relative to an undefined location within an existing policy tree.	Name of parameter with invalid value.
PolicyElementUpdateReferenceNotDefined	An error occurred because a PAWS instance was requested to update a policy element relative to an undefined location within an existing policy tree.	

4 Taken from table 8 of the OGC Web Service Common Implementation Specification (see [OGC 06-121r9])

PolicyElementDeleteReferenceNotDefined	An error occurred because a PAWS instance was requested to delete a policy element at an undefined location within an existing policy tree	Name of parameter with invalid value.
PolicyElementInvalidAtDestination	An error occurred because a PAWS instance was requested to create a policy element at a location within an existing policy tree that does not allow the corresponding node type as one of its children or sibling elements	Name of parameter with invalid value.
PolicyElementNotSupported	An error occurred because a PAWS instance was requested to create a policy element that is not of type xacml:PolicySet, xacml:Policy, xacml:Rule, xacml PolicySetReference or xacml:PolicyReference	Name of parameter with invalid value.
PolicyElementInvalid	An error occurred because a PAWS instance was requested to insert an invalid (i.e. a not XACML schema conformant) XACML policy element of type xacml:PolicySet, xacml:Policy or xacml:Rule, xacml PolicySetReference or xacml:PolicyReference	Name of parameter with invalid value.
QueryInvalid	An error occurred because a PAWS instance was requested to perform an UpdatePolicyElement operation with a Query parameter that is empty or does not point to one or multiple element nodes of type xacml:PolicySet, xacml:Policy, xacml:Rule, xacml:PolicySetIdReference or xacml:PolicyIdReference (in case of an UpdatePolicyElement or DeletePolicyElement operation)	Name of parameter with invalid value.
DereferenceFailed	An error occurred because a PAWS instance was requested to dereference policy references but failed to do so.	None, omit locator parameter.
AuthenticationRequired	An error occurred because a PAWS instance was requested to perform a CreateLock or ReleaseLock operation without prior authenticating the interacting user.	None, omit locator parameter.

PAWS instances shall implement the generic exception codes in Table 25 of [OGC 06-121r9] for all conformance classes defined in this specification.

NOTE Multiple exceptions can be reported in a single exception report so service implementations should endeavor to report as many exceptions as necessary to clearly describe a problem.

EXAMPLE If parsing an operation fails because the value of a parameter is invalid, the service implementation should report an OperationParsingFailed exception and an InvalidParameterValue exception.

Annex D contains an example of an exception report.

6.7 PAWS Lock Concept

Web connections are stateless and therefore the semantics of serialize-able transactions are not preserved. Consider e.g. a select and update operation sequence: The policy administrator selects a policy element, updates it and then submits an update request to the PAWS. Serializeability is not guaranteed since there is nothing to ensure that while the policy element was being modified on the administrator side, another administrator modified that same feature in the policy store through the PAWS.

To ensure serialize-ability this specification supports a CreateLock (see section 15), ShowLocks (see section 16) and a ReleaseLock (see section 17) operation through which mutually exclusive access can be enforced. All lock operations belong to the basic conformance class as all transactional operations also do. This specification supports locking of entities of the following types: policy store, policy container and XACML policy tree (see section 10 for a definition of these entity types). Section 6.7.1 defines state machines that define the semantics of locks itself and the influence of locks on all other PAWS operations.

In cases of concurrent policy administration by multiple administrators, this specification highly recommends a client to lock policy stores, policy containers and policy trees prior to operating with transactional actions upon them. Through appropriate locking of entities, policy administrators can avoid unwanted side effects (e.g. dirty reads[5]) of concurrent transactions. It is in the responsibility of the administrator to lock the appropriate entity depending on the intended transaction.

PAWS instances support the ability to lock more than one policy store, policy container and policy tree where each entity may only be locked by one lock. Note that a policy tree is part of a policy container and the container in turn is part of a policy store. Hence a Lock on entity x affects all entities y that are members of the transitive closure of the part-of relationship (i.e. all entities y that are directly or indirectly part of entity x). Each of the locks is independent when viewed from the service defined by the PAWS specification.

6.7.1 State machines for PAWS locking

This section describes the state machines that define the semantics of the PAWS locking concept.

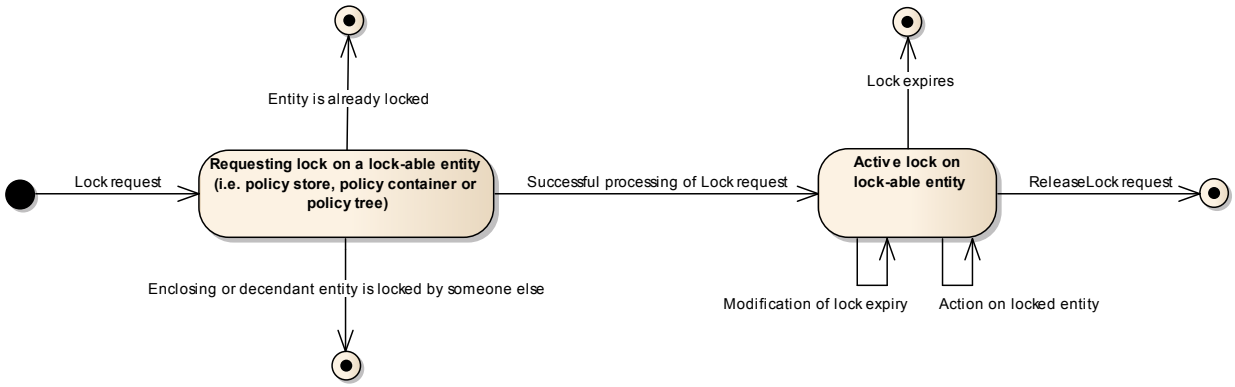

Figure 2 — State diagram for a PAWS lock

The state diagram above shows the allowed transitions between the states. All other state transitions are disallowed and are considered as errors if exhibited by a PAWS instance.

The state diagram below shows the influence of locked policy stores, policy containers and policy trees on the different PAWS request types. Note that the CreateLock and ReleaseLock operations are not addressed in Figure 3 as these operations are already covered in Figure 2. Further operations uninfluenced by locks (i.e. the GetCapabilities and ShowLocks operation) are not included in Figure 2.

[5] A dirty read occurs when a transaction is allowed to read data that has been modified by another running transaction and has not yet been committed by that other transaction.

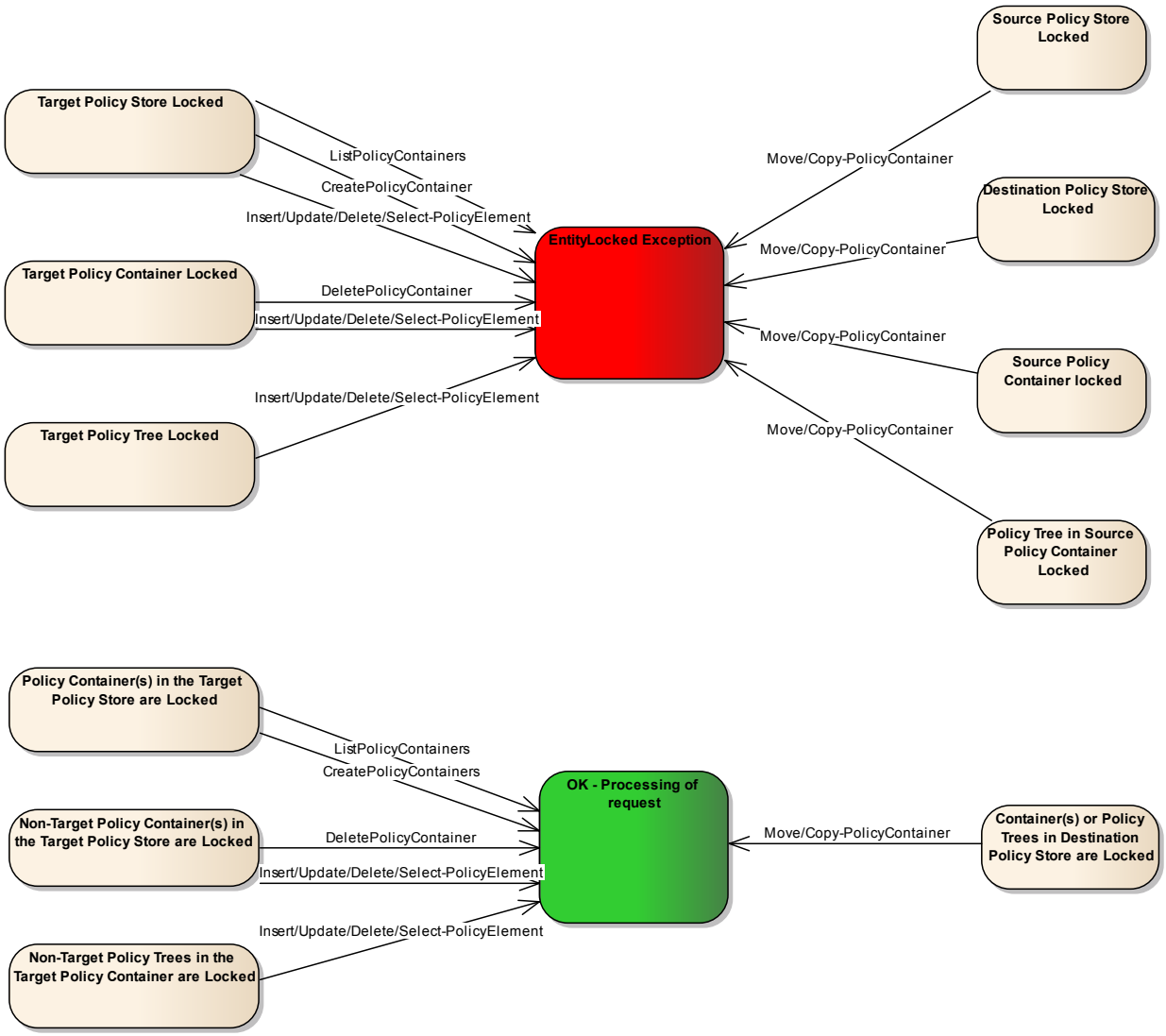

Figure 3 — State diagram defining the influence of locked entities on PAWS operations

6.8 Version numbering and negotiation

6.8.1 Version number form and value

Version numbers shall be encoded as described in section 7.3.1 of OGC 06-121r9. Implementations of this specification shall use the value "1.0.0" as the protocol version number.

6.8.2 Version number in service metadata and in requests

A PAWS instance that conforms to this specification shall list the version 1.0.0 in its service metadata (see section 9.3). A server may support several versions whose values clients may discover according to the version negotiation rules (cp. OGC 06-121r9, 7.3.2).

The version number used in requests to a server shall be equal to a version number that the server has declared it supports (except during negotiation as described below). If the server receives a request with a version number that it does not support, the server shall raise an InvalidParameterValue exception (cp. section 6.6).

The version number 1.0.0 shall be specified for any request that conforms to this specification.

6.8.3 Version number negotiation

All PAWS implementations shall support version negotiation according to the rules described in Section 7.3.2 of OGC 06-121r9.

6.9 Namespaces

Namespaces (see NAMESPACE) are used to discriminate XML vocabularies from one another. For the PAWS specification the following namespace definitions are of relevance:

- http://www.opengis.net/paws/1.0 - for the PAWS interface vocabulary
- http://www.opengis.net/ows/1.1 - for the OWS Common vocabulary (see [OGC 06-121r9])
- urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 - for the XACML v3.0 vocabulary
- urn:oasis:names:tc:xacml:2.0:core:schema:os - for the XACML v2.0 vocabulary
- urn:ogc:def:dataType:geoxacml:1.0 – for the GeoXACML vocabulary
- urn:ogc:def:function:geoxacml:1.0 – for the GeoXACML vocabulary
- http://www.w3.org/XML/1998/namespace -for the XML vocabulary
- http://www.w3.org/2001/XMLSchema - for the XML Schema vocabulary
- http://www.w3.org/2001/XMLSchema-instance
- http://www.w3.org/1999/xlink - for the XLink vocabulary

In PAWS requests and responses used namespaces shall be encoded using the notation "xmlns:prefix=namespace_uri" in the root element of the request (see [NAMESPACE]).

7 Common request parameters

7.1 Introduction

This section describes request parameters common to all or several operations defined in this specification.

Note: Only an XML encoding is defined for PAWS request messages.

7.2 Base request type

7.2.1 XML encoding and semantics

The following XML Schema fragment specifies the XML encoding of the type BaseRequest:

```
<xsd:complexType name="BaseRequestType" abstract="true">
  <xsd:attribute name="service" type="xsd:string" use="required" fixed="PAWS"/>
  <xsd:attribute name="version" type="xsd:string" use="required" fixed="1.0.0"/>
</xsd:complexType>
```

The base request is an abstract type from which all PAWS operations, except the GetCapabilities operation, are sub-typed.

NOTE The root element of all PAWS requests encode the name of the service operation being invoked (cp. 6.2).

7.2.1.1 service parameter

The mandatory service parameter shall be encoded using an XML attribute named service (cp. 7.2.1).

The service parameter shall be used to indicate which of the available service types, at a particular server, is being invoked. When invoking a PAWS, the value of the service parameter shall be "PAWS".

7.2.1.2 version parameter

The mandatory version parameter shall be encoded using an XML attribute named version (cp. 7.2.1). All PAWS requests (except the GetCapabilities operation) shall include the version parameter.

The version parameter shall be used to indicate to which version of the PAWS specification the request encoding conforms to and is used in version negotiation as described in section 6.8.3. When encoding a PAWS request in accordance with this specification, the value of the version attribute shall be fixed to 1.0.0, which corresponds to the version of this specification.

7.3 Query expressions

7.3.1 Introduction

Most PAWS operations support a Query parameter through which users can identify a complete policy or policy fragment or a set of policies or policy fragments respectively that shall be operated upon. This query parameter can be encoded through a valid XPath 2.0 expression. Valid values for Query parameters are XPath 2.0 expressions that point to exactly one element node of one of the following node types:

- xacml:PolicySet element
- xacml:Policy element
- xacml:PolicySetIdReference
- xacml:PolicIdReference
- xacml:Rule element

EXAMPLE The SelectPolicyElement operation (see Chapter 21) uses query parameter to identify a subset of policies or policy fragments in a specific policy container that will then be presented to a client in a response document. Similarly, the CreateLock operation on a policy tree (see chapter 15) uses a query expression to identify the root element of a policy tree that shall be locked.

7.3.2 XML encoding

The following XML Schema fragment defines the XML encoding for the two types of PAWS query expressions:

```
<xsd:element name="Query">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="namespace" type="xsd:anyURI" use="optional"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

The text node of a paws:Query element is of type xsd:string and shall represent a valid XPath 2.0 expression (cp. [XPath 2.0]) that points to exactly one node of type xacml:PolicySet, xacml:Policy or xacml:Rule.

The optional namespace parameter is used to map the namespace prefixes specified in the text node of the Query parameter to the corresponding namespaces.

8 Common response parameters

8.1 Introduction

This section describes response parameter common to all operations defined in this specification.

Note: Only an XML encoding is defined for PAWS response messages.

8.2 Base response type

8.2.1 XML encoding and semantics

The following XML Schema fragment specifies the XML encoding of the type BaseResponse:

```
<xsd:complexType name="BaseResponseType" abstract="true">
    <xsd:attribute name="timeStamp" type="xsd:dateTime" use="required"/>
</xsd:complexType>
```

8.2.1.1 timeStamp parameter (mandatory)

The mandatory timeStamp parameter (based on ISO 8601) shall be used by a PAWS to indicate the time and date when a response was generated. The timeStamp parameter shall be encoded as an XML attribute of the PAWS response root element.

8.3 Use of the schemaLocation attribute

Any document generated by a PAWS shall reference an appropriate XML application schema document so that the output may be validated. This shall be accomplished using the schemaLocation attribute, as defined in the XML Schema specification (see [XMLSCHEMA1]).

EXAMPLE The following XML fragment shows the use of the schemaLocation attribute on the root element indicating the location of the XML Schema document that may be used for validation:

```
<?xml version="1.0" ?>
<paws:CreatePolicyContainerResponse timeStamp="2013-04-01T21:00:00Z"
  xmlns="http://www.opengis.net/paws/1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:paws="http://www.opengis.net/paws/1.0"
  xsi:schemaLocation="http://www.opengis.net/paws/1.0
http://schemas.opengis.net/paws/1.0.0/paws.xsd">
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
</paws:CreatePolicyContainerResponse>
```

9 GetCapabilities operation

9.1 Introduction

The GetCapabilities operation generates a service metadata document describing a PAWS instance provided by a server. This operation also supports negotiation of the specification version being used for client-server interactions.

Note: It is recommended that PAWS instances do support a KVP encoded version of the GetCapabilities operation as all OGC Web Service do so. This is however not a mandatory requirement (cp. section 6.2).

9.2 Request

9.2.1 XML encoding and semantics

The following XML Schema fragment defines the XML-encoding of the GetCapabilities request:

```
<xsd:element name="GetCapabilities" type="paws:GetCapabilitiesType"/>
<xsd:complexType name="GetCapabilitiesType">
  <xsd:complexContent>
    <xsd:extension base="ows:GetCapabilitiesType">
      <xsd:attribute name="service" type="ows:ServiceType" use="required" fixed="PAWS"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The base type, ows:GetCapabilitiesType, and its semantics are defined in the OWS Common Implementation Specification (see OGC 06-121-r9, 7.2.4).

9.2.2 Example

```
<paws:GetCapabilities service="PAWS"
  xmlns="http://www.opengis.net/paws/1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:paws="http://www.opengis.net/paws/1.0"
  xsi:schemaLocation="http://www.opengis.net/paws/1.0
  http://schemas.opengis.net/paws/1.0.0/paws.xsd"/>[6]
```

9.3 Response

9.3.1 XML encoding and semantics

The root element of the response to a GetCapabilities request is the paws:GetCapabilitiesResponse element which is declared by the following XML Schema fragment:

```
<xsd:element name="GetCapabilitiesResponse" type="paws:PAWS_GetCapabilitiesResponseType"/>
<xsd:complexType name="PAWS_GetCapabilitiesResponseType">
  <xsd:complexContent>
    <xsd:extension base="ows:CapabilitiesBaseType">
      <xsd:sequence>
        <xsd:element name="WSDL">
          <xsd:complexType>
```

[6] Note: Namespace definition will be obmitted in the following examples.

```
            <xsd:complexContent>
              <xsd:restriction base="xsd:anyType">
                <xsd:attributeGroup ref="xlink:simpleAttrs"/>
              </xsd:restriction>
            </xsd:complexContent>
          </xsd:complexType>
        </xsd:element>
        <xsd:element ref="paws:PolicyStoreList"/>
        <xsd:element ref="paws:SupportedConformanceClassesList"/>
      </xsd:sequence>
      <xsd:attribute name="timeStamp" type="xsd:dateTime" use="required"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="PolicyStoreList" type="paws:PolicyStoreListType"/>
<xsd:complexType name="PolicyStoreListType">
  <xsd:sequence minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="PolicyStore" type="paws:PolicyStoreType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="PolicyStoreType">
  <xsd:sequence>
    <xsd:element name="Name" type="paws:urnType"/>
    <xsd:element name="Title" type="xsd:string" minOccurs="0"/>
    <xsd:element name="Description" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:element name="SupportedConformanceClassesList"
       type="paws:SupportedConformanceClassesListType"/>
<xsd:complexType name="SupportedConformanceClassesListType">
  <xsd:sequence minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="SupportedConformanceClass" type="paws:SupportedConformanceClassType"
       maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

The base type, ows:CapabilitiesBaseType, is defined in the OWS Common Implementation Specification (see OGC 06-121-r9, 7.2.4).

In addition to the sections defined in section 7.4 of OGC 06-121r9, the capabilities response document shall contain the following sections:

1. **WSDL section (mandatory)**
   This section allows a server to reference an WSDL document that describes the operations and bindings that the service offers (see Annex D).

2. **PolicyStoreList section (mandatory)**
   This section defines a list of policy stores offered by the PAWS instance. Lightweight metadata is provided about each policy store as described in 9.3.1.1. More details on the policy store concept can be found in section 10.

3. **SupportedConformanceClasses (mandatory)**
   This section defines a list of PAWS conformance classes (cp. section 2) the corresponding PAWS instance is compliant with. More details on this concept can be found in section 9.3.1.2.

9.3.1.1 PolicyStoreList section

The purpose of the paws:PolicyStoreList element is to contain a list of policy stores offered by a PAWS instance. The following XML Schema fragment defines the paws:PolicyStoreList element:

```
<xsd:element name="PolicyStoreList" type="paws:PolicyStoreListType"/>
<xsd:complexType name="PolicyStoreListType">
  <xsd:sequence minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="PolicyStore" type="paws:PolicyStoreType" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="PolicyStoreType">
  <xsd:sequence>
    <xsd:element name="Name" type="paws:urnType"/>
    <xsd:element name="Title" type="xsd:string" minOccurs="0"/>
    <xsd:element name="Description" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

The paws:PolicyStoreList element shall contain a paws:PolicyStore element for each policy store that the service offers. The paws:PolicyStore element contains metadata about the policy store.

Table 11 lists the elements that are used to describe each policy store listed within the paws:PolicyStoreList element.

Table 4 — Elements to describe policy stores

Element name	Description
Name	The unique[7] name of a policy store in URN format. This element is mandatory.
Title	A human-readable title that briefly identify this policy store in menus.
Description	A paws:Description elements is a descriptive narrative for more information about the policy store (e.g. type of the underlying storage model for this policy store – e.g. file-system-folder, table-in-xml-db).

9.3.1.2 SupportedConformanceClass section

In response to a GetCapabilities request, the corresponding PAWS instance shall list all the PAWS conformance classes it complies with. Allowed entries in this list are URN values as defined in table 1 in section 2 (cp. column “Conformance Class URN”). Note that in cases where a conformance class implies conformance to another conformance class, the compliance to the inherited conformance class shall however be explicitly expressed and hence listed in the SupportedConformanceClass section.

9.3.2 Example

```
<paws:GetCapabilitiesResponse timeStamp="2013-04-01T21:00:00Z" version="1.0.0" ...>
  <paws:WSDL xlink:href="http://www.someserver.com/paws.wsdl"/>
  <paws:PolicyStoreList>
    <paws:PolicyStore>
      <paws:Name>urn:mydomain:MyFirstPolicyStore</paws:Name>
      <paws:Title>MyFirstPolicyStore (Testing Environment)</paws:Title>
      <paws:Description>
        This policy store contains policies protecting organisation unit A's WFS instances. It is a
        test environment. The implemenation of the policy store is based on file system folders.
      </paws:Description>
    </paws:PolicyStore>
```

[7] At least unique for this PAWS instance.

```
  </paws:PolicyStoreList>
  <paws:SupportedConformanceClassesList>
    <paws:SupportedConformanceClass>
      urn:ogc:conf-class:paws:1.0.0:basic
    </paws:SupportedConformanceClass>
  </paws:SupportedConformanceClassesList>
</paws:GetCapabilitiesResponse>
```

9.4 Exceptions

In the event that a PAWS instance encounters an error parsing a GetCapabilities request, it shall raise an **OperationParsingFailed** exception as described in section 6.6.

In the event that a web feature service encounters an error processing a GetCapabilities request, it shall raise an **OperationProcessingFailed** exception as described in section 6.6.

Additionally the exception codes for the GetCapabilities operation taken from table 8 of the OWS Common Implementation Specification (see OGC 06-121-r9, 7.4.1) can occur (see section 6.6).

10 CreatePolicyContainer operation

In the following, the abstract term XACML policy store describes an entity that contains XACML policy containers. Each of these XACML policy containers can be used to store one or more XACML policies or parts of XACML policies in form of <PolicySet> or <Policy> elements including their ancestors.

It is assumed that PAWS instances provide policy stores. These policy stores must be setup and configured by the PAWS administrators. Unique names for these stores shall follow the naming schema defined in section 9.3.1.1 and shall be published in the corresponding section of the capabilities documents returned in response to PAWS getCapabilities requests (cp. Section 9.3.1).

The implementation type of a policy store shall be completely opaque to PAWS users. How policy stores are realized in a PAWS implementation is left open to the implementers (e.g. via file system folders or databases). It is also left open whether the available stores are of the same or of different implementation types.

10.1 Introduction

The CreatePolicyContainer operation creates an (Geo)XACML policy container within one of the policy stores provided by the PAWS instance (cp. section 9 how to query the list of provided policy stores).

In cases of concurrent policy administration by multiple administrators, it is highly recommended to lock the destination policy store upfront the CreatePolicyContainer operation (see section 15 for details).

10.2 Request

10.2.1 XML encoding and semantics

The following XML Schema fragment defines the XML-encoding of the CreatePolicyContainer request:

```
<xsd:element name="CreatePolicyContainer" type="paws:CreatePolicyContainerType"/>
<xsd:complexType name="CreatePolicyContainerType">
  <xsd:complexContent>
    <xsd:extension base="paws:BaseRequestType">
      <xsd:sequence minOccurs="1" maxOccurs="1">
        <xsd:element name="PolicyStoreId" type="paws:urnType"/>
        <xsd:element name="PolicyContainerId" type="paws:urnType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The base type, paws:BaseRequestType is defined in section 7.2.

10.2.1.1 PolicyStoreId parameter (mandatory)

The value of the PolicyStoreId parameter shall be a valid URN (cp. 6.5) and specifies the policy store in which the new policy container shall be created. The PAWS instance shall

report an exception in case a user tries to create a policy container in an unknown policy store or in a policy store locked by someone else (cp. section 10.4)

10.2.1.2 PolicyContainerId parameter (mandatory)

The value of the PolicyContainerId parameter shall be a valid URN (cp. 6.5) and defines the name of the to be created policy container. The names of containers with a policy store shall be unique URN values. This implies that a PAWS instance shall report an exception in case a user tries to create a policy container with a name equal to a container already existing within that store (cp. section 10.4).

10.2.2 Example

The XML document below shows a CreatePolicyContainer request. The effect of this request is to create a policy container named "urn:mydomain:MyFirstContainer" within the policy store "urn:mydomain:MyFirstPolicyStore".

```
<paws:CreatePolicyContainer service="PAWS" version="1.0.0" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
</paws:CreatePolicyContainer>
```

10.3 Response

10.3.1 XML encoding and semantics

The root element of the response to a CreatePolicyContainer request is the paws:CreatePolicyContainerResponse element which is declared by the following XML Schema fragment:

```
<xsd:element name="CreatePolicyContainerResponse" type="paws:CreatePolicyContainerResponseType"/>
<xsd:complexType name="CreatePolicyContainerResponseType">
  <xsd:complexContent>
    <xsd:extension base="paws:BaseResponseType">
      <xsd:sequence minOccurs="1" maxOccurs="1">
        <xsd:element name="PolicyStoreId" type="paws:urnType"/>
        <xsd:element name="PolicyContainerId" type="paws:urnType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The base type, paws:BaseResponseType is defined in section 8.2.

The CreatePolicyContainerResponse document shall contain the following elements:

1. **PolicyStoreId element (mandatory)**
   This mandatory element shall specify the policy store that was operated upon while processing the CreatePolicyContainer request. It shall therefore be a copy of the request's PolicyStoreId element.
2. **PolicyContainerId element (mandatory)**
   This mandatory element shall specify the policy container that was created in the policy store and shall therefore be a copy of the request's PolicyContainerId element.

10.3.2 Example

The XML document below shows a valid response to a CreatePolicyContainer request. The response represents an acknowledgement that the policy container "urn:mydomain:MyFirstContainer" has been created successfully in the policy store "urn:mydomain:MyFirstPolicyStore".

```
<paws:CreatePolicyContainerResponse timeStamp="2013-04-01T21:00:00Z" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
</paws:CreatePolicyContainerResponse>
```

10.4 Exceptions

In the event that a PAWS instance encounters an error parsing a CreatePolicyContainer request, it shall raise an **OperationParsingFailed** exception as described in section 6.6.

In the event that a PAWS instance encounters an error processing a CreatePolicyContainer request, it shall raise an **OperationProcessingFailed** exception as described in section 6.6.

In the event that a PAWS instance is requested to create a policy container within a policy store that is not known to the PAWS instance, it shall raise a **PolicyStoreUnknown** exception as described in section 6.6.

In the event that a PAWS instance is requested to create a policy container within a policy store with a name of an already existing policy contaner within that same store, it shall raise a **PolicyContainerAlreadyExists** exception as described in section 6.6.

In the event that a PAWS instance is requested to create a policy container within a policy store with an invalid name (i.e. not URN format compliant), it shall raise a **PolicyContainerNameInvalid** exception as described in section 6.6.

In the event that a PAWS instance is requested to operate on a policy store locked by some other user, it shall raise an **EntityLocked** exception as described in section 6.6.

Note that it does not constitute an error if a user tries to create a container within a policy store who has some contained containers or policy trees that are locked (see section 15 for details on the lock concept).

11 ListPolicyContainers operation

11.1 Introduction

The ListPolicyContainers operation is used to query a list of existing policy containers within a specific policy store.

11.2 Request

11.2.1 XML encoding and semantics

The following XML Schema fragment defines the XML-encoding of the ListPolicyContainers request:

```
<xsd:element name="ListPolicyContainers" type="paws:ListPolicyContainersType"/>
xsd:complexType name="ListPolicyContainersType">
  <xsd:complexContent>
    <xsd:extension base="paws:BaseRequestType">
      <xsd:sequence minOccurs="1" maxOccurs="1">
        <xsd:element name="PolicyStoreId" type="paws:urnType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The base type, paws: BaseRequestType is defined in section 7.2.

11.2.1.1 PolicyStoreId parameter (mandatory)

The value of the PolicyStoreId parameter shall be a valid URN (cp. 6.5) and specifies one of the policy stores known to the PAWS instance. The Ids of the policy containers that currently exist within that policy store represent the response of the ListPolicyContainer operation.

11.2.2 Example

The XML document below shows a ListPolicyContainer request. The effect of this request is to list the policy container Ids of policy containers that exist within the policy store "urn:mydomain:MyFirstPolicyStore".

```
<paws:ListPolicyContainers service="PAWS" version="1.0.0" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
</paws:ListPolicyContainers>
```

11.3 Response

11.3.1 XML encoding and semantics

The root element of the response to a ListPolicyContainer request is the paws:ListPolicyContainerResponse element which is declared by the following XML Schema fragment:

```
<xsd:element name="ListPolicyContainersResponse" type="paws:ListPolicyContainersResponseType"/>
<xsd:complexType name="ListPolicyContainersResponseType">
  <xsd:complexContent>
    <xsd:extension base="paws:BaseResponseType">
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="PolicyContainerId" type="paws:urnType"/>
      </xsd:sequence>
    </xsd:extension>
```

```
</xsd:complexType>
```

The base type, paws:BaseResponseType is defined in section 8.2.

The ListPolicyContainerResponse document shall contain the following element:

1. **PolicyContainerId element**
   This element can occur zero to unbounded times and describes the Ids of all existing policy containers of the policy store specified by the PolicyStoreId parameter of the request. Each PolicyContainerId element shall specify exactly one Id of one of the existing policy containers within that policy store.

11.3.2 Example

The XML document below shows a valid response to a ListPolicyContainer request. The response states that the policy containers "urn:mydomain:MyFirstContainer" and "urn:mydomain:MySecondContainer" exist within the policy store "urn:mydomain:MyFirstPolicyStore".

```
<paws:ListPolicyContainersResponse timeStamp="2013-04-01T21:00:00Z" ...>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
  <paws:PolicyContainerId>urn:mydomain:MySecondContainer</paws:PolicyContainerId>
</paws:ListPolicyContainersResponse>
```

11.4 Exceptions

In the event that a PAWS instance encounters an error parsing a ListPolicyContainers request, it shall raise an **OperationParsingFailed** exception as described in section 6.6.

In the event that a PAWS instance encounters an error processing a ListPolicyContainers request, it shall raise an **OperationProcessingFailed** exception as described in section 6.6.

In the event that a PAWS instance is requested to list the policy containers of a policy store that is not known to the PAWS instance, it shall raise a **PolicyStoreUnknown** exception as described in section 6.6.

Further, the PAWS instance shall report an **EntityLocked** exception in case a user tries to query the Ids of policy containers from a locked policy store.

Note: The listing of locked policy containers is not considered to be an error (see section 15 for details).

12 CopyPolicyContainer operation

12.1 Introduction

The CopyPolicyContainer operation copies an (Geo)XACML policy container from one policy store to another.

In cases of concurrent policy administration by multiple administrators, it is highly recommended to lock the source policy container and the destination policy store upfront the CopyPolicyContainer operation to avoid unwanted side effects through concurrent transactions (see section 15 for details).

12.2 Request

12.2.1 XML encoding and semantics

The following XML Schema fragment defines the XML-encoding of the CopyPolicyContainer request:

```
<xsd:element name="CopyPolicyContainer" type="paws:CopyPolicyContainerType"/>
<xsd:complexType name="CopyPolicyContainerType">
  <xsd:complexContent>
    <xsd:extension base="paws:BaseRequestType">
      <xsd:sequence minOccurs="1" maxOccurs="1">
        <xsd:element name="SourcePolicyStoreId" type="paws:urnType"/>
        <xsd:element name="SourcePolicyContainerId" type="paws:urnType"/>
        <xsd:element name="DestinationPolicyStoreId" type="paws:urnType"/>
        <xsd:element name="DestinationPolicyContainerId" type="paws:urnType" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The base type paws:BaseRequestType is defined in section 7.2.

12.2.1.1 SourcePolicyStoreId parameter (mandatory)

The value of the SourcePolicyStoreId parameter shall be a valid URN (cp. 6.5) and specifies the policy store from which the source policy container shall be selected.

12.2.1.2 SourcePolicyContainerId parameter (mandatory)

The value of the SourcePolicyContainerId parameter shall be a valid URN (cp. 6.5) and specifies the name of the to be copied source policy container.

Note: In contrast to the MovePolicyContainer operation (see section 13), after successfully processing a CopyPolicyContainer operation, the selected policy container will exist in the source and destination policy store.

12.2.1.3 DestinationPolicyStoreId parameter (mandatory)

The value of the DestinationPolicyStoreId parameter shall be a valid URN (cp. 6.5) and specifies the destination policy store of the copy operation.

12.2.1.4 DestinationPolicyContainerId parameter (optional)

The value of the optional DestinationPolicyContainerId parameter shall be a valid URN (cp.

DestinationPolicyContainerId parameter means that the name of the to be copied policy container shall remain the same as in the source policy store.

12.2.2 Example

The XML document below shows a CopyPolicyContainer request. The effect of this request is to copy the policy container named "urn:mydomain:MyFirstContainer" from policy store "urn:mydomain:MyFirstPolicyStore" to the policy store "urn:mydomain:MyBackupPolicyStore". The name of the to be copied policy container will remain "urn:mydomain:MyFirstContainer" after the termination of the copy operation (cp. the absence of the DestinationPolicyContainerId element).

```
<paws:CopyPolicyContainer service="PAWS" version="1.0.0" ...>
  <paws:SourcePolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:SourcePolicyStoreId>
  <paws:SourcePolicyContainerId>urn:mydomain:MyFirstContainer</paws:SourcePolicyContainerId>
  <paws:DestinationPolicyStoreId>urn:mydomain:MyBackupPolicyStore</paws:DestinationPolicyStoreId>
</paws:CopyPolicyContainer>
```

12.3 Response

12.3.1 XML encoding and semantics

The root element of the response to a CopyPolicyContainer request is the paws:CopyPolicyContainerResponse element which is declared by the following XML Schema fragment:

```
<xsd:element name="CopyPolicyContainerResponse" type="paws:CopyPolicyContainerResponseType"/>
<xsd:complexType name="CopyPolicyContainerResponseType">
  <xsd:complexContent>
    <xsd:extension base="paws:BaseResponseType">
      <xsd:sequence minOccurs="1" maxOccurs="1">
        <xsd:element name="SourcePolicyStoreId" type="paws:urnType"/>
        <xsd:element name="SourcePolicyContainerId" type="paws:urnType"/>
        <xsd:element name="DestinationPolicyStoreId" type="paws:urnType"/>
        <xsd:element name="DestinationPolicyContainerId" type="paws:urnType" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The base type, paws:BaseResponseType is defined in section 8.2.

The CopyPolicyContainerResponse document shall contain the same elements as the corresponding request in case of a successful termination of the requested copy operation. In all other situations, an exception report shall be generated (cp. 12.4).

12.3.2 Example

The XML document below shows a valid response to a CopyPolicyContainer request. The response represents an acknowledgement that the policy container "urn:mydomain:MyFirstContainer" has been copied successfully from the policy store "urn:mydomain:MyFirstPolicyStore" to the policy store "urn:mydomain:MyBackupPolicyStore" without any name change.

```
<paws:CopyPolicyContainerResponse timeStamp="2013-04-01T21:00:00Z" ...>
  <paws:SourcePolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:SourcePolicyStoreId>
  <paws:SourcePolicyContainerId>urn:mydomain:MyFirstContainer</paws:SourcePolicyContainerId>
  <paws:DestinationPolicyStoreId>urn:mydomain:MyBackupPolicyStore</paws:DestinationPolicyStoreId>
</paws:CopyPolicyContainerResponse>
```

12.4 Exceptions

In the event that a PAWS instance encounters an error parsing a CopyPolicyContainer request, it shall raise an **OperationParsingFailed** exception as described in section 6.6.

In the event that a PAWS instance encounters an error processing a CopyPolicyContainer request, it shall raise an **OperationProcessingFailed** exception as described in section 6.6.

In the event that a PAWS instance is requested to copy a policy container from a policy store that is not known to the PAWS instance, it shall raise a **SourcePolicyStoreUnknown** exception as described in section 6.6.

In the event that a PAWS instance is requested to copy a policy container to a policy store that is not known to the PAWS instance, it shall raise a **DestinationPolicyStoreUnknown** exception as described in section 6.6.

In the event that a PAWS instance is requested to copy a nonexistent policy container from a policy store, it shall raise a **SourcePolicyContainerUnknown** exception as described in section 6.6.

In the event that a PAWS instance is requested to copy a policy container with its existing or newly defined name to a policy store that already contains a policy container with such a name, it shall raise a **PolicyContainerAlreadyExists** exception as described in section 6.6.

In the event that a PAWS instance is requested to rename the policy container during the copy process to an a invalid name (i.e. not URN format compliant), it shall raise a **PolicyContainerNameInvalid** exception as described in section 6.6.

In the event that a PAWS instance is requested to operate on a source policy store locked by some other user, it shall raise **EntityLocked** exceptions for the locked entity as described in section 6.6.

In the event that a PAWS instance is requested to operate on a destination policy store locked by some other user, it shall raise **EntityLocked** exceptions for the locked entity as described in section 6.6.

In the event that a PAWS instance is requested to copy a policy container locked by some other user, it shall raise an **EntityLocked** exception as described in section 6.6.

Note: It does not constitute an error if a user tries to copy a policy container from and to a policy store that has some contained containers (with exception of the one that is operated on) or policy trees that are locked (see section 15 for details on the lock concept).

13 MovePolicyContainer operation

13.1 Introduction

The MovePolicyContainer operation moves a (Geo)XACML policy container from one policy store to another and deletes the policy container in the source store.

In cases of concurrent policy administration by multiple administrators, it is highly recommended to lock the source policy container and the destination policy store upfront the MovePolicyContainer operation to avoid unwanted side effects through concurrent transactions (see section 15 for details).

Note: This specification does not define a RenamePolicyContainer operation as the MovePolicyContainer operation with appropriately set parameters can achieve the same result.

13.2 Request

13.2.1 XML encoding and semantics

The following XML Schema fragment defines the XML-encoding of the MovePolicyContainer request:

```
<xsd:element name="MovePolicyContainer" type="paws:MovePolicyContainerType"/>
<xsd:complexType name="MovePolicyContainerType">
  <xsd:complexContent>
    <xsd:extension base="paws:BaseRequestType">
      <xsd:sequence minOccurs="1" maxOccurs="1">
        <xsd:element name="SourcePolicyStoreId" type="paws:urnType"/>
        <xsd:element name="SourcePolicyContainerId" type="paws:urnType"/>
        <xsd:element name="DestinationPolicyStoreId" type="paws:urnType"/>
        <xsd:element name="DestinationPolicyContainerId" type="paws:urnType" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The base type, paws: BaseRequestType is defined in section 7.2.

13.2.1.1 SourcePolicyStoreId parameter (mandatory)

The value of the SourcePolicyStoreId parameter shall be a valid URN (cp. 6.5) and specifies the policy store from which the to be moved policy container shall be selected.

13.2.1.2 SourcePolicyContainerId parameter (mandatory)

The value of the PolicyContainerId parameter shall be a valid URN (cp. 6.5) and specifies the name of the to be moved policy container. After the successful processing of the MovePolicyContainer operation, the selected PolicyContainer will not exist anymore in the source policy store.

13.2.1.3 DestinationPolicyStoreId parameter (mandatory)

The value of the DestinationPolicyStoreId parameter shall be a valid URN (cp. 6.5) and specifies the destination policy store of the move operation.

13.2.1.4 DestinationPolicyContainerId parameter (optional)

The value of the optional DestinationPolicyContainerId parameter shall be a valid URN (cp. 6.5) and specifies the new name of the to be moved policy container. The absence of the DestinationPolicyContainerId parameter means that the name of the to be moved policy container shall remain the same as in the source policy store.

13.2.2 Example

The XML document below shows a MovePolicyContainer request. The effect of this request is to move the policy container named "urn:mydomain:MyFirstContainer" from policy store "urn:mydomain:MyFirstPolicyStore" to the policy store "urn:mydomain:MyTrashBinPolicyStore". The name of the to be moved policy container will remain "urn:mydomain:MyFirstContainer" after the termination of the move operation (cp. the absence of the DestinationPolicyContainerId element).

```
<paws:MovePolicyContainer service="PAWS" version="1.0.0" ...>
  <paws:SourcePolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:SourcePolicyStoreId>
  <paws:SourcePolicyContainerId>urn:mydomain:MyFirstContainer</paws:SourcePolicyContainerId>
  <paws:DestinationPolicyStoreId>urn:mydomain:MyBackupPolicyStore</paws:DestinationPolicyStoreId>
</paws:MovePolicyContainer>
```

13.3 Response

13.3.1 XML encoding and semantics

The root element of the response to a MovePolicyContainer request is the paws:MovePolicyContainerResponse element which is declared by the following XML Schema fragment:

```
<xsd:element name="MovePolicyContainerResponse" type="paws:MovePolicyContainerResponseType"/>
<xsd:complexType name="MovePolicyContainerResponseType">
  <xsd:complexContent>
    <xsd:extension base="paws:BaseResponseType">
      <xsd:sequence minOccurs="1" maxOccurs="1">
        <xsd:element name="SourcePolicyStoreId" type="paws:urnType"/>
        <xsd:element name="SourcePolicyContainerId" type="paws:urnType"/>
        <xsd:element name="DestinationPolicyStoreId" type="paws:urnType"/>
        <xsd:element name="DestinationPolicyContainerId" type="paws:urnType" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The base type, paws:BaseResponseType is defined in section 8.2.

The MovePolicyContainerResponse document shall contain the same elements as the corresponding request in case of a successful termination of the requested move operation. In all other situations, an exception report shall be generated (see section 13.4).

13.3.2 Example

The XML document below shows a valid response to a MovePolicyContainer request. The response represents an acknowledgement that the policy container "urn:mydomain:MyFirstContainer" has been moved successfully from the policy store "urn:mydomain:MyFirstPolicyStore" to the policy store "urn:mydomain:MyTrashBinPolicyStore" without any name change.

```
<paws:MovePolicyContainerResponse timeStamp="2013-04-01T21:00:00Z" ...>
```

```
  <paws:SourcePolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:SourcePolicyStoreId>
  <paws:SourcePolicyContainerId>urn:mydomain:MyFirstContainer</paws:SourcePolicyContainerId>
  <paws:DestinationPolicyStoreId>urn:mydomain:MyBackupPolicyStore</paws:DestinationPolicyStoreId>
</paws:MovePolicyContainerResponse>
```

13.4 Exceptions

In the event that a PAWS instance encounters an error parsing a MovePolicyContainer request, it shall raise an **OperationParsingFailed** exception as described in section 6.6.

In the event that a PAWS instance encounters an error processing a MovePolicyContainer request, it shall raise an **OperationProcessingFailed** exception as described in section 6.6.

In the event that a PAWS instance is requested to move a policy container from a policy store that is not known to the PAWS instance, it shall raise a **SourcePolicyStoreUnknown** exception as described in section 6.6.

In the event that a PAWS instance is requested to move a policy container to a policy store that is not known to the PAWS instance, it shall raise a **DestinationPolicyStoreUnknown** exception as described in section 6.6.

In the event that a PAWS instance is requested to move a nonexistent policy container from a policy store, it shall raise a **SourcePolicyContainerUnknown** exception as described in section 6.6.

In the event that a PAWS instance is requested to move a policy container with its existing or newly defined name to a policy store that already contains a policy container with such a name, it shall raise a **PolicyContainerAlreadyExists** exception as described in section 6.6.

In the event that a PAWS instance is requested to create a policy container within the destination policy store with an invalid name (i.e. not URN format compliant), it shall raise a **PolicyContainerNameInvalid** exception as described in section 6.6.

In the event that a PAWS instance is requested to operate on a source policy store locked by some other user, it shall raise **EntityLocked** exceptions for the locked entity as described in section 6.6.

In the event that a PAWS instance is requested to operate on a destination policy store locked by some other user, it shall raise **EntityLocked** exceptions for the locked entity as described in section 6.6.

In the event that a PAWS instance is requested to move a policy container locked by some other user, it shall raise an **EntityLocked** exception as described in section 6.6.

Note: It does not constitute an error if a user tries to move a policy container from and to a policy store that has some contained containers (with exception of the one that is operated on) or policy trees that are locked (see section 15 for details on the lock concept).

14 DeletePolicyContainer operation

14.1 Introduction

The DeletePolicyContainer operation deletes an (Geo)XACML policy container from one of the policy stores provided by the PAWS instance.

In cases of concurrent policy administration by multiple administrators, it is highly recommended to lock the policy store upfront the DeletePolicyContainer operation to avoid unwanted side effects through concurrent transactions (see section 15 for details).

14.2 Request

14.2.1 XML encoding and semantics

The following XML schema fragment defines the XML-encoding of the DeletePolicyContainer request:

```
<xsd:element name="DeletePolicyContainer" type="paws:DeletePolicyContainerType"/>
<xsd:complexType name="DeletePolicyContainerType">
  <xsd:complexContent>
    <xsd:extension base="paws:BaseRequestType">
      <xsd:sequence minOccurs="1" maxOccurs="1">
        <xsd:element name="PolicyStoreId" type="paws:urnType"/>
        <xsd:element name="PolicyContainerId" type="paws:urnType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The base type, paws: BaseRequestType is defined in section 7.2.

14.2.1.1 PolicyStoreId parameter (mandatory)

The value of the PolicyStoreId parameter shall be a valid URN (cp. 6.5) and specifies the policy store from which the policy container shall be deleted.

14.2.1.2 PolicyContainerId parameter (mandatory)

The value of the PolicyContainerId parameter shall be a valid URN (cp. 6.5) and defines the name of the to be deleted policy container.

14.2.2 Example

The XML document below shows a DeletePolicyContainer request. The effect of this request is to delete a policy container named "urn:mydomain:MyFirstContainer" from the policy store "urn:mydomain:MyFirstPolicyStore".

```
<paws:DeletePolicyContainer service="PAWS" version="1.0.0" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
</paws:DeletePolicyContainer>
```

14.3 Response

14.3.1 XML encoding and semantics

The root element of the response to a DeletePolicyContainer request is the paws:DeletePolicyContainerResponse element which is declared by the following XML Schema fragment:

```
<xsd:element name="DeletePolicyContainerResponse" type="paws:DeletePolicyContainerResponseType"/>
<xsd:complexType name="DeletePolicyContainerResponseType">
  <xsd:complexContent>
    <xsd:extension base="paws:BaseResponseType">
      <xsd:sequence minOccurs="1" maxOccurs="1">
        <xsd:element name="PolicyStoreId" type="paws:urnType"/>
        <xsd:element name="PolicyContainerId" type="paws:urnType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The base type, paws:BaseResponseType is defined in section 8.2.

The DeletePolicyContainerResponse document shall contain the same elements as the corresponding request in case of a successful termination of the requested delete operation. In all other situations, an exception report shall be generated (see section 14.413.4).

14.3.2 Example

The XML document below shows a valid response to a DeletePolicyContainer request. The response represents an acknowledgement that the policy container "urn:mydomain:MyFirstContainer" has been deleted successfully from the policy store "urn:mydomain:MyFirstPolicyStore".

```
<paws:DeletePolicyContainerResponse timeStamp="2013-04-01T21:00:00Z" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
</paws:DeletePolicyContainerResponse>
```

14.4 Exceptions

In the event that a PAWS instance encounters an error parsing a DeletePolicyContainer request, it shall raise an **OperationParsingFailed** exception as described in section 6.6.

In the event that a PAWS instance encounters an error processing a DeletePolicyContainer request, it shall raise an **OperationProcessingFailed** exception as described in section 6.6.

In the event that a PAWS instance is requested to delete a policy container from a policy store that is not known to the PAWS instance, it shall raise a **PolicyStoreUnknown** exception as described in section 6.6.

In the event that a PAWS instance is requested to delete a nonexistent policy container from a policy store, it shall raise a **PolicyContainerUnknown** exception as described in section 6.6.

In the event that a PAWS instance is requested to operate on a policy store locked by some other user, it shall raise an **EntityLocked** exception as described in section 6.6.

Note: It does not constitute an error if a user tries to delete a container within a policy store who has some contained containers (with exception of the one that is operated on) or policy trees that are locked (see section 15 for details on the lock concept).

15 CreateLock operation

15.1 Introduction

Through a CreateLock request, policy administrators can either lock a policy store, a policy container or a complete (Geo)XACML policy tree, starting from its root element down to all leaf elements. For the latter case, it is important to highlight that the lock covers the whole tree but is not effective across reference boundaries.

In general, a user can set multiple locks at the same time. However, no lockable entity (i.e. a policy store, a policy container and an XACML policy tree) can be locked by multiple locks at the same time (see section 6.7).

Setting a lock on a lockable entity ensures exclusive access to the locked entity and all its “ancestors” for subsequent transactional and non-transactional operations. Every lock shall be bound to the user context of the currently interacting user. This implies that upfront locking some sort of user authentication must take place. After successful authentication, the user can use all its previously created locks (persistently stored in its user context), add new locks or release locks as needed.

It is the duty of the policy administrator to set locks appropriately depending on the intended operation. This includes that administrators need to ensure that a lock achieves the needed exclusive access without unnecessarily influencing other administrators. Note that through appropriate policy structuring measures based on connected policy trees, concurrent administration by multiple administrators with significantly reduced influencing can be achieved.

15.2 Request

15.2.1 XML encoding and semantics

The following XML Schema fragment defines the mandatory XML encoding of CreateLock requests:

```
<xsd:element name="CreateLock" type="paws:CreateLockType"/>
<xsd:complexType name="CreateLockType">
  <xsd:complexContent>
    <xsd:extension base="paws:BaseRequestType">
      <xsd:sequence minOccurs="1" maxOccurs="1">
        <xsd:element name="PolicyStoreId" type="paws:urnType" minOccurs="0"/>
        <xsd:element name="PolicyContainerId" type="paws:urnType" minOccurs="0"/>
        <xsd:element ref="paws:Query" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="expiry" type="xsd:positiveInteger" use="optional" />
      <xsd:attribute name="expiryDate" type="xsd:dateTime" use="optional" />
      <xsd:attribute name="lockId" type="xsd:string" use="optional"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The base type, paws: BaseRequestType is defined in section 7.2.

15.2.1.1 PolicyStoreId parameter (optional)

The value of the PolicyStoreId parameter shall be a valid URN (cp. 6.5) and either specifies

- the to be locked policy store (if the PolicyContainerId parameter is absent),
- the policy store to which the to be locked policy container belongs to (if the Query parameter is absent) or
- the policy store to which the policy container belongs to, whose contained (GEO)XACML policy shall be locked.

The PolicyStoreId parameter shall never be empty and shall only be missing in cases where the CreateLock request is used to reset the expiry time/date of a lock (see section 15.2.1.6).

15.2.1.2 PolicyContainerId parameter (optional)

The value of the PolicyContainerId parameter shall be a valid URN (cp. 6.5) and specifies

- the name of the to be locked policy container (if the Query parameter is absent or empty) or
- the name of the policy container whose contained (GEO)XACML policy shall be locked.

If the PolicyContainerId parameter is absent, there shall be no paws:Query element in the Lock request. The absence of the PolicyContainerId parameter expresses a CreateLock operation on a complete policy store.

If the PolicyContainerId parameter is present and no paws:Query element exists then this expresses a CreateLock operation referring to a policy container.

If the PolicyContainerId parameter is present and if there is a non-empty paws:Query element (pointing to an (GEO)XACML policy root element – see section 15.2.1.3) then this expresses a CreateLock operation referring to a complete (GEO)XACML policy tree (contained within the specified policy store and container).

15.2.1.3 Query parameter (optional)

The value of the Query parameter shall be a valid XPath 2.0 expression (see section 7.3) and defines the to be locked (GEO)XACML policy tree. The XPath expression value shall point to exactly one policy root element (i.e. an xacml:PolicySet or xacml:Policy element without parent element) in the specified policy container (cp. PolicyContainerId parameter). The lock covers all direct ancestors of the selected policy root element.

It is important to highlight that the effect of any CreateLock operation shall not spread over XACML PolicySetReference and PolicyReference elements. To lock a complete XACML policy that is represented through an XACML forest – i.e. XACML trees linked through XACML policy reference elements – administrators are required to lock the required XACML trees of the forest-based XACML policy.

15.2.1.4 expiry parameter (optional)

The expiry parameter can be used to set a limit on how long a PAWS instance shall hold a lock on a lockable entity. Using the expiry parameter the expiry limit shall be specified in number of seconds.

Every CreateLock request shall either exclusively contain (i.e. XOR) an expiry or an expiryDate parameter (see below).

The expiry timer shall be started once the entire lock request has been processed and the lock response has been completely transmitted to the client.

Once the specified time period has elapsed, the PAWS shall release the lock.

If there is no (or an empty) expiry parameter and no (or an empty) expiryDate parameter (see section 15.2.1.5) in the CreateLock request the PAWS shall create a persistent lock, i.e. a lock that is active for an unlimited time period.

The expiry time of an existing lock can be extended as describe in section 15.2.1.6.

15.2.1.5 expiryDate parameter (optional)

To set a time limited lock one can use the expiryDate parameter as alternative to the expiry parameter. Hence every CreateLock request shall either exclusively contain an expiry or an expiryDate parameter. The value of this parameter shall be ISO 8601 conformant (i.e. xsd:dateTime)

Oposed to the expiry parameter that allows to set a time limited lock by specifying a number of seconds, the expiryDate parameter allows to set the expiry date of a lock through a xsd:dateTime value. As soon as the system clock of the PAWS instance reaches the specifyed xsd:dateTime value the PAWS instance shall release the lock.

If there is no (or an empty) expiry parameter and no (or an empty) expiryDate parameter (see section 15.2.1.5) in the CreateLock request the PAWS shall create a persistent lock, i.e. a lock that is active for an unlimited time period.

The expiry time of an existing lock can be extended as describe in section 15.2.1.6.

15.2.1.6 lockId parameter (optional)

A lockId parameter specified within a CreateLock request shall be used to reset the expiry time/date of an existing lock.

EXAMPLE 1 The following Lock request resets the lock expiry of the specified lock (lockId="1234") to 600 seconds:

```
<paws:CreateLock lockId="1234" expiry="600"/>
```

The expiry time/date shall replace the currently set expiry time/date. It shall not be regarded as an error if a second based expiry time is reset to an xsd:dateTime value based expiry date.

EXAMPLE 2 If a lock was originally acquired for 300 seconds, and after 50 seconds, the lock expiry is reset to 600 seconds, the lock shall remain valid for an additional 600 seconds.

15.2.2 Examples

Example 1: Locking of a policy store

The XML document below shows a CreateLock request on a policy store. The effect of this request is to lock a policy store named "urn:mydomain:MyFirstPolicyStore" from for subsequent exclusive access for unlimited time (cp missing expiry and expiryDate parameter).

```
<paws:CreateLock service="PAWS" version="1.0.0"...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
</paws:CreateLock>
```

Example 2: Locking of a policy container

The XML document below shows a CreateLock request on a policy container. The effect of this request is to lock a policy container named "urn:mydomain:MyFirstContainer" from the policy store "urn:mydomain:MyFirstPolicyStore" for subsequent exclusive access. The expiry parameter sets the initial validity period for the lock to 120 seconds.

```
<paws:CreateLock service="PAWS" version="1.0.0" expiry="120" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
</paws:CreateLock>
```

Example 3: Locking of a policy tree

The XML document below shows a CreateLock request on a XACML policy tree. The effect of this request is to lock a XACML policy tree starting from its root element node as identified by the Query parameter's XPath expression (cp /xacml:PolicySet[xacml:PolicySetId="urn:mydomain:12345"]). The PolicyStoreId and PolicyContainerId parameter specify that the PolicySet element node with ID "urn:mydomain:12345" from the policy store "urn:mydomain:MyFirstPolicyStore" and policy container "urn:mydomain:MyFirstContainer" is the starting point of the lock. As defined in section 15 and 6.7 the lock affects all direct ancestors of the selected policy root element and is not effective across XACML Policy and PolicySet reference elements. The expiry parameter defines the initial validity period of 120 seconds for the lock.

```
<paws:CreateLock service="PAWS" version="1.0.0" expiry="120" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
  <paws:Query namespace="xmlns:xacml=urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
      /xacml:PolicySet[xacml:PolicySetId="urn:mydomain:12345"]
  </paws:Query>
</paws:CreateLock>
```

15.3 Response

15.3.1 XML encoding and semantics

The root element of the response to a CreateLock request is the paws:CreateLockResponse element which is declared by the following XML Schema fragment:

```
<xsd:element name="CreateLockResponse" type="paws:CreateLockResponseType"/>
<xsd:complexType name="CreateLockResponseType">
  <xsd:complexContent>
    <xsd:extension base="paws:BaseResponseType">
      <xsd:sequence minOccurs="1" maxOccurs="1">
        <xsd:element name="PolicyStoreId" type="paws:urnType"/>
        <xsd:element name="PolicyContainerId" type="paws:urnType" minOccurs="0"/>
        <xsd:element ref="paws:Query" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="expiry" type="xsd:positiveInteger" default="200" use="optional" />
      <xsd:attribute name="expiryDate" type="xsd:dateTime" use="optional" />
      <xsd:attribute name="lockId" type="xsd:string" use="required"/>
      <xsd:attribute name="lockCreationDateTime" type="xsd:dateTime" use="required" />
    </xsd:extension>
  </xsd:complexContent>
```

```
</xsd:complexType>
```

The base type, paws:BaseResponseType is defined in section 8.2.

In case of a successful CreateLock operation a CreateLockResponse document shall be generated. This document contains a copy of the PolicyStoreId, PolicyContainerId and Query element of the original request (if they were present). The expiry or expiryDate parameter of the request (if present) shall also be present in the response. The mandatory LockId XML attribute in the CreateLockResponse element shall indicate the lock identifier for the created lock. A client application may remember and use this lock identifier to reset the expiry time/date or to release the lock at a certain point in time. Further the mandatory lockCreationDateTime XML attribute in the CreateLockResponse element shall specify the time the lock was set by the PAWS instance.

No restrictions are made about the string format of the lock identifier. The only requirement is that the lockId values can be expressed in the character set of the transaction request (e.g. as string encoded UUID Version 5 values.

15.3.2 Examples

Example 1: Response to example CreateLock request 1 (cp. 15.2.2)

The XML document below shows a valid response to a CreateLock request. The response represents an acknowledgement that the policy store "urn:mydomain:MyFirstPolicyStore" has been locked successfully for an unlimited time.

```
<paws:CreateLockResponse timeStamp="2013-04-01T21:00:01Z" lockId="550e8400-e29b-11d4-a716-
       446655440001" lockCreationDateTime="2013-04-01T21:00:00Z" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
</paws:CreateLockResponse>
```

Example 2: Response to example CreateLock request 2 (cp. 15.2.2)

The XML document below shows a valid response to a CreateLock request. The response represents an acknowledgement that the policy container "urn:mydomain:MyFirstContainer" within the policy store "urn:mydomain:MyFirstPolicyStore" has been locked successfully for the enxt 120 seconds.

```
<paws:CreateLockResponse timeStamp="2013-04-01T21:00:01Z" lockId="550e8400-e29b-11d4-a716-
446655440002" expiry="120" lockCreationDateTime="2013-04-01T21:00:00Z"...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
</paws:CreateLockResponse>
```

Example 3: Response to example CreateLock request 3 (cp. 15.2.2)

The XML document below shows a valid response to a CreateLock request. The response represents an acknowledgement that the XACML policy tree starting from the root xacml:PolicySet element with a PolicySetId equal to "urn:mydomain:12345" within the policy container "urn:mydomain:MyFirstContainer" has been locked successfully.

```
<paws:CreateLockResponse timeStamp="2013-04-01T21:00:01Z" lockId="550e8400-e29b-11d4-a716-
446655440003" expiry="120" lockCreationDateTime="2013-04-01T21:00:00Z"...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
```

```
      /xacml:PolicySet[xacml:PolicySetId="urn:mydomain:12345"]
  </paws:Query>
</paws:CreateLockResponse>
```

15.4 Exceptions

In the event that a PAWS instance encounters an error parsing a CreateLock request, it shall raise an **OperationParsingFailed** exception as described in section 6.6.

In the event that a PAWS instance encounters an error processing a CreateLock request, it shall raise an **OperationProcessingFailed** exception as described in section 6.6.

In the event that a PAWS instance is requested to lock a policy store not known to the PAWS instance, it shall raise a **PolicyStoreUnknown** exception as described in section 6.6.

In the event that a PAWS instance is requested to lock a policy container not known to the PAWS instance, it shall raise a **PolicyContainerUnknown** exception as described in section 6.6.

In the event that a PAWS instance is requested to lock a policy tree not known to the PAWS instance (e.g. by pointing to an unknown or non-root policy element), it shall raise a **PolicyRootUnknown** exception as described in section 6.6.

In the event that a PAWS instance is requested to lock a policy store already locked by some other user, it shall raise an **EntityLocked** exception as described in section 6.6.

In the event that a PAWS instance is requested to lock a policy container already locked by some other user, it shall raise an **EntityLocked** exception as described in section 6.6.

In the event that a PAWS instance is requested to lock a policy store which contains policy containers already locked by some other user(s), it shall raise an **EntityLocked** exception as described in section 6.6.

In the event that a PAWS instance is requested to lock a policy store with containers that contain XACML policy trees already locked by some other user(s), it shall raise an **EntityLocked** exception as described in section 6.6.

In the event that a PAWS instance is requested to lock a policy container where the enclosing policy store is already locked by some other user, it shall raise an **EntityLocked** exception as described in section 6.6.

In the event that a PAWS instance is requested to lock a policy container already locked by some other user, it shall raise an **EntityLocked** exception as described in section 6.6.

In the event that a PAWS instance is requested to lock a policy container that contains XACML policy trees already locked by some other user(s), it shall raise an **EntityLocked** exception as described in section 6.6.

In the event that a PAWS instance is requested to lock a policy tree whos enclosing policy store or policy container is already locked by some other user, it shall raise an **EntityLocked** exception as described in section 6.6.

In the event that a PAWS instance is requested to lock a policy tree already locked by some other user, it shall raise an **EntityLocked** exception as described in section 6.6.

In the event that a PAWS instance is requested to reset the expiry time/date of a lock not known to the PAWS instance or not know in the currently relevant user context, it shall raise a **LockIdParameterValueUnknown** exception as described in section 6.6.

In the event that a PAWS instance is not able to extend the expiry value of a lock (e.g. because of an invalid expiry or expiryDate parameter value), it shall raise a **LockExpiryUpdateFailed** exception as described in section 6.6.

In the event that a PAWS instance is requested to release a lock but cannot retrieve the relevant user context because the interacting user has not logged in, it shall raise an **AuthenticationRequired** exception as described in section 6.6.

16 ShowLocks operation

16.1 Introduction

Through a ShowLocks request, policy administrators can query all of their currently active locks including some metadata related to these locks (e.g. lock definition parameters including its expiry time/date definition and the time the lock was set). Further, the ShowLocks operation supports that an administrator can also query the locks set by other users (cp. SubjectId parameter). This feature e.g. allows superadmins to identify and release active but orphan locks set by other administrators. For obvious reasons the use of this feature needs to be restricted accordingly through an access control system for the PAWS itself (e.g. deny ShowLocks requests with a set SubjectId parameter).

16.2 Request

16.2.1 XML encoding and semantics

The following XML Schema fragment defines the mandatory XML encoding of ShowLocks requests:

```
<xsd:element name="ShowLocks" type="paws:ShowLocksType"/>
  <xsd:complexType name="ShowLocksType">
    <xsd:complexContent>
      <xsd:extension base="paws:BaseRequestType">
        <xsd:sequence minOccurs="1" maxOccurs="1">
          <xsd:sequence>
              <xsd:element name="SubjectId" type="paws:SubjectIdType" minOccurs="0"/>
          </xsd:sequence>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
```

The base type, paws: BaseRequestType is defined in section 7.2.

16.2.1.1 SubjectId parameter (optional)

The typical usage of the ShowLocks operation does not need to include a SubjectId parameter. If the SubjectId is not present, the PAWS shall return all locks currently set in the user context of the currently interacting user. This implies that the user needs to be logged in before querying a list of its active locks (see section 15 for details on the authentication and user context assumption).

If one or multiple SubjectId parameters are present, their values shall be either valid, unique subject identifiers that can be resolved by the PAWS instance to the corresponding registered users and its associated (persisted) user context information. Alternatively, the SubjectId parameter value can be set to “*”. By doing so the administrator requests to get a list of all active locks of all users of the PAWS instance.

16.2.2 Example

The XML document below shows a ShowLocks request of a user currently logged in under the username “testUser1”. The effect of this request is to query a list of all active locks set by the user “testUser1”.

```
<paws:ShowLocks service="PAWS" version="1.0.0".../>
```

16.3 Response

16.3.1 XML encoding and semantics

The root element of the response to a ShowLocks request is the paws:ShowLocksResponse element which is declared by the following XML schema fragment:

```
<xsd:element name="ShowLocksResponse" type="paws:ShowLocksResponseType"/>
<xsd:complexType name="ShowLocksResponseType">
  <xsd:complexContent>
    <xsd:extension base="paws:BaseResponseType">
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="Lock" type="paws:LockTypeExtended"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="LockTypeExtended">
  <xsd:complexContent>
    <xsd:extension base="paws:BaseRequestType">
      <xsd:sequence minOccurs="1" maxOccurs="1">
        <xsd:element name="PolicyStoreId" type="paws:urnType" minOccurs="0"/>
        <xsd:element name="PolicyContainerId" type="paws:urnType" minOccurs="0"/>
        <xsd:element ref="paws:Query" minOccurs="0"/>
        <xsd:sequence>
          <xsd:element name="SubjectId" type="paws:SubjectIdType" minOccurs="0"/>
        </xsd:sequence>
      </xsd:sequence>
      <xsd:attribute name="expiry" type="xsd:positiveInteger" default="200" use="optional"/>
      <xsd:attribute name="expiryDate" type="xsd:dateTime" use="optional"/>
      <xsd:attribute name="lockId" type="xsd:string" use="required"/>
      <xsd:attribute name="lockCreationDateTime" type="xsd:dateTime" use="required"/>
      <xsd:attribute name="lockOwner" type="xsd:string" use="required"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The base type, paws:BaseResponseType is defined in section 8.2.

In case of a sucessfull, ShowLocks operation a ShowLocksResponse document shall be generated. This document contains a list of <paws:Lock> elements that describe the active locks set by the interacting or specified users (see section 16.2.1.1). Each <paws:Lock> element describes the parameters used to create the lock. Further every <paws:Lock> element contains the lockId, lockCreationDateTime and lockOwner XML attributes, describing the lock's identifier, the date and time the lock was generated as well as the identifier of the user that created the lock.

If no user context exists (e.g. the user is not logged in), the response does contain an empty list of locks.

16.3.2 Example

The XML document below shows a valid response to the ShowLocks request defined in section 16.2.2. The response expresses that the currently interacting user "testUser1" has one active, not expiring lock (cp. the missing expiry and expiryDate attribute) set for the policy store "urn:mydomain:MyFirstPolicyStore" on april 1st 2013 at 9 pm UTC.

```
<paws:ShowLocksResponse timeStamp="2013-04-05T22:00:10Z" ...>
```

```
  <paws:Lock lockId="550e8400-e29b-11d4-a716-446655440001" lockCreationDateTime="2013-04-
01T21:00:00Z" lockOwner="testUser1">
    <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  </paws:Lock>
</paws:ShowLocksResponse>
```

16.4 Exceptions

In the event that a PAWS instance encounters an error parsing a ShowLocks request, it shall raise an **OperationParsingFailed** exception as described in section 6.6.

In the event that a PAWS instance encounters an error processing a ShowLocks request, it shall raise an **OperationProcessingFailed** exception as described in section 6.6.

In the event that a PAWS instance is requested to show the locks of a user unknown to the PAWS instance, it shall raise a **SubjectIdUnknown** exception as described in section 6.6.

17 ReleaseLock operation

17.1 Introduction

Through a ReleaseLock request, policy administrators can release i.e. discharge one of the active locks. Releasing a lock represents a commit statement for all conducted transaction during the lock period.

The ReleaseLock operation supports that an administrator can release either one of its active locks or one of the locks set by other users (cp. SubjectId parameter). This feature e.g. allows super-admins to release active but orphan locks set by administrators that are not part of the company any more. For obvious reasons the use of this feature needs to be restricted accordingly through an access control system for the PAWS itself (e.g. deny ReleaseLock requests with a set SubjectId parameter).

17.2 Request

17.2.1 XML encoding and semantics

The following XML Schema fragment defines the mandatory XML encoding of ReleaseLock requests:

```
<xsd:element name="ReleaseLock" type="paws:ReleaseLockType"/>
<xsd:complexType name="ReleaseLockType">
  <xsd:complexContent>
   <xsd:extension base="paws:BaseRequestType">
     <xsd:sequence minOccurs="0" maxOccurs="1">
       <xsd:element name="SubjectId" type="xsd:string" minOccurs="0"/>
     </xsd:sequence>
     <xsd:attribute name="lockId" type="xsd:string" use="required"/>
     <xsd:attribute name="rollback" type="xsd:boolean" use="optional" default="false"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The base type, paws: BaseRequestType is defined in section 7.2.

17.2.1.1 SubjectId parameter (optional)

The typical usage of the ReleaseLock operation does not need to include a SubjectId parameter. If the SubjectId is not present, the PAWS shall release the lock as specified by the lockId parameter (see section 17.2.1.2) – of the currently interacting user. This implies that the user needs to be logged in before being able to release one of its active locks (see section 15 for details on the authentication and user context assumption).

If the string based SubjectId parameter is present, the user requests to release the lock – as specified by the lockId parameter (see section 17.2.1.2) - of the user identified by the SubjectId parameter value.

17.2.1.2 lockId parameter (mandatory)

A lockId parameter specified within a ReleaseLock request shall be used to specify the lock to be released.

17.2.1.3 rollback parameter (optional)

Setting the rollback parameter to “true” within a ReleaseLock request has the effect that all transactions performed during the lock period will be rolled back. In this case, the state of the locked entity after releasing the lock is equal to its state at the time setting the lock.

The rollback parameter is absent or set to false the release of a lock commits / persists all changes requested during the lock period and they become visible to other administrators.

17.2.2 Example

The XML document below shows a ReleaseLock request of a user currently logged in under the username “testUser1”. The effect of this request is that the lock with id “550e8400-e29b-11d4-a716-446655440001” set by the user “testUser1” will be released.

```
<paws:ReleaseLock lockId="550e8400-e29b-11d4-a716-446655440001" service="PAWS" version="1.0.0".../>
```

17.3 Response

17.3.1 XML encoding and semantics

The root element of the response to a ReleaseLock request is the paws:ReleaseLockResponse element which is declared by the following XML schema fragment:

```
<xsd:element name="ReleaselockResponse" type="paws:ReleaseLockResponseType"/>
<xsd:complexType name="ReleaseLockResponseType">
  <xsd:complexContent>
    <xsd:extension base="paws:BaseResponseType">
      <xsd:sequence minOccurs="1" maxOccurs="1">
        <xsd:element name="Lock" type="paws:LockTypeExtended"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The base type, paws:BaseResponseType is defined in section 8.2.

The type, paws:LockTypeExtended is defined in section 16.3.1.

In case of a successful, ReleaseLock operation a ReleaseLockResponse document shall be returned. This document approves that the lock as described in the <paws:Lock> element (contained in the response) was released.

17.3.2 Examples

The XML document below shows a valid response to the ReleaseLock request defined in section 17.2.2. The response expresses that the currently interacting user “testUser1” has released its lock with id “550e8400-e29b-11d4-a716-446655440001”.

```
<paws:ReleaseLockResponse timeStamp="2013-04-05T22:00:10" ...>
  <paws:Lock lockId="550e8400-e29b-11d4-a716-446655440001" lockCreationDateTime="2013-04-
01T21:00:00Z" lockOwner="testUser1">
    <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  </paws:Lock>
</paws:ReleaseLockResponse>
```

17.4 Exceptions

In the event that a PAWS instance encounters an error parsing a ReleaseLock request, it shall raise an **OperationParsingFailed** exception as described in section 6.6.

In the event that a PAWS instance encounters an error processing a ReleaseLock request, it shall raise an **OperationProcessingFailed** exception as described in section 6.6.

In the event that a PAWS instance is requested to release a lock of a user unknown to the PAWS instance, it shall raise a **SubjectIdUnknown** exception as described in section 6.6.

In the event that a PAWS instance is requested to release a lock not known to the PAWS instance or not know in the currently relevant user context, it shall raise a **LockIdParameterValueUnknown** exception as described in section 6.6.

In the event that a PAWS instance is requested to release a lock but cannot retrieve the relevant user context because the interacting user has not logged in, it shall raise an **AuthenticationRequired** exception as described in section 6.6.

18 InsertPolicyElement operation

18.1 Introduction

Through InsertPolicyElement requests, policy administrators can create arbitrary complex XACML policy trees.

Note that in multi administrator scenarios it is highly recommended to set a lock before editing an existing policy tree. Because locks are assigned to a user context, a lockId does not need to be specified when performing any transactional operation on policy trees. To benefit from the active locks there is the prerequisite to be logged in, using a suitable user account.

As pointed out in section 6.3 it is recommended to use a unique naming schema for all XACML policy element identifiers within a container (e.g. based on URNs with appended random SHA-256 hash values like “urn:mydomain:myFirstPolicySet:68ac906495480a340...”).

18.2 Request

18.2.1 XML encoding and semantics

The following XML Schema fragment defines the mandatory XML encoding of InsertPolicyElement requests:

```
<xsd:element name="InsertPolicyElement" type="paws:InsertPolicyElementType"/>
<xsd:complexType name="InsertPolicyElementType">
  <xsd:complexContent>
    <xsd:extension base="paws:BaseRequestType">
      <xsd:sequence minOccurs="1" maxOccurs="1">
        <xsd:element name="PolicyStoreId" type="paws:urnType"/>
        <xsd:element name="PolicyContainerId" type="paws:urnType"/>
        <xsd:element ref="paws:Query" minOccurs="0" />
        <xsd:element name="InsertStyle" type="paws:InsertStyleType"/>
        <xsd:element name="XacmlPolicyElement" type="paws:BasePAWS-XACML-PolicyElementDataType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="BasePAWS-XACML-PolicyElementDataType">
  <xsd:choice>
    <xsd:element ref="xacml:PolicySet"/>
    <xsd:element ref="xacml:Policy"/>
    <xsd:element ref="xacml:Rule"/>
    <xsd:element ref="xacml:PolicySetIdReference"/>
    <xsd:element ref="xacml:PolicyIdReference"/>
  </xsd:choice>
</xsd:complexType>

<xsd:simpleType name="InsertStyleType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="as-new-last-child"/>
    <xsd:enumeration value="as-sibling-before"/>
    <xsd:enumeration value="as-sibling-after"/>
  </xsd:restriction>
</xsd:simpleType>
```

Through a single InsertPolicyElement request, one can neither insert exactly one of the following XACML policy elements at one or multiple locations in an existing policy tree of a container or create a new policy tree within the specified container:

- <xacml:PolicySet>...
- <xacml:Policy>...
- <xacml:PolicySetIdReference>...
- <xacml:PolicyIdReference>...
- <xacml:Rule>...

Other types of XACML elements can only be added to an existing XACML policy tree as children of elements of the types listed above.

The base type paws: BaseRequestType is defined in section 7.2.

18.2.1.1 PolicyStoreId parameter (mandatory)

The value of the PolicyStoreId parameter shall be a valid URN (cp. 6.5) and specifies the policy store to which the new policy element shall be added.

18.2.1.2 PolicyContainerId parameter (mandatory)

The value of the PolicyContainerId parameter shall be a valid URN (cp. 6.5) and defines the name of the policy container to which the new policy element shall be added.

18.2.1.3 Query parameter (optional)

The non-empty value of the Query parameter shall be a valid XPath 2.0 expression (cp. 7.3) and defines one or multiple nodes below or next to (cp. 18.2.1.4) the to be inserted XACML policy element node.

A missing <paws:Query> element implies that the XACML policy element shall be a new policy root element in the corresponding container. Hence a missing paws:Query element in an InsertPolicyElement request implies that the to be inserted XACML policy element shall be either of type xacml:PolicySet or xacml:Policy. In this case, the value of the InsertStyle parameter is not of relevance but shall however be set to “as-new-last-child” for uniformity reasons.

18.2.1.4 InsertStyle parameter (mandatory)

The value of the mandatory InsertStyle parameter can have one of the following values:

- “as-new-last-child”
- “as-sibling-before”
- “as-sibling-after

A InsertStyle parameter value equal to “as-new-last-child” indicates that the policy element to be inserted shall be added as new last child of the element node(s) identified by the paws:Query element’s XPath expression.

A InsertStyle parameter value equal to “as-sibling-before” indicates that the to be inserted policy element shall be added as new sibling before the element node(s) identified by the paws:Query element’s XPath expression.

A InsertStyle parameter value equal to “as-sibling-after” indicates that the to be inserted policy element shall be added as new sibling after the element node(s) identified by the paws:Query element’s XPath expression.

Note: The option to control how new policy elements shall be added relative to their siblings is important, as XACML conflict resolution algorithms (e.g. first-applicable) are dependent on the order of policy elements in the policy tree.

18.2.1.5 XacmlPolicyElement parameter (mandatory)

The XacmlPolicyElement parameter defines the one and only to be inserted XACML policy element. This element will get appended as new last child of, as sibling before or as sibling after (cp. 18.2.1.4) the node(s) identified by the XPath expression in the Query parameter (cp. 18.2.1.3 and 7.3). As mentioned above the insertable elements shall only be of type: xacml:PolicySet, xacml:Policy, xacml:Rule, xacml:PolicySetIdReference and xacml:PolicyIdReference.

Identifieres of XACML policy elements within the same policy container shall be unique. This implies that a PAWS instance shall report an exception in case a user tries to create a policy element with an identifier equal to an already existing policy element identifier within that container.

Note: It needs to be highlighted that through a PAWS InsertPolicyElement operation one can add <PolicySet> and <Policy> elements that already have children of type <PolicySet>, <Policy> or <Rule>. It is however highly recommended to deny through an administrative access control rule (protecting the PAWS itself) that <PolicySet> and <Policy> elements can be inserted that have children of type <PolicySet>, <Policy>, <PolicySetIdReference>, <PolicyIdReference> or <Rule>. Such an administrative rule will imply that XACML policy trees have to be build step-by-step – i.e. sucessive – through separate InsertPolicyElement requests. A strict enforcement of such a successive policy generation is an essential property to be able to support the simple definition of rich administrative rights.

18.2.2 Example

The XML document below shows an InsertPolicyElement request on a policy container and store respectively. The effect of this request is to create a new policy root element of type xacml:PolicySet with Id “urn:lam:root-ps:layer:1:12345” within the policy container named "urn:mydomain:MyFirstContainer" within the policy store "urn:mydomain:MyFirstPolicyStore". Note that the missing paws:Query element is the reason why this InsertPolicyRequest does not extend an existing policy tree but adds a new policy tree / root to the container.

To ensure that no unwanted side effects of parallel transactions can occur, the container should have been locked upfront.

```
<paws:InsertPolicyElement service="PAWS" version="1.0.0" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
  <paws:InsertStyle>as-new-last-child</InsertStyle>
  <paws:XacmlPolicyElement>
```

```
    <xacml:PolicySet PolicySetId="urn:lam:root-ps:layer:1:12345" PolicyCombiningAlgId="only-one-
       applicable" Version="0.1" ...>
      <xacml:Target/>
    </xacml:PolicySet>
  </paws:XacmlPolicyElement>
</paws:InsertPolicyElement>
```

18.3 Response

18.3.1 XML encoding and semantics

The root element of the response to an InsertPolicyElement request is the paws:InsertPolicyElementResponse element which is declared by the following XML Schema fragment:

```
<xsd:element name="InsertPolicyElementResponse" type="paws:InsertPolicyElementResponseType"/>
<xsd:complexType name="InsertPolicyElementResponseType">
  <xsd:complexContent>
    <xsd:extension base="paws:BaseResponseType">
      <xsd:sequence minOccurs="1" maxOccurs="1">
        <xsd:element name="PolicyStoreId" type="paws:urnType"/>
        <xsd:element name="PolicyContainerId" type="paws:urnType"/>
        <xsd:element ref="paws:Query" minOccurs="0"/>
        <xsd:element name="InsertStyle" type="paws:InsertStyleType"/>
        <xsd:element name="XacmlPolicyElement" type="paws:BasePAWS-XACML-PolicyElementDataType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The base type, paws:BaseResponseType is defined in section 8.2.

In case of a successful InsertPolicyElement operation a InsertPolicyElementResponse document shall be generated. This document contains a copy of the PolicyStoreId, PolicyContainerId, Query and InsertStyle element of the original request (if present).

18.3.2 Example

The XML document below shows a valid response to an InsertPolicyElement request. The response represents an acknowledgement that a new policy root element in form of a new xacml:PolicySet element (with Id equal "urn:lam:root-ps:layer:1:12345") was successfully added to the policy container "urn:mydomain:MyFirstContainer" within the policy store "urn:mydomain:MyFirstPolicyStore".

```
<paws:InsertPolicyElementResponse timeStamp="2013-04-01T21:00:00Z"...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
  <paws:InsertStyle>as-new-last-child</InsertStyle>
  <paws:XacmlPolicyElement>
    <xacml:PolicySet PolicySetId="urn:lam:root-ps:layer:1:12345" PolicyCombiningAlgId="only-one-
       applicable" Version="0.1"...>
      <xacml:Target/>
    </xacml:PolicySet>
  </paws:XacmlPolicyElement>
</paws:InsertPolicyElementResponse>
```

18.4 Exceptions

In the event that a PAWS instance encounters an error parsing an InsertPolicyElement request, it shall raise an OperationParsingFailed exception as described in section 6.6.

In the event that a PAWS instance encounters an error processing an InsertPolicyElement request, it shall raise an OperationProcessingFailed exception as described in section 6.6

In the event that a PAWS instance is requested to create a policy element within an unknown policy store, it shall raise a **PolicyStoreUnknown** exception as described in section 6.6.

In the event that a PAWS instance is requested to create a policy element within an unknown policy container, it shall raise a **PolicyContainerUnknown** exception as described in section 6.6.

In the event that a PAWS instance is requested to create a policy element with an already used XACML policy element identifier within that container, it shall raise a **PolicyElementIdentifierExists** exception as described in section 6.6.

In the event that a PAWS instance is requested to insert a policy element relative to an undefined location within an existing policy tree, it shall raise a **PolicyElementInsertReferenceNotDefined** exception as described in section 6.6.

In the event that the paws:Query element of the InsertPolicyElement request does not point to one or multiple element nodes of type xacml:PolicySet, xacml:Policy, xacml:Rule, xacml:PolicySetIdReference or xacml:PolicyIdReference, the PAWS instance shall raise a **QueryInvalid** exception as described in section 6.6.

In the event that a PAWS instance is requested to create a policy element at a location within an existing policy tree that does not allow the corresponding node type as one of its children or sibling elements, it shall raise a **PolicyElementInvalidAtDestination** exception as described in section 6.6.

In the event that a PAWS instance is requested to create a policy element that is not of type xacml:PolicySet, xacml:Policy, xacml:Rule, xacml PolicySetReference or xacml:PolicyReference it shall raise a **PolicyElementNotSupported** exception as described in section 6.6.

In the event that a PAWS instance is requested to insert an invalid (i.e. a not XACML schema conformant) XACML policy element of type xacml:PolicySet, xacml:Policy or xacml:Rule, xacml PolicySetReference or xacml:PolicyReference it shall raise a **PolicyElementInvalid** exception as described in section 6.6.

In the event that a PAWS instance is requested to create a policy element in a policy store, policy container or policy tree locked by some other user, it shall raise an **EntityLocked** exception as described in section 6.6.

19 UpdatePolicyElement operation

19.1 Introduction

Through UpdatePolicyElement requests, policy administrators can update policy elements in an existing XACML policy tree.

Note that in multi administrator scenarios it is highly recommended to set a lock before editing an existing policy tree. Because locks are assigned to a user context, a lockId does not need to be specified when performing any transactional operation on policy trees. To benefit from the set locks there is the prerequisite to be loged in, using a suitable user account.

Further it is recommended to use a unique naming schema for all XACML policy element identifiers within a container (e.g. based on URNs with appended random SHA-256 hash values like “urn:mydomain:myFirstPolicySet:68ac906495480a340...”).

19.2 Request

19.2.1 XML encoding and semantics

The following XML Schema fragment defines the mandatory XML encoding of UpdatePolicyElement requests:

```
<xsd:element name="UpdatePolicyElement" type="paws:UpdatePolicyElementType"/>
<xsd:complexType name="UpdatePolicyElementType">
  <xsd:complexContent>
    <xsd:extension base="paws:BaseRequestType">
      <xsd:sequence minOccurs="1" maxOccurs="1">
        <xsd:element name="PolicyStoreId" type="paws:urnType"/>
        <xsd:element name="PolicyContainerId" type="paws:urnType"/>
        <xsd:element ref="paws:Query"/>
        <xsd:element name="XacmlPolicyElement" type="paws:BasePAWS-XACML-PolicyElementDataType"/>
        <xsd:element name="UpdateStyle" type="paws:UpdateStyleType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:complexType name="BasePAWS-XACML-PolicyElementDataType">
  <xsd:choice>
    <xsd:element ref="xacml:PolicySet"/>
    <xsd:element ref="xacml:Policy"/>
    <xsd:element ref="xacml:Rule"/>
    <xsd:element ref="xacml:PolicySetIdReference"/>
    <xsd:element ref="xacml:PolicyIdReference"/>
  </xsd:choice>
</xsd:complexType>

<xsd:simpleType name="UpdateStyleType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="complete"/>
    <xsd:enumeration value="local"/>
  </xsd:restriction>
</xsd:simpleType>
```

Through a single UpdatePolicyElement request, one can update exactly one of the following XACML policy elements at one or multiple locations in existing policy tree of a container:

- <xacml:PolicySet>...

- <xacml:PolicySetIdReference>...
- <xacml:PolicyIdReference>...
- <xacml:Rule>...

The base type paws:BaseRequestType is defined in section 7.2.

19.2.1.1 PolicyStoreId parameter (mandatory)

The value of the PolicyStoreId parameter shall be a valid URN (cp. 6.5) and specifies the policy store in which a policy element shall be updated.

19.2.1.2 PolicyContainerId parameter (mandatory)

The value of the PolicyContainerId parameter shall be a valid URN (cp. 6.5) and defines the name of the policy container in which a policy element shall be updated.

19.2.1.3 Query parameter (mandatory)

The value of the Query parameter shall be a valid XPath 2.0 expression (cp. 7.3) and defines one or multiple nodes that shall be updated by the XACML policy element node defined in the XacmlPolicyElement parameter (cp. 19.2.1.4).

The paws:Query element shall never be missing or empty in an UpdatePolicyElement request and must point to one or multiple element nodes of type xacml:PolicySet, xacml:Policy, xacml:Rule, xacml:PolicySetIdReference or xacml:PolicyIdReference.

19.2.1.4 XacmlPolicyElement parameter (mandatory)

The XacmlPolicyElement parameter defines the updated version of an existing XACML policy element (or at least a part of it). As mentioned above the new update element shall only be of type: xacml:PolicySet, xacml:Policy, xacml:Rule, xacml:PolicySetIdReference and xacml:PolicyIdReference. The details on the update style are defined by the UpdateStyle parameter (cp. 19.2.1.5).

Identifiers of XACML policy elements within the same policy container shall be unique.

Note: It needs to be highlighted that through a PAWS UpdatePolicyElement operation one can add <PolicySet> and <Policy> elements that already have children of type <PolicySet>, <Policy>, <PolicySetIdReference>, <PolicyIdReference> or <Rule>. It is however highly recommended to deny through an administrative access control rule that <PolicySet> and <Policy> elements can be inserted via the update that have children of type <PolicySet>, <Policy> or <Rule>. Such an administrative rule will imply that XACML policy trees have to be build step-by-step – i.e. successive – through separate Insert- or UpdatePolicyElement requests. A strict enforcement of such a successive policy generation is an essential property to be able to support the simple definition of rich administrative rights.

19.2.1.5 UpdateStyle parameter (mandatory)

The value of the mandatory UpdateStyle parameter can have one of the following values:

- “complete”
- “keep-descendants”

A UpdateStyle parameter value equal to “complete” indicates that the to be updated policy element(s) (as identified by the paws:Query element’s XPath expression) and all its descendants shall be removed completely and shall be replaced by the policy element specified in the XacmlPolicyElement parameter.

A UpdateStyle parameter value equal to “keep-descendants” indicates that the to be updated policy element nodes only (as identified by the paws:Query element’s XPath expression) shall be replaced by the policy element specified in the XacmlPolicyElement parameter. In contrast to the “complete” style, the previously existing children of the to be updated element node(s) remain children of the new update node. Previously existing children will remain in order and will be the first children in the subtree of the new node (and its maybe existing children).

The benefit of setting the update style to “keep-descendants” is that update operations close to the root of an XACML policy tree do not imply the burden that previously existing subtrees of the to be updated node must be reinserted by hand after the update.

19.2.2 Example

The XML document below shows an UpdatePolicyElement request on a policy container and store respectively. The effect of this request is to update a policy root element of type xacml:PolicySet with Id “urn:lam:root-ps:layer:1:12345” within the policy container named "urn:mydomain:MyFirstContainer" within the policy store "urn:mydomain:MyFirstPolicyStore" in a way that the version attribute of the policy element is changed to 0.2. Note that because of the UpdateStyle parameter value equal “complete” any previous children of this root policy element would have been deleted.

To ensure that no unwanted side effects of parallel transactions can occur, the policy tree should have been been locked upfront.

```
<paws:UpdatePolicyElement service="PAWS" version="1.0.0" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
  <paws:Query namespace="xmlns:xacml=urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
    /xacml:PolicySet[xacml:PolicySetId="urn:mydomain:12345"]
  </paws:Query>
  <paws:XacmlPolicyElement>
    <xacml:PolicySet PolicySetId="urn:lam:root-ps:layer:1:12345" PolicyCombiningAlgId="only-one-
        applicable" Version="0.2" ...>
      <xacml:Target/>
    </xacml:PolicySet>
  </paws:XacmlPolicyElement>
  <paws:UpdateStyle>complete</paws:UpdateStyle>
</paws:UpdatePolicyElement>
```

19.3 Response

19.3.1 XML encoding and semantics

The root element of the response to a UpdatePolicyElement request is the paws:UpdatePolicyElementResponse element which is declared by the following XML Schema fragment:

```
<xsd:element name="UpdatePolicyElementResponse" type="paws:UpdatePolicyElementResponseType"/>
<xsd:complexType name="UpdatePolicyElementResponseType">
  <xsd:complexContent>
    <xsd:extension base="paws:BaseResponseType">
      <xsd:sequence minOccurs="1" maxOccurs="1">
        <xsd:element name="PolicyStoreId" type="paws:urnType"/>
        <xsd:element name="PolicyContainerId" type="paws:urnType"/>
        <xsd:element ref="paws:Query"/>
        <xsd:element name="XacmlPolicyElement" type="paws:BasePAWS-XACML-PolicyElementDataType"/>
        <xsd:element name="UpdateStyle" type="paws:UpdateStyleType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The base type, paws:BaseResponseType is defined in section 8.2.

In case of a successful UpdatePolicyElement operation, an UpdatePolicyElementResponse document shall be generated. This document contains a copy of the PolicyStoreId, PolicyContainerId, Query, XacmlPolicyElement and UpdateStyle element of the original request.

19.3.2 Example

The XML document below shows a valid response to an UpdatePolicyElement request. The response represents an acknowledgement that policy root element with Id equal to "urn:lam:root-ps:layer:1:12345" was successfully updated by changing the version attribute of the PolicySet to 0.2.

```
<paws:UpdatePolicyElementResponse timeStamp="2013-04-01T21:00:00Z" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
  <paws:Query namespace="xmlns:xacml=urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
    /xacml:PolicySet[xacml:PolicySetId="urn:mydomain:12345"]
  </paws:Query>
  <paws:XacmlPolicyElement>
    <xacml:PolicySet PolicySetId="urn:lam:root-ps:layer:1:12345" PolicyCombiningAlgId="only-one-
        applicable" Version="0.2">
      <xacml:Target/>
    </xacml:PolicySet>
  </paws:XacmlPolicyElement>
  <paws:UpdateStyle>complete</paws:UpdateStyle>
</paws:UpdatePolicyElementResponse>
```

19.4 Exceptions

In the event that a PAWS instance encounters an error parsing an UpdatePolicyElement request, it shall raise an OperationParsingFailed exception as described in section 6.6.

In the event that a PAWS instance encounters an error processing an UpdatePolicyElement request, it shall raise an OperationProcessingFailed exception as described in section 6.6.

In the event that a PAWS instance is requested to update a policy element within an unknown policy store, it shall raise a **PolicyStoreUnknown** exception as described in section 6.6.

In the event that a PAWS instance is requested to update a policy element within an unknown policy container, it shall raise a **PolicyContainerUnknown** exception as described in section 6.6.

In the event that a PAWS instance is requested to update a policy element by an element with an already used XACML policy element identifier, it shall raise a **PolicyElementIdentifierExists** exception as described in section 6.6.

In the event that a PAWS instance is requested to update a policy element relative to an undefined location within an existing policy tree, it shall raise a **PolicyElementUpdateReferenceNotDefined** exception as described in section 6.6.

In the event that a PAWS instance is requested to replace a node by the update policy element at a location that does not allow the corresponding update node type, it shall raise a **PolicyElementInvalidAtDestination** exception as described in section 6.6.

In the event that a PAWS instance is requested to update and/or insert a policy element that is not of type xacml:PolicySet, xacml:Policy, xacml:Rule, xacml PolicySetReference or xacml:PolicyReference, it shall raise a **PolicyElementNotSupported** exception as described in section 6.6.

In the event that a PAWS instance is requested to insert an invalid (i.e. a not XACML schema conformant) XACML policy element of type xacml:PolicySet, xacml:Policy or xacml:Rule, xacml PolicySetReference or xacml:PolicyReference through the update, it shall raise a **PolicyElementInvalid** exception as described in section 6.6.

In the event that a PAWS instance is requested to create a policy element in a policy store, policy container or policy tree locked by some other user, it shall raise an **EntityLocked** exception as described in section 6.6.

In the event that the paws:Query element of the UpdatePolicyElement request is empty or does not point to one or multiple element nodes of type xacml:PolicySet, xacml:Policy, xacml:Rule, xacml:PolicySetIdReference or xacml:PolicyIdReference, the PAWS instance shall raise a **QueryInvalid** exception as described in section 6.6.

20 DeletePolicyElement operation

20.1 Introduction

Through DeletePolicyElement requests policy administrators can delete certain policy elements from XACML policy trees.

Note that in multi administrator scenarios it is highly recommended to set a lock before editing an exisiting policy tree. Because locks are assigned to a user context, a lockId does not need to be specified when performing any transactional operation on policy trees. To benefit from the set locks there is the prerequisite to be loged in, using a suitable user account (see section 6.7 and 15).

20.2 Request

20.2.1 XML encoding and semantics

The following XML Schema fragment defines the mandatory XML encoding of DeletePolicyElement requests:

```
<xsd:element name="DeletePolicyElement" type="paws:DeletePolicyElementType"/>
<xsd:complexType name="DeletePolicyElementType">
  <xsd:complexContent>
    <xsd:extension base="paws:BaseRequestType">
      <xsd:sequence minOccurs="1" maxOccurs="1">
        <xsd:element name="PolicyStoreId" type="paws:urnType"/>
        <xsd:element name="PolicyContainerId" type="paws:urnType"/>
        <xsd:element ref="paws:Query" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

Through a single DeletePolicyElement request one can delete one of the following XACML policy element types at one or multiple locations in an existing policy tree of a container:

- <xacml:PolicySet>...
- <xacml:Policy>...
- <xacml:PolicySetIdReference>...
- <xacml:PolicyIdReference>...
- <xacml:Rule>...

Other types of XACML elements can only be deleted indirectly when deleting one of the node types listed above.

The base type paws: BaseRequestType is defined in section 7.2.

20.2.1.1 PolicyStoreId parameter (mandatory)

The value of the PolicyStoreId parameter shall be a valid URN (cp. 6.5) and specifies the policy store from which a policy element shall be deleted.

20.2.1.2 PolicyContainerId parameter (mandatory)

The value of the PolicyContainerId parameter shall be a valid URN (cp. 6.5) and defines the name of the policy container from which a policy element shall be deleted.

20.2.1.3 Query parameter (optional)

The value of the Query parameter shall be a valid XPath 2.0 expression (cp. 7.3) and defines one or multiple nodes that will be deleted.

If a policy element node will be deleted that has ancestors, these shall also be deleted.

An empty or missing <paws:Query> element implies that all policy root elements in the corresponding container will be deleted. The same semantic can of course be achieved by an XPath expression within the query element that points to all policy root nodes with the Container (e.g. /xacml:PolicySet).

Delete operations never propagate throughout policy references.

20.2.2 Example

The XML document below shows a DeletePolicyElement request on a policy container and store respectively. The effect of this request is to delete the policy root element of type xacml:PolicySet with Id “urn:lam:root-ps:layer:1:12345” within the policy container named "urn:mydomain:MyFirstContainer" within the policy store "urn:mydomain:MyFirstPolicyStore".

To ensure that no unwanted side effects of parallel transactions can occur, the policy tree should have been locked upfront.

```
<paws:DeletePolicyElement service="PAWS" version="1.0.0" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
  <paws:Query namespace="xmlns:xacml=urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
    /xacml:PolicySet[xacml:PolicySetId="urn:mydomain:12345"]
  </paws:Query>
</paws:DeletePolicyElement>
```

20.3 Response

20.3.1 XML encoding and semantics

The root element of the response to a DeletePolicyElement request is the paws:DeletePolicyElementResponse element which is declared by the following XML Schema fragment:

```
<xsd:element name="DeletePolicyElementResponse" type="paws:DeletePolicyElementResponseType"/>
<xsd:complexType name="DeletePolicyElementResponseType">
  <xsd:complexContent>
    <xsd:extension base="paws:BaseResponseType">
      <xsd:sequence minOccurs="1" maxOccurs="1">
        <xsd:element name="PolicyStoreId" type="paws:urnType"/>
        <xsd:element name="PolicyContainerId" type="paws:urnType"/>
        <xsd:element ref="paws:Query" minOccurs="0"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The base type, paws:BaseResponseType is defined in section 8.2.

In case of a sucessfull DeletePolicyElement operation a DeletePolicyElementResponse document shall be generated. This document contains a copy of the PolicyStoreId, PolicyContainerId and Query (if present) element of the original request.

20.3.2 Example

The XML document below shows a valid response to a DeletePolicyElement request. The response represents an acknowledgement that the policy root element with Id equal to "urn:lam:root-ps:layer:1:12345" was successfully deleted from the policy container "urn:mydomain:MyFirstContainer" within the policy store "urn:mydomain:MyFirstPolicyStore".

```
<paws:DeletePolicyElementResponse timeStamp="2013-04-01T21:00:00Z" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
  <paws:Query namespace="xmlns:xacml=urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
    /xacml:PolicySet[xacml:PolicySetId="urn:mydomain:12345"]
  </paws:Query>
</paws:DeletePolicyElementResponse>
```

20.4 Exceptions

In the event that a PAWS instance encounters an error parsing an DeletePolicyElement request, it shall raise an OperationParsingFailed exception as described in section 6.6.

In the event that a PAWS instance encounters an error processing an DeletePolicyElement request, it shall raise an OperationProcessingFailed exception as described in section 6.6.

In the event that a PAWS instance is requested to delete a policy element within an unknown policy store, it shall raise a **PolicyStoreUnknown** exception as described in section 6.6.

In the event that a PAWS instance is requested to delete a policy element within an unknown policy container, it shall raise a **PolicyContainerUnknown** exception as described in section 6.6.

In the event that a PAWS instance is requested to delete a policy element at an undefined location within an existing policy tree, it shall raise a **PolicyElementDeleteReferenceNotDefined** exception as described in section 6.6.

In the event that the paws:Query element of the DeletePolicyElement request is not empty and does not point to one or multiple element nodes of type xacml:PolicySet, xacml:Policy, xacml:Rule, xacml:PolicySetIdReference or xacml:PolicyIdReference, the PAWS instance shall raise a **QueryInvalid** exception as described in section 6.6.

In the event that a PAWS instance is requested to delete a policy element in a policy store, policy container or policy tree locked by some other user, it shall raise an **EntityLocked** exception as described in section 6.6.

21 SelectPolicyElement operation

21.1 Introduction

Through SelectPolicyElement requests, policy administrators can select specific elements of XACML policy trees for read access.

Note that in multi administrator scenarios and with respect to following transactional operations it is recommended to set a lock before reading (followed by editing) an existing policy tree. Because locks are assigned to a user context, a lockId does not need to be specified when performing any transactional operation on policy trees. To benefit from the set locks there is the prerequisite to be logged in, using a suitable user account (see section 6.7 and 15). It is however important to highlight that in case of pure read operations (i.e. reads without the intension to modify the policy afterwards) there is no need to set locks upfront.

21.2 Request

21.2.1 XML encoding and semantics

The following XML Schema fragment defines the mandatory XML encoding of SelectPolicyElement requests:

```
<xsd:element name="SelectPolicyElement" type="paws:SelectPolicyElementType"/>
<xsd:complexType name="SelectPolicyElementType">
  <xsd:complexContent>
    <xsd:extension base="paws:BaseRequestType">
      <xsd:sequence minOccurs="1" maxOccurs="1">
        <xsd:element name="PolicyStoreId" type="paws:urnType"/>
        <xsd:element name="PolicyContainerId" type="paws:urnType"/>
        <xsd:element ref="paws:Query" minOccurs="0"/>
        <xsd:element ref="paws:Dereference"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="Dereference">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="paws:DereferenceType">
        <xsd:attribute name="timeOut" type="xsd:positiveInteger" default="200" use="optional"/>
        <xsd:attribute name="depth" type="xsd:positiveInteger" use="optional"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>

<xsd:simpleType name="DereferenceType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="local"/>
    <xsd:enumeration value="remote"/>
    <xsd:enumeration value="localAndRemote"/>
  </xsd:restriction>
</xsd:simpleType>
```

Through a single SelectPolicyElement request, one can select one or multiple XACML policy elements from a specific policy container. The selected nodes in one SelectPolicyElement request shall all be of one and only one of the node types listed below:

- xacml:PolicySet

- xacml:Policy
- xacml:PolicySetIdReference
- xacml:PolicyIdReference
- xacml:Rule

Other types of XACML elements can only be selected as ancestors of the elements of the types listed above.

The base type paws: BaseRequestType is defined in section 7.2.

Note that the SelectPolicyElement operation is affected by locks as all transactional operations are. This ensures that dirty or phantom read problems are mitigated through setting locks.

21.2.1.1 PolicyStoreId parameter (mandatory)

The value of the PolicyStoreId parameter shall be a valid URN (cp. 6.5) and specifies the policy store from which the policy elements shall be selected.

21.2.1.2 PolicyContainerId parameter (mandatory)

The value of the PolicyContainerId parameter shall be a valid URN (cp. 6.5) and defines the name of the policy container from which the policy element shall be selected.

21.2.1.3 Query parameter (optional)

The value of the Query parameter shall be a valid XPath 2.0 expression (cp. 7.3) and defines one or multiple nodes or subtrees that will be selected from the specified policy container. An empty or missing <paws:Query> element implies that all policy trees in the corresponding container will be selected.

21.2.1.4 Dereference parameter (mandatory)

The mandatory Dereference parameter specifies whether <PolicySetIdReference> and <PolicyIdReference> elements shall be dereferenced – if possible - during the read access. The dereferencing shall always succeed or fail completely.

A Dereference (text node) value equal to “local” expresses that all local references shall be dereferenced. Local means that the references point to policy elements that reside in policy stores provided by the PAWS instance.

A Dereference (text node) value equal to “remote” expresses that all remote references shall be dereferenced. Remote means that the policy references point to policy elements that reside in policy stores not provided by the PAWS instance.

A Dereference (text node) value equal to “localAndRemote” expresses that all local and all remote references shall be dereferenced.

Note: How policy references are resolved in a response document can only be controlled on a per-operation basis, not a per-policy basis.

Note: How policy references are defined and resolved is left open to the PAWS and attached PDP implementations. It is highly recommended to define an XACML Reference profile for a concrete use case to achieve maximum interoperability with respect the PolicySetIdReference and PolicyIdReference element values.

The optional “depth” XML attribute of the Dereference element specifies to which depth references in referenced policy elements shall be dereferenced. If the depth attribute is missing, the PAWS instance shall derenference all references to the lowest level.

The optional “timeout” XML attribute of the Dereference element specifies a timeout in seconds. The PAWS instance shall follow the dereference process until the timeout value is reached. In this case the PAWS instance shall generate an DereferenceTimeOut exception (cp. section 21.4).

21.2.2 Example

The XML document below shows a SelectPolicyElement request on a policy container and store respectively. The effect of this request is to select the policy root element of type xacml:PolicySet with Id “urn:lam:root-ps:layer:1:12345” within the policy container named "urn:mydomain:MyFirstContainer" within the policy store "urn:mydomain:MyFirstPolicyStore".

Note that the missing “depth” attribute enforces that if there would be local (cp. the Dereference text node value) references to other policy elements below the selected xacml:PolicySet element, these and the recursively occurring references will be dereferenced until all references are resolved. However the dereferenciation process will stop after 300 seconds and the PAWS will return an error if the dereferenciation was not completed by then.

```
<paws:SelectPolicyElement service="PAWS" version="1.0.0"...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
  <paws:Query namespace="xmlns:xacml=urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
    /xacml:PolicySet[xacml:PolicySetId="urn:mydomain:12345"]
  </paws:Query>
  <paws:Dereference timeOut="300">local</paws:Dereference>
</paws:SelectPolicyElement>
```

21.3 Response

21.3.1 XML encoding and semantics

The root element of the response to a SelectPolicyElement request is the paws: PolicyElementCollection element which is declared by the following XML Schema fragment:

```
<xsd:element name="PolicyElementCollection" type="paws:PolicyElementCollectionType"/>
<xsd:complexType name="PolicyElementCollectionType">
  <xsd:complexContent>
    <xsd:extension base="paws:BaseResponseType">
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
        <xsd:element name="XacmlPolicyElement" type="paws:XacmlPolicyElementType"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

```
  <xsd:complexContent>
    <xsd:extension base="paws:BasePAWS-XACML-PolicyElementDataType">
      <xsd:attribute name="xpath" type="xsd:string" use="required"/>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>
```

The base type, paws:BaseResponseType is defined in section 8.2.

In case of a successful SelectPolicyElement operation a PolicyElementCollection document shall be generated. This document contains the set of selected XACML policy elements.

Each XacmlPolicyElement element node shall include a paws:xpath XML attribute that represents a unique xpath expression identifiying the “top-level” element node of the selected node(set).

21.3.2 Example

The XML document below shows a valid response to a SelectPolicyElement request. The response contains the selected policy root element in form of an xacml:PolicySet element with Id equal "urn:lam:root-ps:layer:1:12345".

```
<paws:PolicyElementCollection timeStamp="2013-04-01T21:00:00Z" ...>
  <paws:XacmlPolicyElement xmlns:xacml="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
    xpath="/xacml:PolicySet[1]">
    <xacml:PolicySet PolicySetId="urn:lam:root-ps:layer:1:12345" PolicyCombiningAlgId="only-one-
       applicable" Version="0.2" ...>
      <xacml:Target/>
    </xacml:PolicySet>
  </paws:XacmlPolicyElement>
</paws:PolicyElementCollection>
```

21.4 Exceptions

In the event that a PAWS instance encounters an error parsing a SelectPolicyElement request, it shall raise an OperationParsingFailed exception as described in section 6.6.

In the event that a PAWS instance encounters an error processing a SelectPolicyElement request, it shall raise an OperationProcessingFailed exception as described in section 6.6.

In the event that a PAWS instance is requested to select a policy element within an unknown policy store, it shall raise a **PolicyStoreUnknown** exception as described in section 6.6.

In the event that a PAWS instance is requested to select a policy element within an unknown policy container, it shall raise a **PolicyContainerUnknown** exception as described in section 6.6.

In the event that a PAWS instance is requested to select a policy element at an undefined location within an existing policy tree, it shall raise a **PolicyElementDeleteReferenceNotDefined** exception as described in section 6.6.

In the event that the paws:Query element of the SelectPolicyElement request is not empty and does not point to one or multiple element nodes of type xacml:PolicySet, xacml:Policy,

xacml:Rule, xacml:PolicySetIdReference or xacml:PolicyIdReference, the PAWS instance shall raise a **QueryInvalid** exception as described in section 6.6.

In the event that a PAWS instance is requested to select a policy element in a policy store, policy container or policy tree locked by some other user, it shall raise an **EntityLocked** exception as described in section 6.6.

In the event that a PAWS instance fails to dereference all of the to be dereferenced policy references (cp. the Dereference parameter value), it shall raise an **DereferenceFailed** exception as described in section 6.6.

Annex A
(normative)

Conformance testing

A.1 Requirements listing

This section defines requirements of the PAWS specification.

REQ 1. All requirements defined in chapter 6 shall be met.
REQ 2. All requirements defined in chapter 7 shall be met.
REQ 3. All requirements defined in chapter 8 shall be met.
REQ 4. All requirements defined in chapter 9 shall be met.
REQ 5. All requirements defined in chapter 10 shall be met.
REQ 6. All requirements defined in chapter 11 shall be met.
REQ 7. All requirements defined in chapter 12 shall be met.
REQ 8. All requirements defined in chapter 13 shall be met.
REQ 9. All requirements defined in chapter 14 shall be met.
REQ 10. All requirements defined in chapter 15 shall be met.
REQ 11. All requirements defined in chapter 16 shall be met.
REQ 12. All requirements defined in chapter 17 shall be met.
REQ 13. All requirements defined in chapter 18 shall be met.
REQ 14. All requirements defined in chapter 19 shall be met.
REQ 15. All requirements defined in chapter 20 shall be met.
REQ 16. All requirements defined in chapter 21 shall be met.

A.2 Conformance classes and tests

Conformance class	Conformance test
Basic PAWS	Test that the implementation is compliant with REQ 1 to 16
Analyze PAWS	todo
Optimize PAWS	todo
Transform PAWS	todo
Test PAWS	todo
Trust PAWS	todo

Annex B
(informative)

Consolidated XML Schema

B.1 Introduction

This Annex consolidates the XML fragments found in this specification into a single file called paws.xsd that may be used with an XML parser to validate paws requests.

B.2 paws.xsd

```
<?xml version="1.1" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.opengis.net/paws/1.0"
xmlns:paws="http://www.opengis.net/paws/1.0" xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:xacml="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xml="http://www.w3.org/XML/1998/namespace" elementFormDefault="qualified"
version="2.0.1">
   <xsd:annotation>
      <xsd:documentation> copyright disclaimer must be added in the end </xsd:documentation>
   </xsd:annotation>

   <!-- ============================================================================
-->
   <!-- =   Includes and Imports                                   = -->
   <!-- ============================================================================
-->
   <xsd:import namespace="http://www.w3.org/XML/1998/namespace"
schemaLocation="http://www.w3.org/2001/xml.xsd"/>
   <xsd:import namespace="http://www.w3.org/1999/xlink"
schemaLocation="http://www.w3.org/1999/xlink.xsd"/>
   <xsd:import namespace="http://www.opengis.net/ows/1.1"
schemaLocation="http://schemas.opengis.net/ows/1.1.0/owsAll.xsd"/>
   <xsd:import namespace="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
schemaLocation="http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"/>

   <!-- ============================================================================
-->
   <!-- =   BASE REQUEST TYPE                                     = -->
   <!-- ============================================================================
-->
   <xsd:complexType name="BaseRequestType" abstract="true">
      <xsd:attribute name="service" type="xsd:string" use="required" fixed="PAWS"/>
      <xsd:attribute name="version" type="xsd:string" use="required" fixed="1.0.0"/>
   </xsd:complexType>
```

```
   <!-- ================================================================================
-->
   <!-- =   BASE Response TYPE                                    = -->
   <!-- ================================================================================
-->
   <xsd:complexType name="BaseResponseType" abstract="true">
      <xsd:attribute name="timeStamp" type="xsd:dateTime" use="required"/>
   </xsd:complexType>

   <!-- ================================================================================
-->
   <!-- =   Global Data Types                                    = -->
   <!-- ================================================================================
-->
   <xsd:simpleType name="urnType">
      <xsd:annotation>
         <xsd:documentation> Regular expression to validate RFC 2141 URN syntax.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:anyURI">
         <xsd:pattern value="urn:[a-zA-Z0-9][a-zA-Z0-9-]{1,31}:([a-zA-Z0-9()+,.:=@;$_!*'-]|%[0-
9A-Fa-f]{2})+"/>
      </xsd:restriction>
   </xsd:simpleType>

   <xsd:simpleType name="SupportedConformanceClassType">
      <xsd:restriction base="paws:urnType">
         <xsd:enumeration value="urn:ogc:conf-class:paws:1.0.0:basic"/>
         <xsd:enumeration value="urn:ogc:conf-class:paws:1.0.0:test-paws"/>
         <xsd:enumeration value="urn:ogc:conf-class:paws:1.0.0:analyze-paws"/>
         <xsd:enumeration value="urn:ogc:conf-class:paws:1.0.0:transform-paws"/>
         <xsd:enumeration value="urn:ogc:conf-class:paws:1.0.0:optimize-paws"/>
      </xsd:restriction>
   </xsd:simpleType>

   <xsd:complexType name="BasePAWS-XACML-PolicyElementDataType">
      <xsd:choice>
         <xsd:element ref="xacml:PolicySet"/>
         <xsd:element ref="xacml:Policy"/>
         <xsd:element ref="xacml:Rule"/>
         <xsd:element ref="xacml:PolicySetIdReference"/>
         <xsd:element ref="xacml:PolicyIdReference"/>
      </xsd:choice>
   </xsd:complexType>

   <xsd:simpleType name="InsertStyleType">
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="as-new-last-child"/>
```

```
            <xsd:enumeration value="as-sibling-before"/>
            <xsd:enumeration value="as-sibling-after"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="UpdateStyleType">
        <xsd:restriction base="xsd:string">
            <xsd:enumeration value="complete"/>
            <xsd:enumeration value="keep-descendants"/>
        </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="SubjectIdType">
        <xsd:union>
            <xsd:simpleType>
                <xsd:restriction base="xsd:string"/>
            </xsd:simpleType>
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="*"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:union>
    </xsd:simpleType>

    <xsd:complexType name="LockTypeExtended">
        <xsd:complexContent>
            <xsd:extension base="paws:BaseRequestType">
                <xsd:sequence minOccurs="1" maxOccurs="1">
                    <xsd:element name="PolicyStoreId" type="paws:urnType" minOccurs="0"/>
                    <xsd:element name="PolicyContainerId" type="paws:urnType" minOccurs="0"/>
                    <xsd:element ref="paws:Query" minOccurs="0"/>
                    <xsd:sequence>
                        <xsd:element name="SubjectId" type="paws:SubjectIdType" minOccurs="0"/>
                    </xsd:sequence>
                </xsd:sequence>
                <xsd:attribute name="expiry" type="xsd:positiveInteger" default="200" use="optional"/>
                <xsd:attribute name="expiryDate" type="xsd:dateTime" use="optional"/>
                <xsd:attribute name="lockId" type="xsd:string" use="required"/>
                <xsd:attribute name="lockCreationDateTime" type="xsd:dateTime" use="required"/>
                <xsd:attribute name="lockOwner" type="xsd:string" use="required"/>
            </xsd:extension>
        </xsd:complexContent>
    </xsd:complexType>


    <!-- ========================================================================
-->
    <!-- =   QUERY ELEMENT                                    = -->
```

```
-->
   <xsd:element name="Query">
      <xsd:complexType>
         <xsd:simpleContent>
            <xsd:extension base="xsd:string">
               <xsd:attribute name="namespace" type="xsd:anyURI" use="optional"/>
            </xsd:extension>
         </xsd:simpleContent>
      </xsd:complexType>
   </xsd:element>

   <!-- ========================================================================
-->
   <!-- =   Dereference ELEMENT                                  = -->
   <!-- ========================================================================
-->
   <xsd:element name="Dereference">
      <xsd:complexType>
         <xsd:simpleContent>
            <xsd:extension base="paws:DereferenceType">
               <xsd:attribute name="timeOut" type="xsd:positiveInteger" default="200"
use="optional"/>
               <xsd:attribute name="depth" type="xsd:positiveInteger" use="optional"/>
            </xsd:extension>
         </xsd:simpleContent>
      </xsd:complexType>
   </xsd:element>

   <xsd:simpleType name="DereferenceType">
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="local"/>
         <xsd:enumeration value="remote"/>
         <xsd:enumeration value="localAndRemote"/>
      </xsd:restriction>
   </xsd:simpleType>


   <!-- ========================================================================
-->
   <!-- =   GETCAPABILITIES Request and Response                 = -->
   <!-- ========================================================================
-->
   <!-- REQUEST -->
   <xsd:element name="GetCapabilities" type="paws:GetCapabilitiesType"/>
   <xsd:complexType name="GetCapabilitiesType">
      <xsd:complexContent>
         <xsd:extension base="ows:GetCapabilitiesType">
            <xsd:attribute name="service" type="ows:ServiceType" use="required"
```

```
fixed="PAWS"/>
        </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>

   <!-- RESPONSE -->
   <xsd:element name="GetCapabilitiesResponse"
type="paws:PAWS_GetCapabilitiesResponseType"/>
   <xsd:complexType name="PAWS_GetCapabilitiesResponseType">
      <xsd:complexContent>
         <xsd:extension base="ows:CapabilitiesBaseType">
            <xsd:sequence>
               <xsd:element name="WSDL">
                  <xsd:complexType>
                     <xsd:complexContent>
                        <xsd:restriction base="xsd:anyType">
                           <xsd:attributeGroup ref="xlink:simpleAttrs"/>
                        </xsd:restriction>
                     </xsd:complexContent>
                  </xsd:complexType>
               </xsd:element>
               <xsd:element ref="paws:PolicyStoreList"/>
               <xsd:element ref="paws:SupportedConformanceClassesList"/>
            </xsd:sequence>
            <xsd:attribute name="timeStamp" type="xsd:dateTime" use="required"/>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>

   <xsd:element name="PolicyStoreList" type="paws:PolicyStoreListType"/>
   <xsd:complexType name="PolicyStoreListType">
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
         <xsd:element name="PolicyStore" type="paws:PolicyStoreType"
maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="PolicyStoreType">
      <xsd:sequence>
         <xsd:element name="Name" type="paws:urnType"/>
         <xsd:element name="Title" type="xsd:string" minOccurs="0"/>
         <xsd:element name="Description" type="xsd:string"/>
      </xsd:sequence>
   </xsd:complexType>

   <xsd:element name="SupportedConformanceClassesList"
type="paws:SupportedConformanceClassesListType"/>
   <xsd:complexType name="SupportedConformanceClassesListType">
      <xsd:sequence minOccurs="0" maxOccurs="unbounded">
         <xsd:element name="SupportedConformanceClass"
```

```
      </xsd:sequence>
    </xsd:complexType>


    <!-- ==========================================================================
-->
    <!-- =   CreatePolicyContainer Request and Response                  = -->
    <!-- ==========================================================================
-->
    <!-- REQUEST -->
    <xsd:element name="CreatePolicyContainer" type="paws:CreatePolicyContainerType"/>
    <xsd:complexType name="CreatePolicyContainerType">
      <xsd:complexContent>
        <xsd:extension base="paws:BaseRequestType">
          <xsd:sequence minOccurs="1" maxOccurs="1">
            <xsd:element name="PolicyStoreId" type="paws:urnType"/>
            <xsd:element name="PolicyContainerId" type="paws:urnType"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <!-- RESPONSE -->
    <xsd:element name="CreatePolicyContainerResponse"
type="paws:CreatePolicyContainerResponseType"/>
    <xsd:complexType name="CreatePolicyContainerResponseType">
      <xsd:complexContent>
        <xsd:extension base="paws:BaseResponseType">
          <xsd:sequence minOccurs="1" maxOccurs="1">
            <xsd:element name="PolicyStoreId" type="paws:urnType"/>
            <xsd:element name="PolicyContainerId" type="paws:urnType"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>


    <!-- ==========================================================================
-->
    <!-- =   ListPolicyContainers Request and Response                  = -->
    <!-- ==========================================================================
-->
    <!-- REQUEST -->
    <xsd:element name="ListPolicyContainers" type="paws:ListPolicyContainersType"/>
    <xsd:complexType name="ListPolicyContainersType">
      <xsd:complexContent>
        <xsd:extension base="paws:BaseRequestType">
```

```
            <xsd:sequence minOccurs="1" maxOccurs="1">
               <xsd:element name="PolicyStoreId" type="paws:urnType"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>

<!-- note :  ListPolicyContainer on Locked Store will fail. ListPolicyContainer if one Conatiner is
locked will list all unlocked an locked containers-->

   <!-- RESPONSE -->
   <xsd:element name="ListPolicyContainersResponse"
type="paws:ListPolicyContainersResponseType"/>
   <xsd:complexType name="ListPolicyContainersResponseType">
      <xsd:complexContent>
         <xsd:extension base="paws:BaseResponseType">
            <xsd:sequence minOccurs="0" maxOccurs="unbounded">
               <xsd:element name="PolicyContainerId" type="paws:urnType"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>


   <!-- ========================================================================
-->
   <!-- =   CopyPolicyContainer Request and Response                    = -->
   <!-- ========================================================================
-->
   <!-- REQUEST -->
   <xsd:element name="CopyPolicyContainer" type="paws:CopyPolicyContainerType"/>
   <xsd:complexType name="CopyPolicyContainerType">
      <xsd:complexContent>
         <xsd:extension base="paws:BaseRequestType">
            <xsd:sequence minOccurs="1" maxOccurs="1">
               <xsd:element name="SourcePolicyStoreId" type="paws:urnType"/>
               <xsd:element name="PolicyContainerId" type="paws:urnType"/>
               <xsd:element name="DestinationPolicyStoreId" type="paws:urnType"/>
               <xsd:element name="DestinationPolicyContainerId" type="paws:urnType"
minOccurs="0"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>

   <!-- RESPONSE -->
   <xsd:element name="CopyPolicyContainerResponse"
type="paws:CopyPolicyContainerResponseType"/>
   <xsd:complexType name="CopyPolicyContainerResponseType">
```

```
        <xsd:extension base="paws:BaseResponseType">
          <xsd:sequence minOccurs="1" maxOccurs="1">
            <xsd:element name="SourcePolicyStoreId" type="paws:urnType"/>
            <xsd:element name="PolicyContainerId" type="paws:urnType"/>
            <xsd:element name="DestinationPolicyStoreId" type="paws:urnType"/>
            <xsd:element name="DestinationPolicyContainerId" type="paws:urnType"
minOccurs="0"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <!-- ========================================================================
-->
    <!-- =   MovePolicyContainer Request and Response                  = -->
    <!-- ========================================================================
-->
    <!-- REQUEST -->
    <xsd:element name="MovePolicyContainer" type="paws:MovePolicyContainerType"/>
    <xsd:complexType name="MovePolicyContainerType">
      <xsd:complexContent>
        <xsd:extension base="paws:BaseRequestType">
          <xsd:sequence minOccurs="1" maxOccurs="1">
            <xsd:element name="SourcePolicyStoreId" type="paws:urnType"/>
            <xsd:element name="SourcePolicyContainerId" type="paws:urnType"/>
            <xsd:element name="DestinationPolicyStoreId" type="paws:urnType"/>
            <xsd:element name="DestinationPolicyContainerId" type="paws:urnType"
minOccurs="0"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <!-- RESPONSE -->
    <xsd:element name="MovePolicyContainerResponse"
type="paws:MovePolicyContainerResponseType"/>
    <xsd:complexType name="MovePolicyContainerResponseType">
      <xsd:complexContent>
        <xsd:extension base="paws:BaseResponseType">
          <xsd:sequence minOccurs="1" maxOccurs="1">
            <xsd:element name="SourcePolicyStoreId" type="paws:urnType"/>
            <xsd:element name="PolicyContainerId" type="paws:urnType"/>
            <xsd:element name="DestinationPolicyStoreId" type="paws:urnType"/>
            <xsd:element name="DestinationPolicyContainerId" type="paws:urnType"
minOccurs="0"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
```

```
    </xsd:complexType>


    <!-- ==========================================================================
-->
    <!-- =   DeletePolicyContainer Request and Response                  = -->
    <!-- ==========================================================================
-->
    <!-- REQUEST -->
    <xsd:element name="DeletePolicyContainer" type="paws:DeletePolicyContainerType"/>
    <xsd:complexType name="DeletePolicyContainerType">
      <xsd:complexContent>
        <xsd:extension base="paws:BaseRequestType">
          <xsd:sequence minOccurs="1" maxOccurs="1">
            <xsd:element name="PolicyStoreId" type="paws:urnType"/>
            <xsd:element name="PolicyContainerId" type="paws:urnType"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <!-- RESPONSE -->
    <xsd:element name="DeletePolicyContainerResponse"
type="paws:DeletePolicyContainerResponseType"/>
    <xsd:complexType name="DeletePolicyContainerResponseType">
      <xsd:complexContent>
        <xsd:extension base="paws:BaseResponseType">
          <xsd:sequence minOccurs="1" maxOccurs="1">
            <xsd:element name="PolicyStoreId" type="paws:urnType"/>
            <xsd:element name="PolicyContainerId" type="paws:urnType"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <!-- ==========================================================================
-->
    <!-- =   CreateLock Request and Response                            = -->
    <!-- ==========================================================================
-->
    <!-- REQUEST -->
    <xsd:element name="CreateLock" type="paws:CreateLockType"/>
    <xsd:complexType name="CreateLockType">
      <xsd:complexContent>
        <xsd:extension base="paws:BaseRequestType">
          <xsd:sequence minOccurs="1" maxOccurs="1">
            <xsd:element name="PolicyStoreId" type="paws:urnType" minOccurs="0"/>
            <xsd:element name="PolicyContainerId" type="paws:urnType" minOccurs="0"/>
            <xsd:element ref="paws:Query" minOccurs="0"/>
```

```
            <!-- the next two attributes shall be used in an xor fasion -->
            <xsd:attribute name="expiry" type="xsd:positiveInteger" use="optional" />
            <xsd:attribute name="expiryDate" type="xsd:dateTime" use="optional" />
            <xsd:attribute name="lockId" type="xsd:string" use="optional"/>
          </xsd:extension>
        </xsd:complexContent>
      </xsd:complexType>

      <!-- RESPONSE -->
      <xsd:element name="CreateLockResponse" type="paws:CreateLockResponseType"/>
      <xsd:complexType name="CreateLockResponseType">
        <xsd:complexContent>
          <xsd:extension base="paws:BaseResponseType">
            <xsd:sequence minOccurs="1" maxOccurs="1">
              <xsd:element name="PolicyStoreId" type="paws:urnType" minOccurs="0"/>
              <xsd:element name="PolicyContainerId" type="paws:urnType" minOccurs="0"/>
              <xsd:element ref="paws:Query" minOccurs="0"/>
            </xsd:sequence>
            <xsd:attribute name="expiry" type="xsd:positiveInteger" use="optional" />
            <xsd:attribute name="expiryDate" type="xsd:dateTime" use="optional" />
            <xsd:attribute name="lockId" type="xsd:string" use="required"/>
            <xsd:attribute name="lockCreationDateTime" type="xsd:dateTime" use="required" />
          </xsd:extension>
        </xsd:complexContent>
      </xsd:complexType>

      <!-- ========================================================================
    -->
      <!-- =   ShowLocks Request and Response                  = -->
      <!-- ========================================================================
    -->
      <!-- REQUEST -->
      <xsd:element name="ShowLocks" type="paws:ShowLocksType"/>
      <xsd:complexType name="ShowLocksType">
        <xsd:complexContent>
          <xsd:extension base="paws:BaseRequestType">
            <xsd:sequence minOccurs="1" maxOccurs="1">
              <xsd:sequence>
                <xsd:element name="SubjectId" type="paws:SubjectIdType" minOccurs="0"/>
              </xsd:sequence>
            </xsd:sequence>
          </xsd:extension>
        </xsd:complexContent>
      </xsd:complexType>

      <!-- RESPONSE -->
      <xsd:element name="ShowLocksResponse" type="paws:ShowLocksResponseType"/>
      <xsd:complexType name="ShowLocksResponseType">
```

```
    <xsd:complexContent>
      <xsd:extension base="paws:BaseResponseType">
        <xsd:sequence minOccurs="0" maxOccurs="unbounded">
          <xsd:element name="Lock" type="paws:LockTypeExtended"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- ====================================================================
-->
  <!-- =   ReleaseLock Request and Response                    = -->
  <!-- ====================================================================
-->
  <!-- REQUEST -->
  <xsd:element name="ReleaseLock" type="paws:ReleaseLockType"/>
  <xsd:complexType name="ReleaseLockType">
    <xsd:complexContent>
      <xsd:extension base="paws:BaseRequestType">
        <xsd:sequence minOccurs="0" maxOccurs="1">
          <xsd:element name="SubjectId" type="xsd:string" minOccurs="0"/>
        </xsd:sequence>
        <xsd:attribute name="lockId" type="xsd:string" use="required"/>
        <xsd:attribute name="rollback" type="xsd:boolean" use="optional" default="false"/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <!-- RESPONSE -->
  <xsd:element name="ReleaseLockResponse" type="paws:ReleaseLockResponseType"/>
  <xsd:complexType name="ReleaseLockResponseType">
    <xsd:complexContent>
      <xsd:extension base="paws:BaseResponseType">
        <xsd:sequence minOccurs="1" maxOccurs="1">
          <xsd:element name="Lock" type="paws:LockTypeExtended"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

    <!-- ====================================================================
-->
    <!-- =   InsertPolicyElement Request and Response                = -->
    <!-- ====================================================================
-->
    <!-- REQUEST -->
    <xsd:element name="InsertPolicyElement" type="paws:InsertPolicyElementType"/>
    <xsd:complexType name="InsertPolicyElementType">
      <xsd:complexContent>
```

```
            <xsd:sequence minOccurs="1" maxOccurs="1">
               <xsd:element name="PolicyStoreId" type="paws:urnType"/>
               <xsd:element name="PolicyContainerId" type="paws:urnType"/>
               <xsd:element ref="paws:Query" minOccurs="0"/>
               <xsd:element name="InsertStyle" type="paws:InsertStyleType" minOccurs="0"/>
               <xsd:element name="XacmlPolicyElement" type="paws:BasePAWS-XACML-
PolicyElementDataType"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>

   <!-- RESPONSE -->
   <xsd:element name="InsertPolicyElementResponse"
type="paws:InsertPolicyElementResponseType"/>
   <xsd:complexType name="InsertPolicyElementResponseType">
      <xsd:complexContent>
         <xsd:extension base="paws:BaseResponseType">
            <xsd:sequence minOccurs="1" maxOccurs="1">
               <xsd:element name="PolicyStoreId" type="paws:urnType"/>
               <xsd:element name="PolicyContainerId" type="paws:urnType"/>
               <xsd:element ref="paws:Query" minOccurs="0"/>
               <xsd:element name="InsertStyle" type="paws:InsertStyleType"/>
               <xsd:element name="XacmlPolicyElement" type="paws:BasePAWS-XACML-
PolicyElementDataType"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>

   <!-- ========================================================================
-->
   <!-- =   UpdatePolicyElement Request and Response                = -->
   <!-- ========================================================================
-->
   <!-- REQUEST -->
   <xsd:element name="UpdatePolicyElement" type="paws:UpdatePolicyElementType"/>
   <xsd:complexType name="UpdatePolicyElementType">
      <xsd:complexContent>
         <xsd:extension base="paws:BaseRequestType">
            <xsd:sequence minOccurs="1" maxOccurs="1">
               <xsd:element name="PolicyStoreId" type="paws:urnType"/>
               <xsd:element name="PolicyContainerId" type="paws:urnType"/>
               <xsd:element ref="paws:Query"/>
               <xsd:element name="XacmlPolicyElement" type="paws:BasePAWS-XACML-
PolicyElementDataType"/>
               <xsd:element name="UpdateStyle" type="paws:UpdateStyleType"/>
            </xsd:sequence>
```

```
        </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>

   <!-- RESPONSE -->
   <xsd:element name="UpdatePolicyElementResponse"
type="paws:UpdatePolicyElementResponseType"/>
   <xsd:complexType name="UpdatePolicyElementResponseType">
      <xsd:complexContent>
         <xsd:extension base="paws:BaseResponseType">
            <xsd:sequence minOccurs="1" maxOccurs="1">
               <xsd:element name="PolicyStoreId" type="paws:urnType"/>
               <xsd:element name="PolicyContainerId" type="paws:urnType"/>
               <xsd:element ref="paws:Query"/>
               <xsd:element name="XacmlPolicyElement" type="paws:BasePAWS-XACML-
PolicyElementDataType"/>
               <xsd:element name="UpdateStyle" type="paws:UpdateStyleType"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>


   <!-- ====================================================================
-->
   <!-- =   DeletePolicyElement Request and Response                    = -->
   <!-- ====================================================================
-->
   <!-- REQUEST -->
   <xsd:element name="DeletePolicyElement" type="paws:DeletePolicyElementType"/>
   <xsd:complexType name="DeletePolicyElementType">
      <xsd:complexContent>
         <xsd:extension base="paws:BaseRequestType">
            <xsd:sequence minOccurs="1" maxOccurs="1">
               <xsd:element name="PolicyStoreId" type="paws:urnType"/>
               <xsd:element name="PolicyContainerId" type="paws:urnType"/>
               <xsd:element ref="paws:Query" minOccurs="0"/>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>

   <!-- RESPONSE -->
   <xsd:element name="DeletePolicyElementResponse"
type="paws:DeletePolicyElementResponseType"/>
   <xsd:complexType name="DeletePolicyElementResponseType">
      <xsd:complexContent>
         <xsd:extension base="paws:BaseResponseType">
            <xsd:sequence minOccurs="1" maxOccurs="1">
```

```
            <xsd:element name="PolicyContainerId" type="paws:urnType"/>
            <xsd:element ref="paws:Query" minOccurs="0"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <!-- ========================================================================
-->
    <!-- =   SelectPolicyElement Request and Response                    = -->
    <!-- ========================================================================
-->
    <!-- REQUEST -->
    <xsd:element name="SelectPolicyElement" type="paws:SelectPolicyElementType"/>
    <xsd:complexType name="SelectPolicyElementType">
      <xsd:complexContent>
        <xsd:extension base="paws:BaseRequestType">
          <xsd:sequence minOccurs="1" maxOccurs="1">
            <xsd:element name="PolicyStoreId" type="paws:urnType"/>
            <xsd:element name="PolicyContainerId" type="paws:urnType"/>
            <xsd:element ref="paws:Query" minOccurs="0"/>
            <xsd:element ref="paws:Dereference"/>
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

    <!-- RESPONSE -->
    <xsd:element name="PolicyElementCollection" type="paws:PolicyElementCollectionType"/>
    <xsd:complexType name="PolicyElementCollectionType">
     <xsd:complexContent>
       <xsd:extension base="paws:BaseResponseType">
          <xsd:sequence minOccurs="0" maxOccurs="unbounded">
            <xsd:element name="XacmlPolicyElement" type="paws:XacmlPolicyElementType"/>
          </xsd:sequence>
         <xsd:attribute name="hits" type="xsd:integer" use="optional"/>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>

  <xsd:complexType name="XacmlPolicyElementType">
     <xsd:complexContent>
      <xsd:extension base="paws:BasePAWS-XACML-PolicyElementDataType">
       <xsd:attribute name="xpath" type="xsd:string" use="required"/>
      </xsd:extension>
     </xsd:complexContent>
    </xsd:complexType>
</xsd:schema>
```

Annex C
(informative)

Examples

C.1 Introduction

This Annex consolidates the examples found in this specification.

C.2 Exception report example

The following is an example of an exception report. In this particular case, a CreatePolicyContainer operation has failed because the specified policy store identifier was not know to the PAWS instance.

```
<?xml version="1.0" ?>
<ExceptionReport
   version="2.0.0"
   xmlns="http://www.opengis.net/ows/1.1"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opengis.net/ows/1.1
                       http://schemas.opengis.net/ows/1.1.0/owsAll.xsd">
   <Exception exceptionCode="PolicyStoreUnknown">
      <ExceptionText>PolicyContainer could not be created as the specified PolicyStore "urn:my-
default-policy-store" is not know to the PAWS instance.</ExceptionText>
   </Exception>
</ExceptionReport>
```

C.3 GetCapabilities request and response example

```
<paws:GetCapabilities service="PAWS"
  xmlns="http://www.opengis.net/paws/1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:paws="http://www.opengis.net/paws/1.0"
  xsi:schemaLocation="http://www.opengis.net/paws/1.0
  http://schemas.opengis.net/paws/1.0.0/paws.xsd"/>[8]

<paws:GetCapabilitiesResponse timeStamp="2013-04-01T21:00:00Z" version="1.0.0" ...>
  <paws:WSDL xlink:href="http://www.someserver.com/paws.wsdl"/>
  <paws:PolicyStoreList>
    <paws:PolicyStore>
      <paws:Name>urn:mydomain:MyFirstPolicyStore</paws:Name>
      <paws:Title>MyFirstPolicyStore (Testing Environment)</paws:Title>
      <paws:Description>
        This policy store contains policies protecting organisation unit A's WFS instances. It is a
       test environment. The implemenation of the policy store is based on file system folders.
      </paws:Description>
    </paws:PolicyStore>
  </paws:PolicyStoreList>
  <paws:SupportedConformanceClassesList>
    <paws:SupportedConformanceClass>
      urn:ogc:conf-class:paws:1.0.0:basic
    </paws:SupportedConformanceClass>
  </paws:SupportedConformanceClassesList>
</paws:GetCapabilitiesResponse>
```

[8] Note: Namespace definition will be obmitted in the following examples

C.4 Container related requests and responses examples

Example 1: CreatePolicyContainer request and response

```
<paws:CreatePolicyContainer service="PAWS" version="1.0.0" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
</paws:CreatePolicyContainer>

<paws:CreatePolicyContainerResponse timeStamp="2013-04-01T21:00:00Z" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
</paws:CreatePolicyContainerResponse>
```

Example 2: ListPolicyContainers request and response

```
<paws:ListPolicyContainers service="PAWS" version="1.0.0" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
</paws:ListPolicyContainers>

<paws:ListPolicyContainersResponse timeStamp="2013-04-01T21:00:00Z" ...>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
  <paws:PolicyContainerId>urn:mydomain:MySecondContainer</paws:PolicyContainerId>
</paws:ListPolicyContainersResponse>
```

Example 3: CopyPolicyContainer request and response

```
<paws:CopyPolicyContainer service="PAWS" version="1.0.0" ...>
  <paws:SourcePolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:SourcePolicyStoreId>
  <paws:SourcePolicyContainerId>urn:mydomain:MyFirstContainer</paws:SourcePolicyContainerId>
  <paws:DestinationPolicyStoreId>urn:mydomain:MyBackupPolicyStore</paws:DestinationPolicyStoreId>
</paws:CopyPolicyContainer>

<paws:CopyPolicyContainerResponse timeStamp="2013-04-01T21:00:00Z" ...>
  <paws:SourcePolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:SourcePolicyStoreId>
  <paws:SourcePolicyContainerId>urn:mydomain:MyFirstContainer</paws:SourcePolicyContainerId>
  <paws:DestinationPolicyStoreId>urn:mydomain:MyBackupPolicyStore</paws:DestinationPolicyStoreId>
</paws:CopyPolicyContainerResponse>
```

Example 4: MovePolicyContainer request and response

```
<paws:MovePolicyContainer service="PAWS" version="1.0.0" ...>
  <paws:SourcePolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:SourcePolicyStoreId>
  <paws:SourcePolicyContainerId>urn:mydomain:MyFirstContainer</paws:SourcePolicyContainerId>
  <paws:DestinationPolicyStoreId>urn:mydomain:MyBackupPolicyStore</paws:DestinationPolicyStoreId>
</paws:MovePolicyContainer>

<paws:MovePolicyContainerResponse timeStamp="2013-04-01T21:00:00Z" ...>
  <paws:SourcePolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:SourcePolicyStoreId>
  <paws:SourcePolicyContainerId>urn:mydomain:MyFirstContainer</paws:SourcePolicyContainerId>
  <paws:DestinationPolicyStoreId>urn:mydomain:MyBackupPolicyStore</paws:DestinationPolicyStoreId>
</paws:MovePolicyContainerResponse>
```

Example 6: DeletePolicyContainer request and response

```
<paws:DeletePolicyContainer service="PAWS" version="1.0.0" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
</paws:DeletePolicyContainer>

<paws:DeletePolicyContainerResponse timeStamp="2013-04-01T21:00:00Z" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
</paws:DeletePolicyContainerResponse>
```

C.5 CreateLock, ShowLocks and ReleaseLock request and response examples

Example 1: Locking of a policy store and the resulting response

```
<paws:CreateLock service="PAWS" version="1.0.0"...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
</paws:CreateLock>

<paws:CreateLockResponse timeStamp="2013-04-01T21:00:01Z" lockId="550e8400-e29b-11d4-a716-
       446655440001" lockCreationDateTime="2013-04-01T21:00:00Z" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
</paws:CreateLockResponse>
```

Example 2: Locking of a policy container and the resulting response

```
<paws:CreateLock service="PAWS" version="1.0.0" expiry="120" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
</paws:CreateLock>

<paws:CreateLockResponse timeStamp="2013-04-01T21:00:01Z" lockId="550e8400-e29b-11d4-a716-
446655440002" expiry="120" lockCreationDateTime="2013-04-01T21:00:00Z"...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
</paws:CreateLockResponse>
```

Example 3: Locking of a policy tree and the resulting response

```
<paws:CreateLock service="PAWS" version="1.0.0" expiry="120" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
  <paws:Query namespace="xmlns:xacml=urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
      /xacml:PolicySet[xacml:PolicySetId="urn:mydomain:12345"]
  </paws:Query>
</paws:CreateLock>

<paws:CreateLockResponse timeStamp="2013-04-01T21:00:01Z" lockId="550e8400-e29b-11d4-a716-
446655440003" expiry="120" lockCreationDateTime="2013-04-01T21:00:00Z"...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
  <paws:Query namespace="xmlns:xacml=urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
       /xacml:PolicySet[xacml:PolicySetId="urn:mydomain:12345"]
  </paws:Query>
</paws:CreateLockResponse>
```

Example 4: ShowLocks request and response

```
<paws:ShowLocks service="PAWS" version="1.0.0".../>

<paws:ShowLocksResponse timeStamp="2013-04-05T22:00:10" ...>
  <paws:Lock lockId="550e8400-e29b-11d4-a716-446655440001" lockCreationDateTime="2013-04-
01T21:00:00Z" lockOwner="testUser1">
    <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  </paws:Lock>
</paws:ShowLocksResponse>
```

Example 5: ReleaseLock request and response

```
<paws:ReleaseLock lockId="550e8400-e29b-11d4-a716-446655440001" service="PAWS" version="1.0.0".../>

<paws:ReleaseLockResponse timeStamp="2013-04-05T22:00:10" ...>
  <paws:Lock lockId="550e8400-e29b-11d4-a716-446655440001" lockCreationDateTime="2013-04-
01T21:00:00Z" lockOwner="testUser1">
    <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  </paws:Lock>
</paws:ReleaseLockResponse>
```

C.6 CRUD PolicyElement request and response examples

Example 1: InsertPolicyElement request and response

```
<paws:InsertPolicyElement service="PAWS" version="1.0.0" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
  <paws:InsertStyle>as-new-last-child</InsertStyle>
  <paws:XacmlPolicyElement>
    <xacml:PolicySet PolicySetId="urn:lam:root-ps:layer:1:12345" PolicyCombiningAlgId="only-one-
        applicable" Version="0.1" ...>
      <xacml:Target/>
    </xacml:PolicySet>
  </paws:XacmlPolicyElement>
</paws:InsertPolicyElement>

<paws:InsertPolicyElementResponse timeStamp="2013-04-01T21:00:00Z"...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
  <paws:InsertStyle>as-new-last-child</InsertStyle>
  <paws:XacmlPolicyElement>
    <xacml:PolicySet PolicySetId="urn:lam:root-ps:layer:1:12345" PolicyCombiningAlgId="only-one-
        applicable" Version="0.1"...>
      <xacml:Target/>
    </xacml:PolicySet>
  </paws:XacmlPolicyElement>
</paws:InsertPolicyElementResponse>
```

Example 2: UpdatePolicyElement request and response

```
<paws:UpdatePolicyElement service="PAWS" version="1.0.0" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
  <paws:Query namespace="xmlns:xacml=urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
    /xacml:PolicySet[xacml:PolicySetId="urn:mydomain:12345"]
  </paws:Query>
  <paws:XacmlPolicyElement>
    <xacml:PolicySet PolicySetId="urn:lam:root-ps:layer:1:12345" PolicyCombiningAlgId="only-one-
        applicable" Version="0.2" ...>
      <xacml:Target/>
    </xacml:PolicySet>
  </paws:XacmlPolicyElement>
  <paws:UpdateStyle>complete</paws:UpdateStyle>
</paws:UpdatePolicyElement>

<paws:UpdatePolicyElementResponse timeStamp="2013-04-01T21:00:00Z" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
  <paws:Query namespace="xmlns:xacml=urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
    /xacml:PolicySet[xacml:PolicySetId="urn:mydomain:12345"]
  </paws:Query>
  <paws:XacmlPolicyElement>
    <xacml:PolicySet PolicySetId="urn:lam:root-ps:layer:1:12345" PolicyCombiningAlgId="only-one-
        applicable" Version="0.2">
      <xacml:Target/>
    </xacml:PolicySet>
  </paws:XacmlPolicyElement>
  <paws:UpdateStyle>complete</paws:UpdateStyle>
</paws:UpdatePolicyElementResponse>
```

Example 3: DeletePolicyElement request and response

```
<paws:DeletePolicyElement service="PAWS" version="1.0.0" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
  <paws:Query namespace="xmlns:xacml=urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
    /xacml:PolicySet[xacml:PolicySetId="urn:mydomain:12345"]
```

```
  </paws:Query>
</paws:DeletePolicyElement>

<paws:DeletePolicyElementResponse timeStamp="2013-04-01T21:00:00Z" ...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
  <paws:Query namespace="xmlns:xacml=urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
    /xacml:PolicySet[xacml:PolicySetId="urn:mydomain:12345"]
  </paws:Query>
</paws:DeletePolicyElementResponse>
```

Example 4: SelectPolicyElement request and response

```
<paws:SelectPolicyElement service="PAWS" version="1.0.0"...>
  <paws:PolicyStoreId>urn:mydomain:MyFirstPolicyStore</paws:PolicyStoreId>
  <paws:PolicyContainerId>urn:mydomain:MyFirstContainer</paws:PolicyContainerId>
  <paws:Query namespace="xmlns:xacml=urn:oasis:names:tc:xacml:3.0:core:schema:wd-17">
    /xacml:PolicySet[xacml:PolicySetId="urn:mydomain:12345"]
  </paws:Query>
  <paws:Dereference timeOut="300">local</paws:Dereference>
</paws:SelectPolicyElement>

<paws:PolicyElementCollection timeStamp="2013-04-01T21:00:00Z" ...>
  <paws:XacmlPolicyElement xmlns:xacml="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
    xpath="/xacml:PolicySet[1]">
    <xacml:PolicySet PolicySetId="urn:lam:root-ps:layer:1:12345" PolicyCombiningAlgId="only-one-
       applicable" Version="0.2" ...>
      <xacml:Target/>
    </xacml:PolicySet>
  </paws:XacmlPolicyElement>
</paws:PolicyElementCollection>
```

Annex D
(Normative)

Web Service Description Language (WSDL)

D.1 Introduction

The Web Service Description Language 1.1 [WSDL 1.1] describes WSDL as “an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services).”

This specification defines the WSDL operations, messages and binding for PAWS instances.

This specification defines the following additional constraints on WSDL documents for paws instances to improve interoperability:

The namespace prefix wsdl in this section is bound to http://schemas.xmlsoap.org/wsdl/.

todo