



HERAS^{AF}
PDP - PEP

User's Guide
HERAS^{AF}: PEP-PDP Communication

Department of Computer Science
University of Applied Sciences Rapperswil (HSR)
Spring Semester 2008

Artifact of HERAS^{AF}: PEP and PDP Web Service Integration [RW08PEP]

Students: Daniel Regli, Yannick Winiger

Examiner: Wolfgang Giersche

Coaches: René Eggenschwiler, Florian Huonder

Table of Contents

| | |
|------------------------------------------------------------------------------|-----------|
| PART I: INTRODUCTION | 3 |
| 1 INTENDED AUDIENCE..... | 4 |
| 2 OVERVIEW | 5 |
| 2.1 DEFINITION..... | 5 |
| 3 RUNTIME ENVIRONMENT | 7 |
| PART II: HERAS^{AF} PDP SERVER INTEGRATION | 8 |
| 1 OVERVIEW | 9 |
| 2 INSTALLATION AND CONFIGURATION | 9 |
| 2.1 WEB SERVER..... | 9 |
| 2.2 PDP | 10 |
| 2.3 PDP CONTEXT WEB SERVICE | 10 |
| 2.3.1 Deployment Descriptor | 10 |
| 2.3.2 Spring Application Context..... | 11 |
| 2.3.3 WSDL | 14 |
| 3 REFERENCE AND CONFIGURATION | 16 |
| 3.1 SAML HANDLING | 16 |
| 3.2 SPRING WEB SERVICE | 17 |
| PART III: HERAS^{AF} PEP WEB SERVICE CLIENT INTEGRATION | 18 |
| 1 OVERVIEW | 19 |
| 2 INSTALLATION & CONFIGURATION | 19 |
| 2.1 HERAS ^{AF} PEP | 19 |
| 2.2 HERAS ^{AF} PEP WS MODULE | 19 |
| 3 REFERENCE AND CONFIGURATION | 23 |
| 3.1 SAML HANDLING | 23 |
| 3.2 SPRING WEB SERVICE | 24 |
| APPENDIX A: GENERAL | 25 |
| 1 GLOSSARY | 26 |
| 2 BIBLIOGRAPHY | 28 |
| 2.1 SPECIFICATIONS AND STANDARDS | 28 |
| 2.2 HERAS ^{AF} DOCUMENTS | 28 |
| 2.3 OTHER RESOURCES | 30 |

Part I: Introduction

1 Intended audience

General

This User's Guide provides instructions for using and installation of the HERAS^{AF} PDP Context-WS. The guide will assist users to install and operate on the HERAS^{AF} PDP Context-WS.

The users should bring basic understanding in the following technologies, frameworks and standards:

- Java 1.5
- HERAS^{AF}
- Spring Framework (Core [Spring], Spring Web Services [SpringWS])
- XACML 2.0 [XACML]
- SAML 2.0 [SAML]

Installation instructions are only given for HERAS^{AF} components. References to installations guide are provided whenever the component needs an additional installation of another vendor.

2 Overview

2.1 Definition

| | |
|-------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>HERAS^{AF}</i> | <p>HERAS^{AF} is an Open Source project in its formation phase.</p> <p>It assists the entire authorization process based on the OASIS XACML 2.0 standard. That means all accesses to protected resources are intercepted by a PEP and sent to a PDP, which evaluates the request on the basis of applicable Evaluatables. In case of a positive answer the PEP allows the access to the resource. Additionally the maintenance is possible through a PAP.</p> |
| <i>HERAS^{AF} PEP</i> | <p>The HERAS^{AF} PEP provides mechanisms to implement intercepting agent's which enforces access control in protection worth applications. Its main responsibility is building authorization requests and interpreting authorization responses.</p> |
| <i>HERAS^{AF} PEP Web Service</i> | <p>The HERAS^{AF} PEP Web Service (WS) implements a Web Service client that can be integrated into the HERAS^{AF} PEP architecture. This integration makes it possible that a PEP can send decision requests to a Web Service interface of a remote PDP.</p> |
| <i>HERAS^{AF} PDP</i> | <p>HERAS^{AF} PDP implements the entire logic for decision making of accesses. This covers fast locating of potential policies, evaluation of a request with the aid of the located policies as well as combining the obtained results.</p> <p>The main focus of the HERAS^{AF} PDP is performance. To achieve this, performance-enhancing mechanisms such as indexing and fast comparison algorithms are developed. Another essential point is accessing the data sink only during the initialization phase of the PDP. Afterwards the policies are kept in the RAM.</p> <p>The HERAS^{AF} PDP will be used as a central unit, which interacts with several PEPs. [DOH07DADev]</p> |
| <i>HERAS^{AF} PDP Context-WS</i> | <p>HERAS^{AF} PDP Context-WS is part of a HERAS^{AF} PDP and implements a Web Service endpoint for a Spring Web Service to process decision requests sent by PEPs.</p> |
| <i>HERAS^{AF} PAP</i> | <p>The HERAS^{AF} PAP is responsible to build and administrate policies. Additionally it deploys the policies to the PDP's. For the building process of complex policies a graphical user interface was developed. For further information to this component see the paper HERAS^{AF} PAP. [NZ08PAPDev]</p> |
| <i>HERAS^{AF} PIP</i> | <p>The HERAS^{AF} PIP is responsible to resolve missing attributes of the request. If a HERAS^{AF} PDP needs further information to evaluate a request, the HERAS^{AF} PIP is called. If possible the HERAS^{AF} PIP returns additional attributes.</p> |
| <i>Spring Web Service</i> | <p>Spring Web Services is a product of the Spring community focused on creating document-driven Web services.</p> <p>Spring Web Services aims to facilitate contract-first SOAP service development,</p> |

allowing for the creation of flexible web services using one of the many ways to manipulate XML payloads. [SpringWS]

SAML

SAML [SAML] is a product of the OASIS Security Services Technical Committee.

Security Assertion Markup Language (SAML) is an XML standard for exchanging authentication and authorization data between security domains, that is between an identity provider (a producer of assertions) and a service provider (a consumer of assertions).

XACML

XACML is an OASIS standard that describes both a policy language and an access control decision request/response language (both written in XML). The policy language is used to describe general access control requirements, and has standard extension points for defining new functions, data types, combining logic, etc. The request/response language lets you form a query to ask whether or not a given action should be allowed, and interpret the result. The response always includes an answer about whether the request should be allowed using one of four values: Permit, Deny, Indeterminate (an error occurred or some required value was missing, so a decision cannot be made) or Not Applicable (the request can't be answered by this service). [XACML]

3 Runtime environment

General The HERAS^{AF} PEP-PDP Communication components run within a standard Java 1.5 Runtime Environment.

Modules It consists of the following modules.

herasaf-pep-ws.jar

The module contains the classes to connect the HERAS^{AF} PEP to a PDP Web Service. This includes a basic SAML Handler and classes to guarantee the standard-conformance of SAML in SOAP. [SAMLBind]

herasaf-pdp-ws-context.jar

The module contains the classes to connect the Web Service Endpoint to a PDP Implementation. The processing of the whole SAML/XACML request is provided by this module.

herasaf-saml.jar

This module contains common SAML handling and processing classes. These are in a separate module because they are used by other HERAS^{AF} sub projects as well.

Module Dependencies The following figure illustrates the HERAS^{AF} PEP-PDP Communication modules and the dependencies between them. Arrows indicate dependencies, i.e. HERAS^{AF} PEP WS depends on HERAS^{AF} SAML.

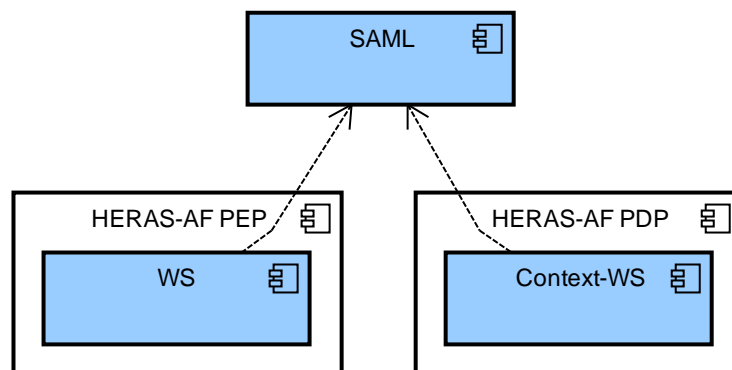


Figure 1: Dependencies of modules

Part II: HERAS^{AF} PDP server integration

1 Overview

Introduction

This chapter describes the integration and installation of the HERAS^{AF} PDP Context Web Service endpoint.

Coverage

Figure 2 shows what the documentation covers. The red framed elements are what this part describes.

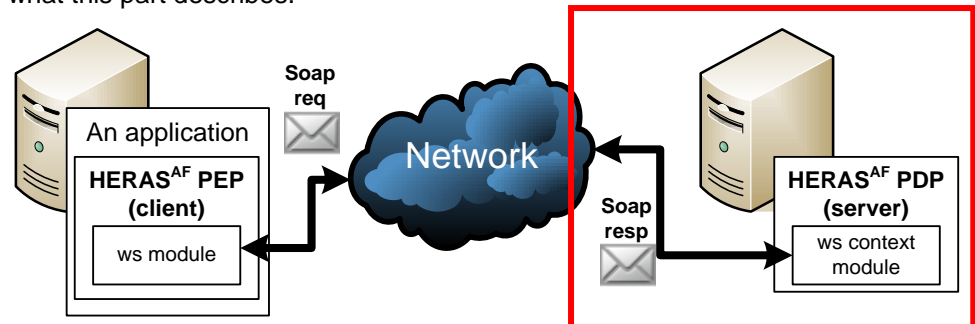


Figure 2: Coverage overview of HERAS^{AF} PDP Context-WS

2 Installation and Configuration

Overview

This part explains which components are required to be installed to run the HERAS^{AF} PDP Context Web Service Endpoint.

The following components are needed to successfully run a HERAS^{AF} PDP Context Web Service Endpoint:

- Web Server with Servlet container
- PDP

The components, which are explained or referenced, are the test components during the development. These are the recommended components to use.

2.1 Web Server

Overview

The HERAS^{AF} PDP Context Web Service Endpoint was developed with Spring Web Service [SpringWS] which needs a Web Service with a Servlet container. There are no additional requirements to the Web Server. Apache Tomcat 5.5x [ApacheTomcat] is the recommend Web Server.

Apache Tomcat 5.5.x

A default installation of an Apache Tomcat is sufficient. As mention before this guide does not provide an installation instruction for other vendor components. See the [ApacheTomcat] Web Page for an installation guide.

2.2 PDP

Overview The HERAS^{AF} PDP Context Web Service Endpoint connects a HERAS^{AF} XACML PDP to the Web Service Endpoint. In this release only HERAS^{AF} XACML PDP's are supported to be connected.

HERAS^{AF} XACML PDP See the HERAS^{AF} XACML 2.0 Developer's Guide for installation instructions. [DOH07DADev]

2.3 PDP Context Web Service

Overview The Configuration of the Web Service itself is contained in minimum 3 different configuration files. The WSDL [WSDL], Application Context file and Deployment Descriptor.

The WSDL describes the Web Service for clients and the application context file describes the components wiring of the Web Service. In addition to that every Web Application needs a Deployment Descriptor (web.xml) which must reside inside the WEB-INF subdirectory.

Recommended configuration structure Example 1 shows the recommended configuration structure of the Web Service.

```

/ (root of Web Service)
/...
/WEB-INF/config/[your Sub Application Context files]
/WEB-INF/context/[your Main Application Context file]
/WEB-INF/schema/[your schemas of the Web Service]
/WEB-INF/wsdL/[your WSDL]
/...

```

Example 1: Recommended Web Service directory structure

2.3.1 Deployment Descriptor

Configuration The web.xml is the entry point of all configuration files. Mostly it does not be changed.

Example 2 shows a minimal configuration of a web.xml. The description of the elements follows beneath.

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
version="2.4">
  <display-name>HERAS-AF PDP Context-WS</display-name>
  <servlet>
    <servlet-name>spring-ws</servlet-name>
    <servlet-class>org.springframework.ws.transport.
http.MessageDispatcherServlet</servlet-class>
  </servlet>

```

```
<servlet-mapping>
  <servlet-name>spring-ws</servlet-name>
  <url-pattern>/*</url-pattern>
</servlet-mapping>
</web-app>
```

Example 2: Minimal Configuration of web.xml

Element
description and
values

General Element description

| | |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>servlet-name</code> | <p>Defines the canonical name of the servlet, used to reference the servlet definition elsewhere in the deployment descriptor.</p> <p>The canonical name decides the name of the application context name. (e.g. <code>servlet-name</code> is <code>spring-ws</code>, the resulting application context file for Spring WS is <code>spring-ws-servlet.xml</code>)</p> |
| <code>servlet-class</code> | <p>The fully-qualified class name of the servlet. This must not be changed.</p> |

2.3.2 Spring Application Context

Common
elements

All Spring configuration files [Spring] have the following common elements.

Beans namespaces

```
<beans xmlns=
"http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation= "http://www.springframework.org/
schema/beans
http://www.springframework.org/schema/beans/spring-beans-
2.5.xsd">
```

Example 3: Common elements of Spring configuration files

Configuration

The name of the application context file depends on the name defined in the Deployment Descriptor's `servlet-name`. It must reside in the `WEB-INF` directory next to the Deployment Descriptor.

In the Deployment Descriptor chapter an example with the `servlet-name` was outlined and the resulting Spring application context file was `spring-ws-servlet.xml`. That filename refers to the Spring application context file in the following chapters.

Example 4 shows a configuration of a `spring-ws-servlet.xml`. The description of the elements follows beneath.

```
<context:annotation-config />
<import resource="./context/PDPContext.xml"/>

<bean id="endpoint"
class="org.herasaf.pdp.ws.context.saml.
SamlPdpEndpoint">
```

```

        <property name="samlHandler" ref="samlHandler" />
    </bean>

    <bean id="samlHandler"
        class="org.herasaf.pdp.ws.context.saml.
        DefaultServerSamlHandler">
        <property name="pdp" ref="pdpImpl" />
        <property name="issuer">
            <bean
                class="org.herasaf.saml.jaxb.saml.assertion.
                NameIDType">
                <property name="value"
                    value="HERAS-AF PDP" />
            </bean>
        </property>
    </bean>

    <bean id="service"
        class="org.springframework.ws.wsdl.wsdl11.
        SimpleWsdl11Definition">
        <constructor-arg value="/WEB-INF/wsdl/herasaf-
        ws.wsdl">
        </constructor-arg>
    </bean>

    <bean id="mapping"
        class="org.springframework.ws.soap.addressing.
        server.SimpleActionEndpointMapping">
        <property name="mappings">
            <util:map>
                <entry key="http://localhost:8080/herasaf-
                pdp-ws-context/pdp-service/evaluate" value="endpoint" />
            </util:map>
        </property>
    </bean>

```

Example 4: Configuration a spring-ws-servlet.xml*Hint*

This configuration file needs an additional namespace import.

```

xmlns:util="http://www.springframework.org/schema/util"
xsi:schemaLocation=
"http://www.springframework.org/schema/util
http://www.springframework.org/schema/util/spring-util-
2.5.xsd"

```

Example 5: Additional namespace import*Element description and values***General Element description**

| | |
|------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre><context:annotation-config /></pre> | <p>Activates various annotations to be detected in bean classes: Spring's @Required and @Autowired, as well as JSR 250's @PostConstruct, @PreDestroy and @Resource (if available) and JPA's @PersistenceContext and @PersistenceUnit.</p> |
|------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

This must be included.

`<import resource...` Imports an additional application context file.

endpoint bean description

| | |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description: | The SAML PDP endpoint unmarshals the request payload, and marshals the response object. The endpoint provides access to the SAML handlers. The class must not be changed. |
| Property | <code>samlHandler</code> Sets the SAML Handler implementation. (required) |

samlHandler bean description

| | |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description: | The SAML handler handles incoming SAML requests and builds a corresponding SAML response [SAML]. The class can be changed as long as the class implements <code>ServerSamlHandler</code> interface. See 3.1 for further information. |
| Properties | <code>pdp</code> Sets the PDP which should be connected to the handler. (required) <code>issuer</code> Sets the issuer of the SAML response. (required) |

service bean description

| | |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description: | This bean is used for the automatic WSDL exposure. The bean id (e.g. <code>service</code>) defines the name of the exposed WSDL description file. For example the bean id is <code>service</code> then the WSDL is exposed under the name <code>service.wsdl</code> in the main servlet context directory. (<code>http://hostname/servlet-context/service-wsdl</code>). The class must not be changed. |
| Constructor-arg | <code>/WEB-INF/wsdl/herasaf-ws.wsdl</code> Sets the WSDL file path. (required) |

mapping bean description

| | |
|--------------|------------------------------------------------|
| Description: | This bean maps incoming Web Service request to |
|--------------|------------------------------------------------|

an Endpoint using WS-Addressing.

Properties

mappings

A map where the URI is the key and the value the endpoint. The URI represents the action address of WS-Addressing. (required)

2.3.3 WSDL

Configuration

The WSDL provides a model to describe the Web Service. There are tools which are able to generate stubs for applications from a WSDL description.

Example 6 shows a WSDL configuration for the HERAS^{AF} PDP Context Web Service.

```
<definitions targetNamespace="http://schema.herasaf.org/
herasaf-pdp-ws-context"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:tns="http://schema.herasaf.org/herasaf-pdp-ws-
context"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
xmlns:xacml-samlp="urn:oasis:names:tc:xacml:2.0:profile:
saml2.0:v2:schema:protocol">

<import
location="http://schemas.herasaf.org/saml/2.0/saml-
schema-protocol-2.0.xsd"
namespace="urn:oasis:names:tc:SAML:2.0:protocol" />
<import
location="http://schemas.herasaf.org/xacml/2.0/xacml-2.0-
profile-saml2.0-v2-schema-protocol-wd-5.xsd"
namespace="urn:oasis:names:tc:xacml:2.0:profile:
saml2.0:v2:schema:protocol" />

<message name="Request">
  <part name="XACMLAuthzDecisionQuery"
element="xacml-samlp:XACMLAuthzDecisionQuery" />
</message>

<message name="Response">
  <part name="Response" element="samlp:Response" />
</message>

<portType name="Pdp">
  <operation name="evaluate">
    <input message="tns:Request" name="Request"/>
    <output message="tns:Response"
name="Response" />
  </operation>
</portType>

<binding name="PdpBinding" type="tns:Pdp">
  <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http" />
```

```

        <operation name="evaluate">
            <input></input>
            <output></output>
        </operation>
    </binding>

    <service name="PdpService">
        <documentation>HERAS-AF PDP Service</documentation>
        <port name="PdpPort" binding="tns:PdpBinding">
            <soap:address
                location="http://localhost:8080/herasaf-pdp-ws-
                context/pdp-service" />
        </port>
    </service>
</definitions>

```

Example 6: WSDL Configuration

Element description and values

This guide will not give an explanation on each element in the WSDL. See [WSDL] for further information. The most important and configurable elements are mentioned below.

soap:address element description

| | |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description: | The SOAP address binding is used to give a port an address (a URI). A port using the SOAP binding must specify exactly one address. The URI scheme specified for the address must correspond to the transport specified by the soap:binding . |
| Property | location Sets URI of the Web Service. (required) |

3 Reference and Configuration

Overview

This part details the various components that comprise HERAS^{AF} PDP Context Web Service Endpoint.

This includes a chapter that describes SAML handling and a chapter on using WS-Security provided by Spring Web Service.

The logical components of a HERAS^{AF} PDP Context Web Service Endpoint. Are outlined in **Figure 3**.

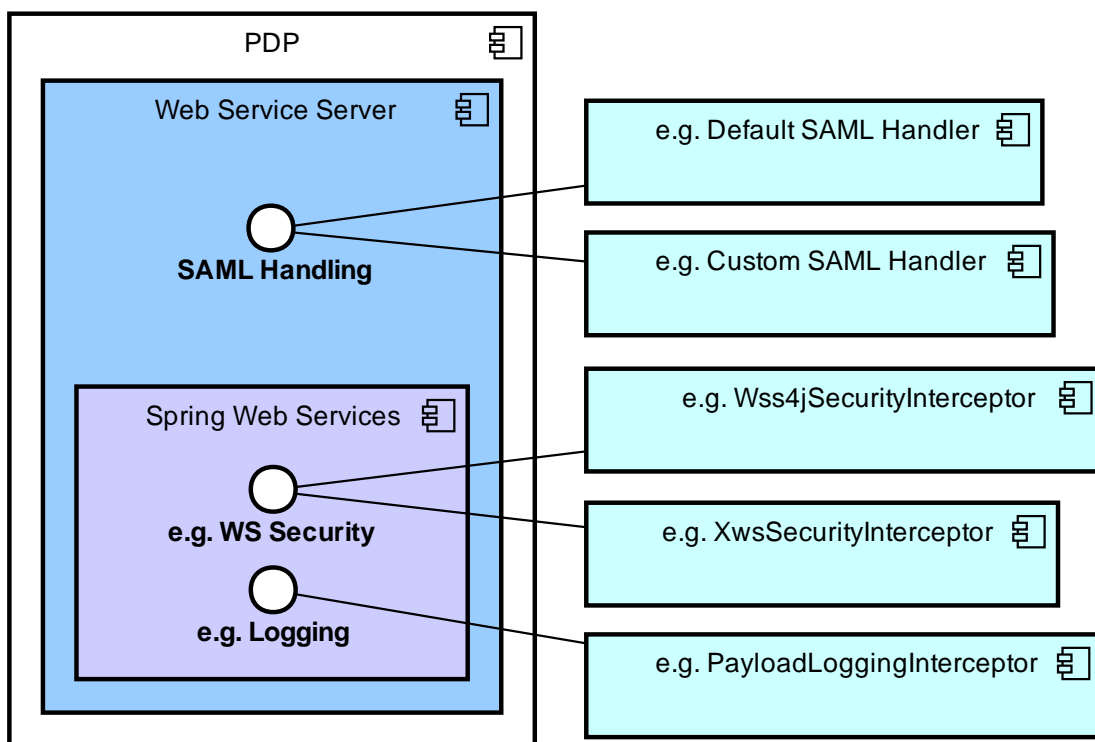


Figure 3: Logical components of a PDP Context Web Service Endpoint

3.1 SAML handling

Overview

SAML handling is an adaption point of the PDP Context Web Service Endpoint. It allows adding custom SAML handling classes, which for example allow handling digital signatures inside SAML.

For this release only a default implementation of a `ServerSamlHandler` exists. This implementation does not handle any custom SAML fields.

Wiring

Example 7 shows how to wire a `DefaultServerSamlHandler` into an Endpoint.

```
<bean id="serverSamlHandler"
      class="org.herasaf.pdp.ws.context.saml.DefaultServe
```

```
rSamlHandler">
  <property name="pdp" ref="pdpImpl" />
  <property name="issuer">
    <bean
      class="org.herasaf.saml.jaxb.saml.assertion.NameIDT
ype">
      <property name="value"
        value="HERAS-AF Policy Deployment Point" />
    </bean>
  </property>
</bean>
```

Example 7: How to wire the DefaultServerSamlHandler

3.2 Spring Web Service

Additional features

The HERAS^{AF} PDP Context Web Service Endpoint is build with Spring Web Service Framework. This enables a lot of additional features for the Web Service which can be simply configured in the application context.

Just to mention some of the additional features of Spring Web Service Framework:

- Supports WS-Security: WS-Security allows signing SOAP messages, encrypting and decrypting them, or authenticating against them.
- Integrates with Spring Security: The WS-Security implementation of Spring Web Services provides integration with Spring Security. This means an existing Spring Security configuration can be used for the SOAP service as well.

These features and their configuration are not described in this guide. See the Spring Web Service website for further information. [SpringWS]

Part III: HERAS^{AF} PEP Web Service client integration

1 Overview

Introduction This chapter describes the integration and installation of the HERAS^{AF} PEP Web Service module.

Coverage Figure 2 shows what the documentation covers. The red framed elements are what this part describes.

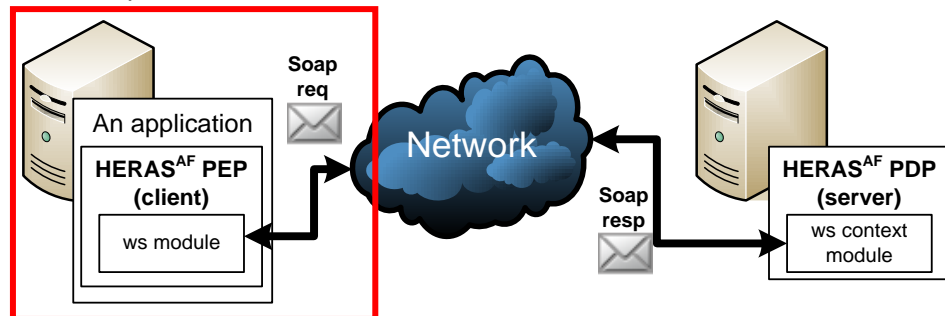


Figure 4: Coverage overview of HERAS^{AF} PEP Web Service module

2 Installation & Configuration

Overview This part explains the installation and configuration of the HERAS^{AF} PEP Web Service module.

The following components are needed to successfully run the module:

- HERAS^{AF} PEP installation and configuration
- XACML Web Service Endpoint

2.1 HERAS^{AF} PEP

Reference See the HERAS^{AF} PEP User's Guide for installation instructions. [RW08PEPUsr]

2.2 HERAS^{AF} PEP WS module

Overview This part only describes the HERAS^{AF} PEP WS client configuration and its integration into the HERAS^{AF} PEP. All other parts of the HERAS^{AF} PEP configuration are described in [RW08PEPDev].

Configuration It is recommended to do the configuration inside the Pdp.xml application context as mentioned in the HERAS^{AF} PEP User's Guide.

Example 8 shows a configuration of a Web Service module configuration. The description of the elements follows beneath.

```
<bean id="wsPdp"
```

```

class="org.herasaf.pep.integration.ws.SamlPdp">
    <property name="clientSamlHandler"
        ref="clientSamlHandler" />
</bean>

<bean id="clientSamlHandler"
    class="org.herasaf.pep.integration.ws.saml.Default
tClientSamlHandler">
    <property name="remotePdpClient"
ref="springWsClient" />
</bean>

<bean id="springWsClient"
    class="org.herasaf.pep.integration.ws.springws.Sp
ringWsClient">
    <property name="marshaller" ref="marshaller" />
    <property name="unmarshaller" ref="marshaller" />
    <property name="defaultUri"
        value="http://example.com/pdp-service" />
    <property name="actionUri"
value="http://example.com/pdp-service/evaluate" />
    <property name="messageSenders">
        <bean
class="org.springframework.ws.transport.http.CommonsHttp
pMessageSender">
            <property name="httpClient">
                <bean
class="org.apache.commons.httpclient.HttpClient">
                    <property name="params"
ref="soapSamlHttpClientParams"
/>
                </bean>
            </property>
        </bean>
    </property>
    <property name="interceptors">
        <bean
class="org.herasaf.saml.springws.SoapActionClient
Interceptor" />
    </property>
</bean>

<bean id="marshaller"
    class="org.herasaf.saml.marshaller.Jaxb2Namespace
Marshaller">
    <property name="contextPath"
        value="org.herasaf.xacml.core.context.impl:org.he
rasaf.saml.jaxb.saml.assertion:org.herasaf.saml.jaxb.sa
ml.protocol:org.herasaf.saml.jaxb.w3.xmlnsig:org.herasa
f.saml.jaxb.w3.xmlenc:org.herasaf.saml.jaxb.xacml.asser
tion:org.herasaf.saml.jaxb.xacml.protocol" />
</bean>

<bean id="soapSamlHttpClientParams"
    class="org.herasaf.saml.springws.SoapSamlHttpClie
ntParams" />

```

Example 8: HERAS^{AF} PEP WS configuration

*Element
description and
values*

[wsPdp](#) bean description

| | |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| Description: | This bean redirects the evaluation of the XACML request to a remote PDP using a transport technology. The class must not be changed. |
| Property | clientSamlHandler Sets the SAML Handler implementation. (required) |

[clientSamlHandler](#) bean description

| | |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description: | The SAML handler handles incoming SAML requests and builds a SAML response [SAML]. The class can be changed as long as the class implements <code>ClientSamlHandler</code> interface. See 3.1 for further information. |
| Property | remotePdpClient Sets a <code>RemotePdpClient</code> implementation. |

[springWsClient](#) bean description

| | |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Description: | The Spring [SpringWS] Web Service client is responsible to send and receive the SOAP messages as well as to marshal and unmarshal the SOAP Body content. |
| Properties | marshaller Sets the marshaller for the XML (required) unmarshaller Sets the unmarshaller for the XML. (required) defaultUri Sets the defaultUri for the Web Service. (required) ActionUri Sets the operation of the specific Web Service. (required) messageSenders Sets the MessageSender which implements the standard-conformance for XACML over SAML over SOAP [SOAP]. (required) interceptors Sets the an interceptor to add the optional Soap Action for SAML. (required to be standard-compliant) |

[marshaller](#) bean description

Description: Marshaller and unmarshaller are customized Jaxb2Marshaller. This marshaller makes sure that namespaces are being inserted correctly.

The class must not be changed.

Property

[contextPath](#)

This property is mandatory and must not be changed. It tells the marshaller where to find its JAXB classes.

[soapSamlHttpClientParams](#) bean description

Description: The bean represents a collection of HTTP protocol parameters applicable to instances of `HttpClient`. The parameters are defined by the SAML Bindings 2.0 specification for the use of SOAP over HTTP [SAMLBind].

The class must not be changed.

3 Reference and Configuration

Overview This part details the various components that comprise HERAS^{AF} PEP Web Service module.

This includes a chapter that describes SAML handling and a chapter on using WS-Security provided by Spring Web Service.

The logical components of the HERAS^{AF} PEP Web Service module are outlined in Figure 3.

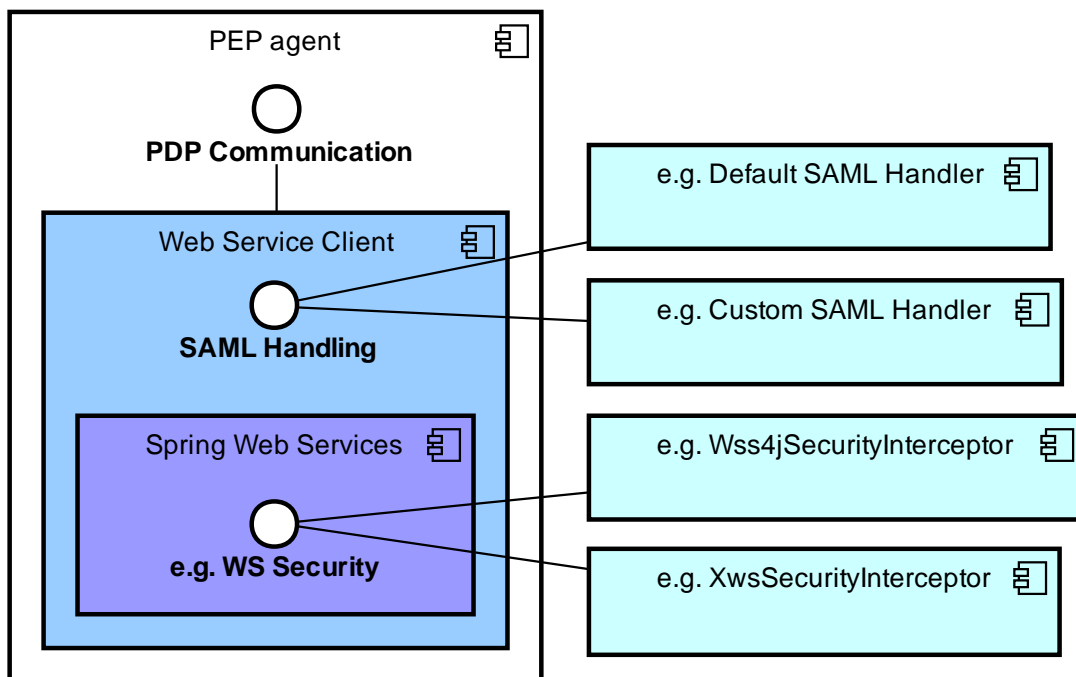


Figure 5: Logical components of the HERAS^{AF} PEP Web Service module

3.1 SAML Handling

Overview SAML handling is an adaption point of the PEP Context Web Service module. It allows adding custom SAML handling classes, which for example allow handling digital signatures inside SAML.

For this release exists a default implementation of a `ClientSamlHandler` only. This implementation does not handle any custom SAML fields.

Wiring Example 9 shows how to wire a `DefaultClientSamlHandler` into an Endpoint.

```
<bean id="clientSamlHandler"
      class="org.herasaf.pep.integration.ws.saml.DefaultClientSamlHandler">
```

```
<property name="remotePdpClient"  
ref="springWsClient" />  
</bean>
```

Example 9: How to wire the DefaultClientSamlHandler

3.2 Spring Web Service

*Additional
Features*

See the previous chapter 2 part 3.2 for information.

Appendix A: General

1 Glossary

| | |
|-------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Action</i> | An operation on a resource |
| <i>Attribute</i> | Characteristic of a subject, resource, action or environment that may be referenced in a predicate or target. |
| <i>Authorization decision</i> | Returned by the PDP to the PEP as a result of evaluating a decision request. |
| <i>Bag</i> | An unordered collection of values, in which there may be duplicate values |
| <i>Decision</i> | The result of evaluating a rule, policy or policy set |
| <i>Decision context</i> | Context containing request information required to create a decision request. |
| <i>Decision request</i> | The request from a PEP to a PDP to render an authorization decision. |
| <i>Effect</i> | The intended consequence of a satisfied rule (either "Permit" or "Deny") |
| <i>Environment</i> | The set of attributes that are relevant to an authorization decision and are independent of a particular subject, resource or action. |
| HERAS ^{AF} | Holistic Enterprise Ready Application Security <small>Architecture Framework</small> |
| <i>Mock object</i> | Mock objects are simulated objects that mimic the behavior of real objects in controlled ways |
| <i>OASIS</i> | Organization for the Advancement of Structured Information Standards |
| <i>Obligation</i> | An operation specified in a policy or policy set that should be performed by the PEP in conjunction with the enforcement of an authorization decision |
| <i>PAP</i> | Policy Administration Point |
| <i>PDP</i> | Policy Decision Point |
| <i>PEP</i> | Policy Enforcement Point |
| <i>PIP</i> | Policy Information Point |

| | |
|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Pluggability</i> | Applications which were designed to provide default functionality that should be suitable for most any application. However in order to provide complete flexibility pluggability applications allow its foundational components to be over-written with custom implementations. |
| <i>Policy</i> | A set of rules, an identifier for the rule-combining algorithm and (optionally) a set of obligations. May be a component of a policy set |
| <i>Prototype</i> | A prototype is often software in a development stage, focusing on a subset of the total requirements for a product. |
| <i>Request context</i> | Context containing a decision request. |
| <i>Request information</i> | Information about the Subjects, Resources, Action and Environment that is used to generate the content of an XACML request. |
| <i>Resolver</i> | An adaption point of the PEP that acquires request information from a data source and parses it into a data structure known by the PEP. |
| <i>Resource</i> | Data, service or system component. |
| <i>Response context</i> | Context containing an authorization decision. |
| <i>SAML</i> | Security Assertion Markup Language |
| <i>SOAP</i> | Simple Object Access Protocol |
| <i>SSL</i> | Secure Socket Layer |
| <i>Subject</i> | An actor whose attributes may be referenced by a predicate. |
| <i>TLS</i> | Transport Layer Security |
| <i>UML</i> | Unified Modeling Language is a standardized visual specification language for object modeling. |
| <i>XACML</i> | eXtensible Access Control Markup Language |
| <i>XML</i> | eXtensible Markup Language |

2 Bibliography

2.1 Specifications and Standards

- [XACML]* OASIS
eXtensible Access Control Markup Language (XACML), Version 2, 5 Jul 2007
<http://www.oasis-open.org/committees/download.php/24548/> (06.06.2008)
- [SAML]* OASIS
[SAMLBind] Security Assertion Markup Language v2.0
<http://docs.oasis-open.org/security/saml/v2.0/saml-2.0-os.zip> (06.06.2008)
- [SAMLXACML]* OASIS
SAML 2.0 Profile of XACML, Version 2, Working Draft 5, 19 July 2007
<http://www.oasis-open.org/committees/download.php/24679> (06.06.2008)
- [SOAP]* W3C
Simple Object Access Protocol v1.1
<http://www.w3.org/TR/2000/NOTE-SOAP-20000508/> (06.06.2008)

2.2 HERAS^{AF} documents

- [RW08PEP]* Daniel Regli, Yannick Winiger:
HERAS^{AF} PEP
Bachelor Thesis
June 2008
Bachelor Thesis at the University of Applied Sciences Rapperswil (HSR)
- [RW08PEPDev]* Daniel Regli, Yannick Winiger:
HERAS^{AF} PEP
Developer's Guide
June 2008
Bachelor Thesis at the University of Applied Sciences Rapperswil (HSR)
- [RW08PEPPDP Dev]* Daniel Regli, Yannick Winiger:
HERAS^{AF} PEP-PDP Communication
Developer's Guide
June 2008

Bachelor Thesis at the University of Applied Sciences Rapperswil (HSR)

[RW08PEPUsr] Daniel Regli, Yannick Winiger:
HERAS^{AF} PEP
User's Guide
June 2008
Bachelor Thesis at the University of Applied Sciences Rapperswil (HSR)

*[RW08PEPPDP
Usr]* Daniel Regli, Yannick Winiger:
HERAS^{AF} PEP-PDP Communication
User's Guide
June 2008
Bachelor Thesis at the University of Applied Sciences Rapperswil (HSR)

[NZ08PAP] Patrick Neyer, Christoph Zellweger:
HERAS^{AF} Policy Deployment Modul
Bachelor Thesis
June 2008
Bachelor Thesis at the University of Applied Sciences Rapperswil (HSR)

[NZ08PAPDev] Patrick Neyer, Christoph Zellweger:
HERAS^{AF} PAP
Developer's Guide
June 2008
Bachelor Thesis at the University of Applied Sciences Rapperswil (HSR)

[NZ08PAPUsr] Patrick Neyer, Christoph Zellweger:
HERAS^{AF} PAP
User's Guide
June 2008
Bachelor Thesis at the University of Applied Sciences Rapperswil (HSR)

[CS07PAP] Massimo Cerqui, Sandro Strebel:
HERAS^{AF}: Policy Administration Point
November 2007
Diploma Thesis at the University of Applied Sciences Rapperswil (HSR)

[CS07PEP] Massimo Cerqui, Sandro Strebel:
HERAS^{AF}: Interzeptoren für Spring AOP und AspectJ
July 2007

Diploma Thesis at the University of Applied Sciences Rapperswil (HSR)

[DOH07DA] Sacha Dolski, Stefan Oberholzer, Florian Huonder:
HERAS^{AF}: XACML 2.0 Implementation
Hauptdokument
November 2007
Diploma Thesis at the University of Applied Sciences Rapperswil (HSR)

[DOH07DADev] Sacha Dolski, Stefan Oberholzer, Florian Huonder:
HERAS^{AF}: XACML 2.0 Implementation
Developer's Guide
November 2007
Diploma Thesis at the University of Applied Sciences Rapperswil (HSR)

[DOH07SA] Sacha Dolski, Stefan Oberholzer, Florian Huonder:
HERAS^{AF}: XACML PDP Web Service Endpoint
July 2007
Diploma Thesis at the University of Applied Sciences Rapperswil (HSR)

[Egg06] René Eggenschwiler:
HERAS^{AF} – Holistic Enterprise-Ready Application Security Architecture
Framework
Manageable policy-based access control for J2EE.
Mai 2006
Diploma Thesis at the University of Applied Sciences Rapperswil (HSR)

[Graf06] Yan Graf:
Distributed Access Control Policies – Enterprise Ready
December 2006
Diploma Thesis at the University of Applied Sciences Rapperswil (HSR)

2.3 Other Resources

[ApacheTomcat] <http://tomcat.apache.org/index.html>

[Spring] <http://www.springframework.org/>

[SpringWS] <http://static.springframework.org/spring-ws/site/>



[WSDL]

<http://www.w3.org/TR/wsdl20/>