

**Sixth Framework Programme
Priority IST 2.5.12
Information Society Technologies**



Integrated Project



Contract No.: 033564

Specification of the Policy Enforcement Service

Version 1.1

Due date of deliverable: 31/05/2009

Internal release date: 24/03/2009
Actual submission date: 24/03/2009

Document Control Page

Title	Specification of the Policy Enforcement Service	
Creator	EIG	
Editor	Pascal Dihé (EIG)	
Description	This document defines the implementation specification of the Policy Enforcement Service for the SANY Web Service Platform.	
Publisher	SANY Consortium	
Contributors	Martin Scholl (EIG)	
Type	Text	
Format	MS Word	
Language	EN-GB	
Creation date	2008-12-01	
Version number	1.1	
Version date	2009-02-09	
Last modified by	EIG	
Rights	Copyright "SANY Consortium". During the drafting process, access is generally limited to the SANY Partners.	
Audience	<input type="checkbox"/> internal <input checked="" type="checkbox"/> public <input type="checkbox"/> restricted, access granted to:	
Review status	<input type="checkbox"/> Draft <input checked="" type="checkbox"/> WP Manager accepted <input type="checkbox"/> SP Manager accepted <input type="checkbox"/> MB quality controlled <input type="checkbox"/> Co-ordinator accepted	Where applicable: <input type="checkbox"/> Accepted by the GA <input type="checkbox"/> Accepted by the GA as public document
Action requested	<input type="checkbox"/> to be revised by Partners involved in the preparation of the Project Deliverable <input type="checkbox"/> to be revised by all SANY Partners <input type="checkbox"/> for approval of the WP Manager <input checked="" type="checkbox"/> for approval of the SP Manager <input type="checkbox"/> for approval of the Quality Manager <input type="checkbox"/> for approval of the Project Co-ordinator <input type="checkbox"/> for approval of the General Assembly	
Requested deadline	<<dd/mm/yyyy >>	

Copyright © 2009, SANY Consortium

The SANY Consortium (www.sany-ip.eu) grants third parties the right to use and distribute all or parts of this document, provided that the SANY project and the document are properly referenced.

THIS DOCUMENT IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS DOCUMENT, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Table of Contents

1. Introduction	8
2. Overview and Architecture Outline.....	8
2.1. Role and Scope of the Policy Enforcement Service	8
2.2. Service Specification Summary.....	10
3. Context of the Policy Enforcement Service	11
3.1. Relations to Standards	11
3.2. Relations to Information Models	11
3.3. Relations to other Service Specifications	11
4. Specification of the Service Capabilities Interface.....	Fehler! Textmarke nicht definiert.
4.1. getCapabilities Operation	Fehler! Textmarke nicht definiert.
4.1.1 OA_GetCapabilitiesRequest Type	14
4.1.2 OA_GetCapabilitiesResponse Type.....	14
5. Specification of the PEP Interface	16
5.1. doRequest Operation	16
5.1.1 doRequest Type.....	17
5.1.2 doRequestResponse Type.....	18
6. References	20
7. Appendix A: XML Schema and WSDL Documents.....	21
7.1. XML Schema Documents.....	21
7.1.1 pep_exceptions.xsd	21
7.1.2 pepcommons_communication_types.xsd	22
7.2. WSDL Document.....	25

Table of Acronyms

OASIS	Organization for the Advancement of Structured Information Standards
ORCHESTRA	Open Architecture and Spatial Data Infrastructure for Risk Management
OSI	ORCHESTRA Service Instance
OGC	Open Geospatial Consortium
PDP	Policy Decision Point
PEP	Policy Enforcement Point
SAML	Security Assertion Markup Language
SensorSA	Sensor Service Architecture
SOAP	Simple Object Access Protocol
WSDL	Web Services Description Language
XACML	eXtensible Access Control Markup Language
XML	eXtensible Markup Language

Tables

Table 1: Specification of the getCapabilities Operation	Fehler! Textmarke nicht definiert.
Table 2: Specification of the doRequest Operation	16

Figures

Figure 1: Non-intrusive web service security	9
Figure 2: SOAP Security (WS-Security)	9
Figure 3: getCapabilities Operation	13
Figure 4: OA_GetCapabilitiesRequest	14
Figure 5: OA_GetCapabilitiesResponse	15
Figure 6: doRequest Operation	17
Figure 7: doRequestRequest	18
Figure 8: doRequestResponse	19

1. Introduction

The present version 1.1 of the Policy Enforcement Service is a new service specification that was developed during the SANY project as an answer to the demand to support non-intrusive security.

This document specifies the interfaces implemented by the Policy Enforcement Service as well as their main purpose and provides a formal description of the implemented operations in WSDL and XML-Schema in conformance to the guidelines and rules of the SANY Web Services Platform defined in the SensorSA [SANY D2.3.3]. The overall Access Control Services Pattern applied in SANY is also described in detail in SensorSA.

The relations to SANY technical requirements and progress of the specification work with respect of the individual project phases are documented in the D2.4.x Deliverables.

2. Overview and Architecture Outline

2.1. Role and Scope of the Policy Enforcement Service

The Policy Enforcement Service is a Policy enforcement point (PEP) that is able to protect an arbitrary web service. It receives a service request to a protected service and requests an authorisation decision from a policy decision point (PDP), usually an instance of the Policy Management and Authorisation Service. Depending on the authorisation decision (currently permit or deny) it forwards the original request to the protected service or returns an error message (access denied).

The advantage of this approach is the ability to provide non-intrusive web service security where the protected service may remain untouched. The main disadvantage of this approach is that the PEP implements a different interface than the protected service and therefore needs extra security-enabled client components. As a consequence the protected service itself is no longer „visible“ and directly accessible within the network.

To overcome this limitation the Service Proxy was specified. It enables both security-enabled and non-security-enabled clients to access a protected web service via the same interface by providing additional security information in a way that the actual service request remains valid for an unsecured service. OASIS standards achieve this by providing security relevant information in the SOAP Header only.

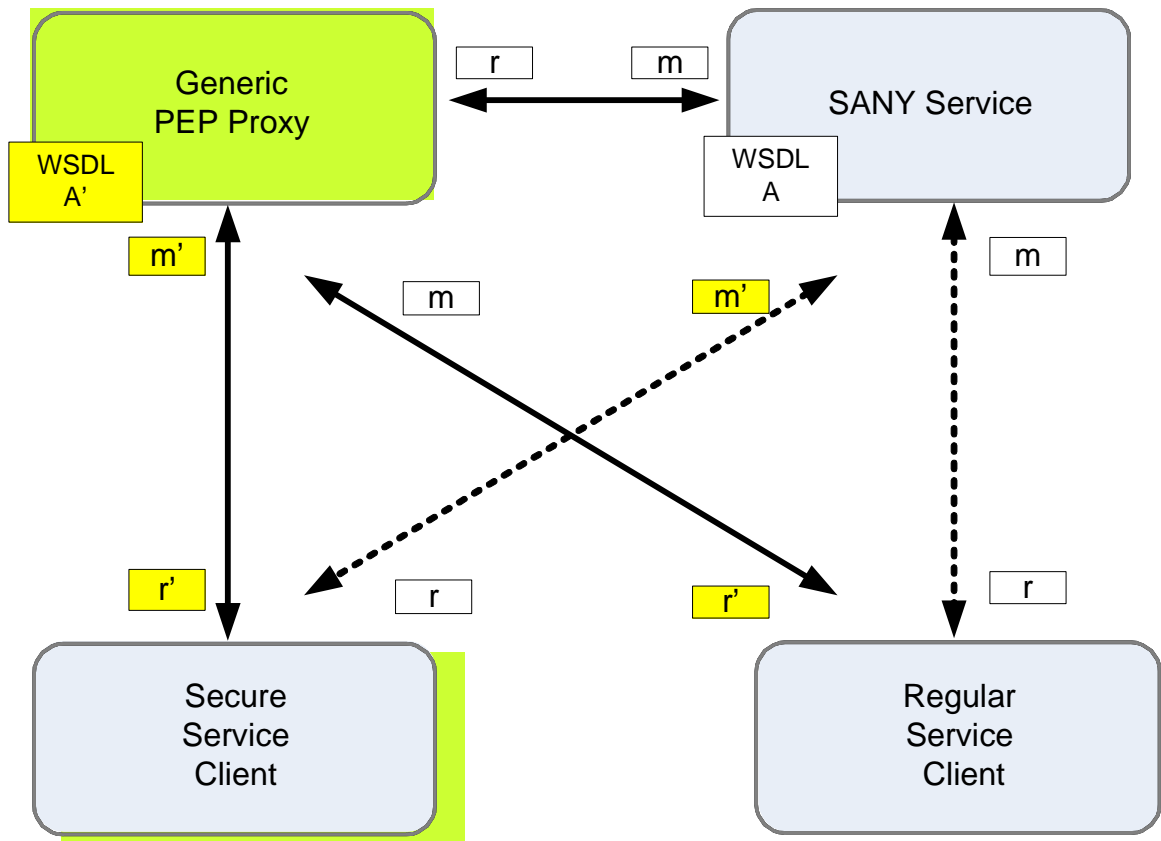


Figure 1: Non-intrusive web service security

To simplify matters the box “Generic PEP Proxy” in the above figure represents the Policy Enforcement Service and the Service Proxy.

In conjunction with OASIS WS-Security standards, the optional security information encoded in SAML is provided in the SOAP header while the actual service request in the SOAP body remains unchanged.

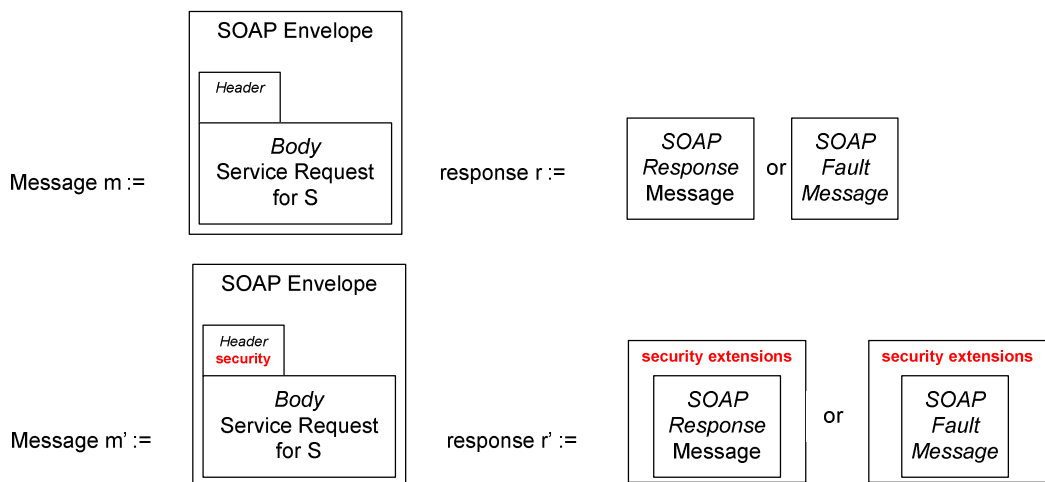


Figure 2: SOAP Security (WS-Security)

Due to the generic OASIS WS-Security based approach the Policy Enforcement Service may be used in conjunction with the Service Proxy to secure any SOAP enabled web service. As already mentioned, the Policy Enforcement Service relies on session information (a SAML token) that has to be provided by a service requestor. A security-enabled client therefore has to interact with the Identity Management and Authentication Service first, before invoking a service through the Service Proxy. To facilitate the integration into existing clients, a component is provided that handles the communication with the Identity Management and Authentication Service and automatically adds the session information to each service request.

2.2. Service Specification Summary

The specification of the Policy Enforcement Service is comprised of the following interfaces that are defined in distinct interface type specifications:

- The Service Capabilities Interface
- The PEP Interface

The formal platform-specific description of the Policy Enforcement Service can be found in Appendix A: XML Schema and WSDL Documents.

3. Context of the Policy Enforcement Service

3.1. Relations to Standards

The Policy Enforcement Service uses the concepts and standards of OASIS WS-Security.

3.2. Relations to Information Models

Apart from the parameter types described in this document, there are no additional relations to information models.

3.3. Relations to other Service Specifications

The Policy Enforcement Service uses the Identity Management and Authentication Service and the Policy Management and Authorisation Service to validate session information (SAML Token) and to request an authorisation decision. It relies furthermore on a service specific Service Proxy if non-intrusive web service security shall be provided.

4. Specification of the Service Capabilities Interface

The Service Capabilities Interface has originally been described in the ORCHESTRA Specification of the OA Basic Service. To maintain backward compatibility to ORCHESTRA Services (e.g. the Catalogue Service), this interface remains unchanged.

4.1. getCapabilities Operation

The specification of the getCapabilities operation has been copied without any modification from the respective ORCHESTRA interface specification. SANY specific changes are only required on the level of the capabilities document itself.

The mandatory getCapabilities operation informs the client of the capabilities of an service instance. This operation takes into account that in addition to capabilities that may be common to all services in a service network a service may provide a specific set of capabilities. Furthermore, this operation allows the capabilities to be delivered according to different service meta-information schemas. This implementation specification therefore does not prescribe a certain schema to be used for the service capabilities. One or several schemas to be supported as well as a default schema have to be defined in the context of a service network.

A service meta-information document is returned to the requesting client, either complete or including selected parts according to the given sections in the request.

A request to perform the getCapabilities operation shall include the parameters listed and defined in Table 1. This table also specifies the data type (Type), the obligation [optional | mandatory] (Use) and a short description (Description) of each listed parameter. Furthermore the “Description” shall state the consequences for service instances if the correspondent parameter is optional and omitted. Although some values listed in the “Name” column appear to contain spaces, they shall not contain spaces.

Compliance	Identical to ORCHESTRA Specification			
Overrides	Not applicable			
Preconditions	none			
Post conditions	Service meta-information document returned to requesting client, either complete or including selected parts according to the given sections in the request			
Use	mandatory			
Receives	Name	Type	Use	Description
	request	OA_GetCapabilities Request	mandatory	Specifies the parts of the meta-information to be returned. If absent, all parts shall be returned using the default schema.

Returns	Type	Description
		OA_CapabilitiesDocument
Throws	Type	Cause
	OA_InvalidParameterValue	Operation request contains an invalid parameter value. Return the name of the parameter with invalid value.
	OA_MissingParameterValue	Operation request does not include a parameter value. Return the name of the missing parameter.
	OA_NoApplicableCode	No other basic or service-specific exception type applies.
	OA_InternalError	A problem occurred in the runtime environment (e.g. out of memory).
	OA_VersionNegotiationFailed	List of versions in acceptSpecVersions parameter value in the getCapabilities request did not include any version supported by the service instance.
	OA_UnsupportedSchema	The sections parameter in the getCapabilities request referred to a schema unsupported by the service instance.

Table 1: Specification of the getCapabilities Operation

The formal platform-specific specification of the getCapabilities operation can be found in Appendix A: XML Schema and WSDL Documents.

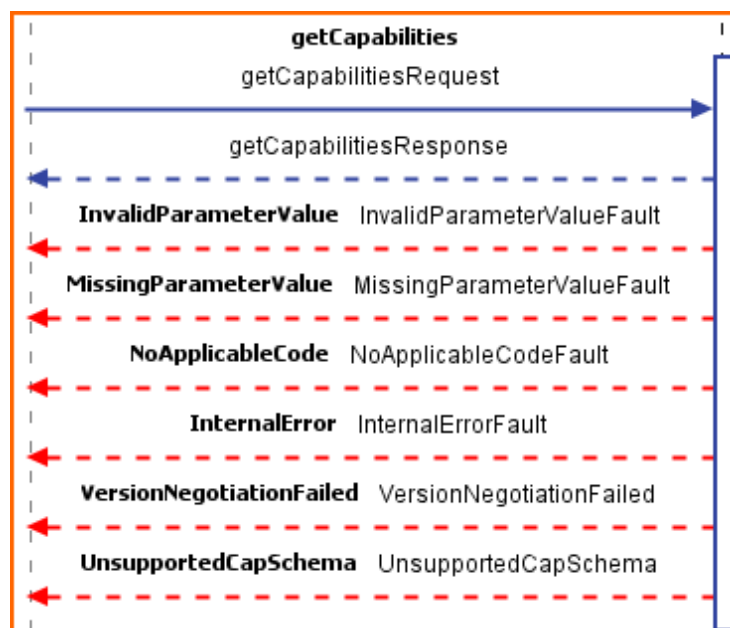


Figure 3: getCapabilities Operation

4.1.1 OA_GetCapabilitiesRequest Type

The OA_GetCapabilitiesRequest Type consists of the following elements:

- **acceptFormats:** Optional parameter containing a prioritized sequence of zero or more response formats desired by the caller, with preferred formats listed first. The formats are to be expressed in terms of MIME types. Independent of the contents of this element, the returned capabilities can always be formatted according to the default MIME type which is defined as “text/xml”.
- **acceptSpecVersions:** Optional parameter containing a prioritized sequence of zero or more specification versions of the service accepted by the client, with the preferred versions listed first. If no versions are included in the request, the highest version that the service supports shall be used.
- **sections:** Optional parameter specifying the schema for service meta-information according to which the capabilities shall be structured. In addition the names of requested sections in the complete set of meta-information elements can be listed. If absent, the complete service capabilities shall be returned structured according to a default schema.



Figure 4: OA_GetCapabilitiesRequest

4.1.2 OA_GetCapabilitiesResponse Type

The OA_GetCapabilitiesResponse Type consists of the following elements:

- **capabilitySections:** Capabilities as meta-information document which is internally structured according to the indicated format and the indicated schema. It is either complete or includes only selected parts according to the given sections in the request.
- **format:** MIME type of the format in which the capabilities are returned as value of the **capabilitySections** parameter. The value of this attribute may always denote the default MIME Type “text/xml” indicating that the capabilities are formatted in XML.
- **schemaName:** Schema according to which the capabilities are returned. The value is the same as the one contained in the **sections** parameter of the request. If not explicitly included in the request, the value indicates the default schema.
- **version:** Version number of the specification to which the delivered service meta-information document is conform.



Figure 5: OA_GetCapabilitiesResponse

5. Specification of the PEP Interface

The PEP Interface is designed to interact with an Identity Management and Authentication and a Policy Management and Authorisation Service. It verifies the genuineness of the security information by calling the Identity Management and Authentication Service and then delegates the evaluation of the access policies to an external policy decision point (PDP), the Policy Management and Authorisation Service.

5.1. doRequest Operation

The mandatory doRequest operation takes as parameter a service request to an unsecured service plus additional security information that are necessary to perform an authorisation request. The doRequest operation verifies the identity(s) of the requestor and asks a Policy Management and Authentication Service if the identity is allowed to perform a defined action (usually the request to the proxied service). If the requestor is permitted to perform the requested action the doRequest delegates the service request to the unsecured service. Otherwise it return an appropriate access denied exception.

A request to perform the doRequest operation shall include the parameters listed and defined in Table 2. This table also specifies the data type (Type), the obligation [optional|mandatory] (Use) and a short description (Description) of each listed parameter. Furthermore the “Description” shall state the consequences for service instances if the correspondent parameter is optional and omitted. Although some values listed in the “Name” column appear to contain spaces, they shall not contain spaces.

Compliance	Not applicable			
Overrides	Not applicable			
Preconditions	None			
Post conditions	None			
Use	Mandatory			
Receives	Name	Type	Use	Description
	request	doRequest Request	mandatory	It contains the original service request plus security information (e.g. SAML tokens).
Returns	Type		Description	
	doRequestResponse		It contains the response of the unsecured service or an error message.	
Throws	Type		Cause	
	OA_InternalError		A problem occurred in the runtime environment (e.g. out of memory).	

Table 2: Specification of the doRequest Operation

The formal platform-specific specification of the doRequest operation can be found in Appendix A: XML Schema and WSDL Documents.

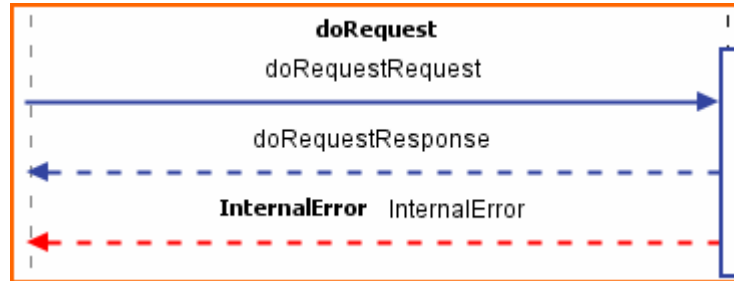


Figure 6: doRequest Operation

5.1.1 doRequest Type

The doRequest Type contains the original service request to the unsecured service, including the original URL of the secured service, the operation to invoke, etc, as well as additional metadata that is required to take an authorisation decision. Usually this metadata consists of SAML tokens and XACML actions.

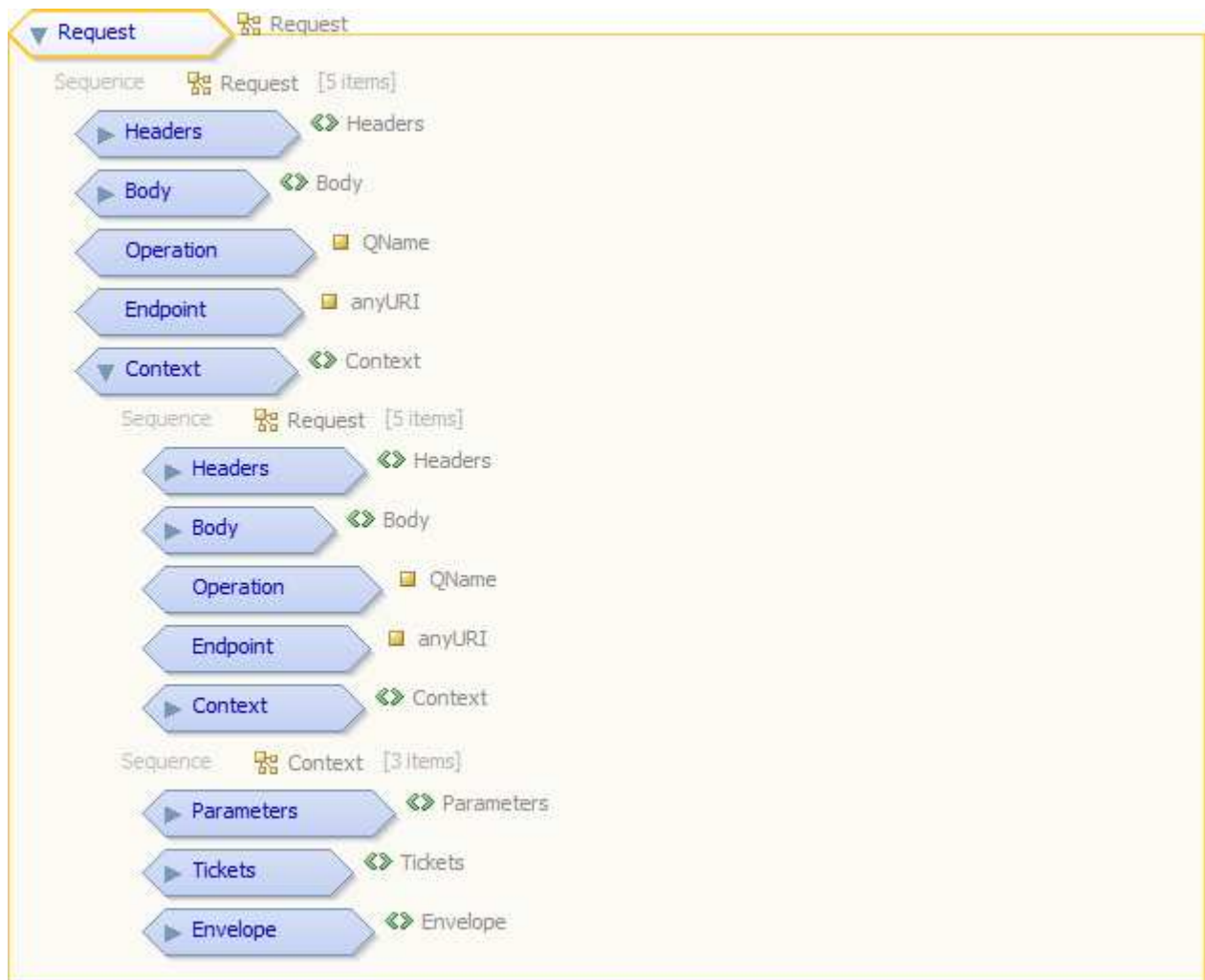


Figure 7: doRequestRequest

5.1.2 doRequestResponse Type

The Response Type consists of the following elements:

- XMLResponse: Contains the unmodified XML response of the unsecured service
- BinaryResponse: Contains the unmodified binary response of the unsecured services (e.g. a binary WMS response)
- ProcessingFault: Contains a detailed error message if an internal error occurred
- PermissionFault: Contains a detailed error message if an authorisation or authentication error occurred (e.g. access denied)



Figure 8: doRequestResponse

6. References

SANY D2.3.3, 2009

SANY D2.3.3 “Specification of the Sensor Service Architecture V2”, SANY IP document; available from SANY website:

<<http://sany-ip.eu/biblio>>

SANY-D2.4.3, 2009

SANY D2.4.3 “Sensor Service Specification V2”, SANY IP document; available from SANY website:

<<http://sany-ip.eu/biblio>>

SANY-Authentication

Specification of the Identity Management and Authentication Service (Version 1.1), SANY Consortium, Editor: Environmental Informatics Group (EIG), available from SANY website:

<<http://www.sany-ip.eu/biblio>>

SANY-Authorisation

Specification of the Policy Management and Authorisation Service (Version 1.1), SANY Consortium, Editor: Environmental Informatics Group (EIG), available from SANY website:

<<http://www.sany-ip.eu/biblio>>

SANY-ProfileManagement

Specification of the Profile Management Service (Version 1.1), SANY Consortium, Editor: Environmental Informatics Group (EIG), available from SANY website:

<<http://www.sany-ip.eu/biblio>>

7. Appendix A: XML Schema and WSDL Documents

7.1. XML Schema Documents

The following XML Schema Documents define the data types of this service.

The XML schema documents of the used data types are bundled in a zip file with the present document and can be downloaded at

<http://repository.sany-ip.eu/svn/schemas/v2/schema/security/>

and from the SANY website

<http://www.sany-ip.eu/>.

In addition to XML Schema Documents specified in this appendix, this specification requires several normative XML Schema Documents. These XML Schema Documents define the common data types, e.g. common Exception Types, Basic Data Types and the GML Profile.

This file contains the XML Schema documents of the OA Basic Service and Policy Enforcement Service.

The namespaces

<http://www.enviromatics.net/WS/PEP/commons/communication/types> and

<http://www.enviromatics.net/WS/PEP/exceptions>

are used.

7.1.1 pep_exceptions.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://www.enviromatics.net/WS/PEP/exceptions"
  xmlns:pex="http://www.enviromatics.net/WS/PEP/exceptions"

  xmlns:pcc="http://www.enviromatics.net/WS/PEP/commons/communication/types"
  elementFormDefault="qualified">
  <import
  namespace="http://www.enviromatics.net/WS/PEP/commons/communication/types"
  schemaLocation="pepcommons_communication_types.xsd" />

  <element name="InternalError" type="pex:InternalError" />
  <complexType name="InternalError">
    <sequence>
      <element name="Message" type="string" />
      <element ref="pex:Stacktrace" />
    </sequence>
  </complexType>

  <element name="Stacktrace" type="pcc:AnyElement" />
</schema>
```

7.1.2 pepcommons_communication_types.xsd

```

<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"

targetNamespace="http://www.enviomatics.net/WS/PEP/commons/communication/types"

xmlns:pcc="http://www.enviomatics.net/WS/PEP/commons/communication/types"
  xmlns:samla="urn:oasis:names:tc:SAML:2.0:assertion"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  elementFormDefault="qualified">
  <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
schemaLocation="http://docs.oasis-open.org/security/saml/v2.0/saml-schema-assertion-2.0.xsd"/>
  <import namespace="http://schemas.xmlsoap.org/soap/envelope/"
schemaLocation="http://schemas.xmlsoap.org/soap/envelope/" />

  <element name="Request" type="pcc:Request" />
  <complexType name="Request">
    <sequence>
      <element ref="pcc:Headers" />
      <element ref="pcc:Body" />
      <element name="Operation" type="QName" />
      <element name="Endpoint" type="anyURI" />
      <element ref="pcc:Context" />
    </sequence>
  </complexType>

  <element name="Headers" type="pcc:Headers" />
  <complexType name="Headers">
    <sequence>
      <element ref="pcc:Header" minOccurs="0" maxOccurs="unbounded" />
    </sequence>
  </complexType>

  <element name="Header" type="pcc:AnyElement" />
  <element name="Body" type="pcc:AnyElement" />
  <complexType name="AnyElement">
    <sequence>
      <any namespace="##other" minOccurs="0" maxOccurs="unbounded"
processContents="lax" />
    </sequence>
  </complexType>

  <element name="Context" type="pcc:Context" />
  <complexType name="Context">
    <complexContent>
      <extension base="pcc:Request">
        <sequence>
          <element ref="pcc:Parameters" />
          <element ref="pcc:Tickets" />
          <element ref="pcc:Envelope" />
        </sequence>
      </extension>
    </complexContent>
  </complexType>

```

```

<element name="Parameters" type="pcc:Parameters" />
<complexType name="Parameters">
  <sequence>
    <element ref="pcc:Parameter" minOccurs="0"
maxOccurs="unbounded" />
  </sequence>
</complexType>

<element name="Parameter" type="pcc:Parameter" />
<complexType name="Parameter">
  <sequence>
    <element name="Value" type="string" />
  </sequence>
  <attribute name="name" type="string" />
</complexType>

<element name="Tickets" type="pcc:Tickets" />
<complexType name="Tickets">
  <sequence>
    <element ref="samla:Assertion" minOccurs="0"
maxOccurs="unbounded" />
  </sequence>
</complexType>

<element name="Envelope" type="pcc:Envelope" />
<complexType name="Envelope">
  <sequence>
    <element ref="soapenv:Envelope" />
  </sequence>
</complexType>

<element name="Response" type="pcc:Response" />
<complexType name="Response">
  <sequence>
    <element ref="pcc:XMLResponse" minOccurs="0" />
    <element ref="pcc:BinaryResponse" minOccurs="0" />
    <element ref="pcc:ProcessingFault" minOccurs="0" />
    <element ref="pcc:PermissionFault" minOccurs="0" />
  </sequence>
</complexType>

<element name="XMLResponse" type="pcc:AnyElement" />

<element name="BinaryResponse" type="string" />

<element name="ProcessingFault" type="pcc:Throwable" />

<element name="PermissionFault" type="pcc:Throwable" />

<element name="Throwable" type="pcc:Throwable" />
<complexType name="Throwable">
  <sequence>
    <element name="Message" type="string" minOccurs="0" />
    <element ref="pcc:StackTrace" />
    <element name="Cause" type="pcc:Throwable" minOccurs="0" />
  </sequence>
  <attribute name="class" type="string" />
</complexType>

<element name="StackTrace" type="pcc:StackTrace" />

```

```
<complexType name="StackTrace">
  <sequence>
    <element ref="pcc:StackTraceElement" maxOccurs="unbounded"/>
  </sequence>
</complexType>

<element name="StackTraceElement" type="pcc:StackTraceElement"/>
<complexType name="StackTraceElement">
  <sequence>
    <element name="DeclaringClass" type="string"/>
    <element name="MethodName" type="string"/>
    <element name="FileName" type="string" minOccurs="0"/>
    <element name="LineNumber" type="integer" minOccurs="0"/>
  </sequence>
</complexType>
</schema>
```

7.2. WSDL Document

The following WSDL Version 1.1 document is the formal specification of the Policy Enforcement Service according to rules of the “SANY Web Services Platform”. It defines the mandatory SOAP binding.

The WSDL document is bundled in a zip file with the present document and can be downloaded at

<http://repository.sany-ip.eu/svn/schemas/v2/wSDL/security/> and from the SANY website

<http://www.sany-ip.eu/>.

```
<?xml version="1.0"?>
<wSDL:definitions xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wSDL/soap/"
  xmlns:http="http://schemas.xmlsoap.org/wSDL/http/"
  xmlns:mime="http://schemas.xmlsoap.org/wSDL/mime/"
  xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"
  xmlns:pep="http://www.enviroomatics.net/WS/PEP/PEPService"

  xmlns:pcc="http://www.enviroomatics.net/WS/PEP/commons/communication/types"
  xmlns:pex="http://www.enviroomatics.net/WS/PEP/exceptions"
  name="PEPServiceInstance"
  targetNamespace="http://www.enviroomatics.net/WS/PEP/PEPService"
  xmlns:plnk="http://docs.oasis-open.org/wsbpel/2.0/plnktype">
  <wSDL:types>
    <xs:schema
      targetNamespace="http://www.enviroomatics.net/WS/PEP/PEPService"
      xmlns:xs="http://www.w3.org/2001/XMLSchema"
      elementFormDefault="qualified"
      attributeFormDefault="qualified">
      <xs:import
        namespace="http://www.enviroomatics.net/WS/PEP/commons/communication/types"
        schemaLocation="pepcommons_communication_types.xsd"/>
      <xs:import
        namespace="http://www.enviroomatics.net/WS/PEP/exceptions"
        schemaLocation="pep_exceptions.xsd"/>
    </xs:schema>
  </wSDL:types>
  <!--wSDL:message name="getCapabilitiesRequest">
    <wSDL:part name="request"
  element="oab_types:OA_GetCapabilitiesRequest"/>
  </wSDL:message>
  <wSDL:message name="getCapabilitiesResponse">
    <wSDL:part name="response"
  element="oab_types:OA_GetCapabilitiesResponse"/>
  </wSDL:message-->
  <wSDL:message name="doRequestRequest">
    <wSDL:part name="request" element="pcc:Request"/>
  </wSDL:message>
  <wSDL:message name="doRequestResponse">
    <wSDL:part name="response" element="pcc:Response"/>
  </wSDL:message-->
</wSDL:definitions>
```

```

</wsdl:message>
<wsdl:message name="InternalError">
  <wsdl:part name="InternalError" element="pex:InternalError"/>
</wsdl:message>
<wsdl:portType name="PEPService">
  wsdl:operation name="getCapabilities">
    <wsdl:input name="capabilitiesRequest"
message="authen:getCapabilitiesRequest"/>
    <wsdl:output name="capabilitiesResponse"
message="authen:getCapabilitiesResponse"/>
    <wsdl:fault name="InvalidParameterValue"
message="authen:InvalidParameterValueFault"/>
    <wsdl:fault name="MissingParameterValue"
message="authen:MissingParameterValueFault"/>
    <wsdl:fault name="NoApplicableCode"
message="authen:NoApplicableCodeFault"/>
    <wsdl:fault name="InternalError"
message="authen:InternalErrorFault"/>
    <wsdl:fault name="VersionNegotiationFailed"
message="authen:VersionNegotiationFailed"/>
    <wsdl:fault name="UnsupportedCapSchema"
message="authen:UnsupportedCapSchema"/>
  </wsdl:operation>
  <wsdl:operation name="doRequest">
    <wsdl:input name="doRequestRequest"
message="pep:doRequestRequest"/>
    <wsdl:output name="doRequestResponse"
message="pep:doRequestResponse"/>
    <wsdl:fault name="InternalError" message="pep:InternalError"/>
  </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="PEPService" type="pep:PEPService">
  <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
  <!--wsdl:operation name="getCapabilities">
    <soap:operation soapAction="getCapabilities" style="document"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
    <wsdl:output>
      <soap:body use="literal"/>
    </wsdl:output>
    <wsdl:fault name="InvalidParameterValue">
      <soap:fault name="InvalidParameterValue" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="MissingParameterValue">
      <soap:fault name="MissingParameterValue" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="NoApplicableCode">
      <soap:fault name="NoApplicableCode" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="InternalError">
      <soap:fault name="InternalError" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="UnsupportedCapSchema">
      <soap:fault name="UnsupportedCapSchema" use="literal"/>
    </wsdl:fault>
    <wsdl:fault name="VersionNegotiationFailed">
      <soap:fault name="VersionNegotiationFailed" use="literal"/>
    </wsdl:fault>
  </wsdl:operation>

```

```
</wsdl:operation-->
<wsdl:operation name="doRequest">
  <soap:operation soapAction="doRequest" style="document"/>
  <wsdl:input>
    <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
    <soap:body use="literal"/>
  </wsdl:output>
  <wsdl:fault name="InternalServerError">
    <soap:fault name="InternalServerError" use="literal"/>
  </wsdl:fault>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="PEPService">
  <wsdl:port name="PEPService" binding="pep:PEPService">
    <soap:address
location="http://localhost:8080/axis2/services/PEPService"/>
  </wsdl:port>
</wsdl:service>
<plnk:partnerLinkType name="PEPService">
  <plnk:role name="role1" portType="pep:PEPService"/>
</plnk:partnerLinkType>
</wsdl:definitions>
```