Document Id: HMAT-ADD-1300-INT
Issue: *1-11/02/2009*
Revision: *0-11/02/2009*

Heterogeneous Mission Accessibility Testbed HMAT

Toolbox Software Security Layer

Architectural design Document

Authors:
S. Puri
11/02/09

Reviewed by:
P. Nencioni
11/02/09

Approved by:
S. Gianfranceschi
11/02/09

Document change record

Issue	Issue date	Pages/section effected	Reason for change
1.0	11/02/2009	All	Draft version

Distribution List

Company	Name	Function	N° of copies

Table of Content

1. Introduction

1.1. Purpose

This document is the Architectural Design Document (ADD) for HMAT Toolbox Software Security Layer.
The Toolbox Software Security Layer is a configurable application that helps to secure Toolbox and Web services.

1.2. Scope

1.3. Glossary

1.3.1. Abbreviations

Acronym	Extended Form
API	Application Programming Interface
EOLI	Earthnet On-Line Interactive
FTP	File Transfer Protocol
GIS	Geographic Information System
HMA	Heterogeneous Mission Accessibility
HMAT	Heterogeneous Mission Accessibility Testbed
HTTP	Hypertext Transfer Protocol
ICD	Interface Control Document
JDBC	Java Database Connectivity
OGC	Open GIS Consortium
PDP	Policy Decision Point
PEP	Policy Enforcement Point
SAML	Security Assertion Markup Language
SOAP	Simple Object Access Protocol
SP	Service Provider
SSE	Service Support Environment
RE	Runtime Environment
UDDI	Universal Description Discovery and Integration
WSDL	Web Services Description Language
XACML	Extensible Access Control Markup Language
XML	eXtensible Mark-up Language

1.3.2. Definition of Terms

- **Back-end service**: it is the service offered by the SP and deployed on the Toolbox. The back-end service communicates with the SSE via the Toolbox.
- **SOAP**: it is a simple XML based protocol to let applications exchange information over HTTP.
- **SOAP fault**: within the SOAP protocol, the SOAP Fault is an element of the SOAP messages used to carry error and/or status information within a SOAP message.
- **Web Service**: the term Web services describes a standardized way of integrating Web-based applications using the XML, SOAP, WSDL and UDDI open standards over an Internet protocol backbone. XML is used to tag the data, SOAP is used to

transfer the data, WSDL is used for describing the services available and UDDI is used for listing what services are available.

- **WS-Security:** it is a communications protocol providing a means for applying security to Web services
- **XML schema**: it is the description of a type of XML document.

1.4. References

1.4.1. Normative References

In case of conflict between two or more applicable documents, the higher document will prevail.

[NR1] User Management Interface for Earth Observation Services, ref 07-1-18r1, version 0.0.2, 2008-04-23.

[NR2] Ergo Architectural Design Document, ERG-ADD-3100-INT_1.1, 16/01/2009.

[NR3] Heterogeneous Mission Accessibility Testbed HMAT – Toolbox Software Requirement Document (Security Layer), HMAT-SRD-1200-INT-1.1

1.4.2. Informative references

The following documents, although not a part of this test procedure, amplify or clarify its contents.

[IR1] WS-Security, SOAP Message Security V1.1, http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

[IR2] SAML, Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1 http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1

[IR3] Web Services Security SAML Token Profile 1.1 http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf

[IR4] eXtensible Access Control Markup Language (XACML) Version 2.0, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

[IR5] Rampart website: http://ws.apache.org/rampart/index.html

[IR6] Axis2 website: http://ws.apache.org/axis2/

[IR7] WS-Security Policy 1.2, http://docs.oasis-open.org/ws-sx/ws-securitypolicy/v1.2/ws-securitypolicy.html

[IR8] WS-Addressing website: http://www.w3.org/Submission/ws-addressing.

1.5. Document Overview

The document is organized as follows:

- Section 1 gives an overall introduction,
- Section 2 briefly introduces the system context and design and discuss the background to the project,
- Section 3 describes the software Top-Level Architectural Design. The top-level structure of the software item design is described, identifying the software components, their relationships any dependency and interfaces between them,
- Section 4 provides the software requirement traceability matrix.

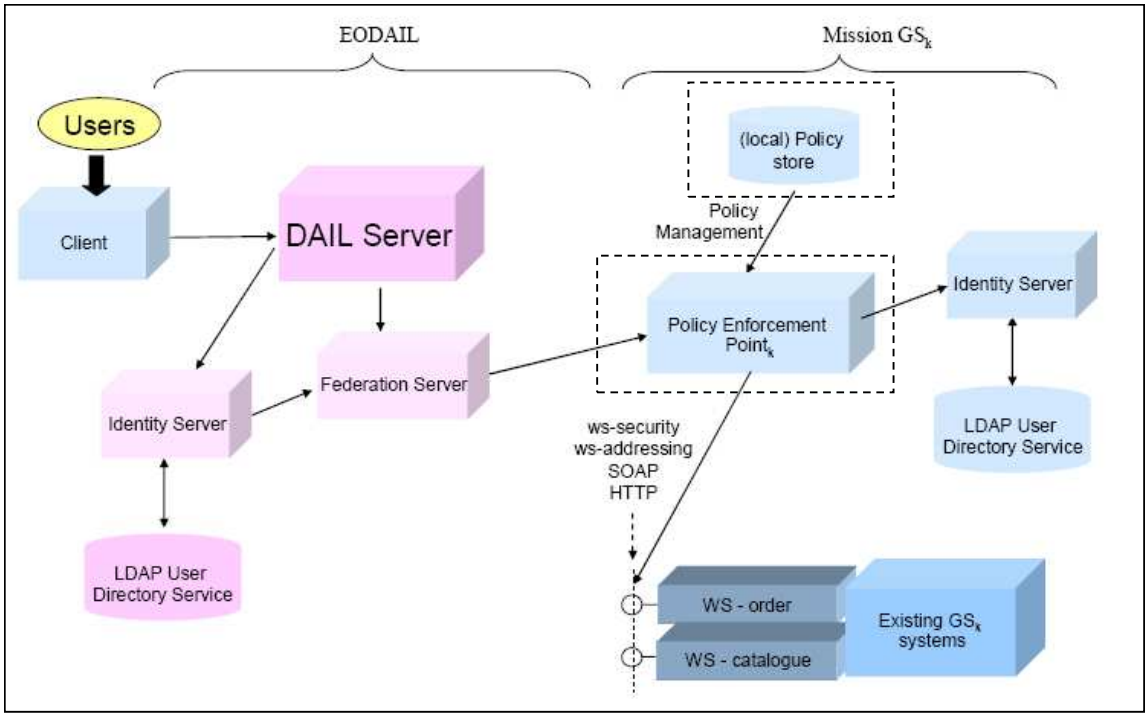
2. SOFTWARE DESIGN OVERVIEW

In accordance with [NR1] a Service Provider (Ground segment) components receiving Web service requests should be able to identify who issued the request and react accordingly; so in [NR1] the following approach is proposed:

1) An authentication Web service (accepting a user identifier password and optionally his identity provider) returns a SAML token [IR2] which authenticates the user to the client (i.e. Web service consumer). This authentication web service may federate the identity within the circle of trust but for the interface context this is irrelevant as the federated identity request would be identical to the initial request.
2) Each subsequent service request by the client (Web service consumer) is to include the SAML token in the SOAP header in the way described below.
3) Each service provider accepts service requests only via a policy enforcement point (PEP). The PEP decides based on the content of the message body, the contents of the message header (including authentication token) and the context (i.e. applicable policies) whether to accept or to refuse the service request or reroute it.

The Toolbox Software Security Layer (hereafter Toolbox Security) implements the item 3) of the proposed approach covering the requirements expressed in [NR3];

Next figure, originally taken from [NR1], illustrates how the software parts implemented in this project fit into the HMA infrastructure.



2.1. Software static architecture

This section provides an overview of the Toolbox Security software static architecture and its integration with the Toolbox software.

Toolbox Security software implements a security layer for the Toolbox software, allowing to specify and check that only *authenticated* and *authorized* clients can request a given service deployed on the Toolbox. Authentication is supported by integrating WS-Security [IR1] element into SOAP request, in particular by using SAML tokens [IR3]. Authorization is enforced by defining policies expressed with XACML language [IR4]; latter policies are also referred as *enterprise level policies*.

At high level the Toolbox is basically composed by two layers: the SOAP layer and the Application layer.
The SOAP layer is in charge of handling all the SOAP issues. It receives the SOAP message, extracts the request, validates it and stores all the information needed to handle the request (e.g. WS-Addressing [IR8] for the asynchronous requests).
The Application layer is in charge of managing the Service, connecting with the Service resource, invoking the operation scripts and creating the response message which is then sent back to the SOAP layer.
Different kind of services can be deployed on the Toolbox. Each service is identified by a Schema defining the file structure and content and some additional resources used internally by the Toolbox.

Toolbox Security layer acts on Toolbox SOAP and Application layer; in particular is put on top of the SOAP layer to provide Web Service security, while is put on top of the Application layer to provide enterprise level security. The new layer allows two different methods for providing a security layer for Web Services: the standalone and the gateway approaches. Figure 1 illustrates the approach.

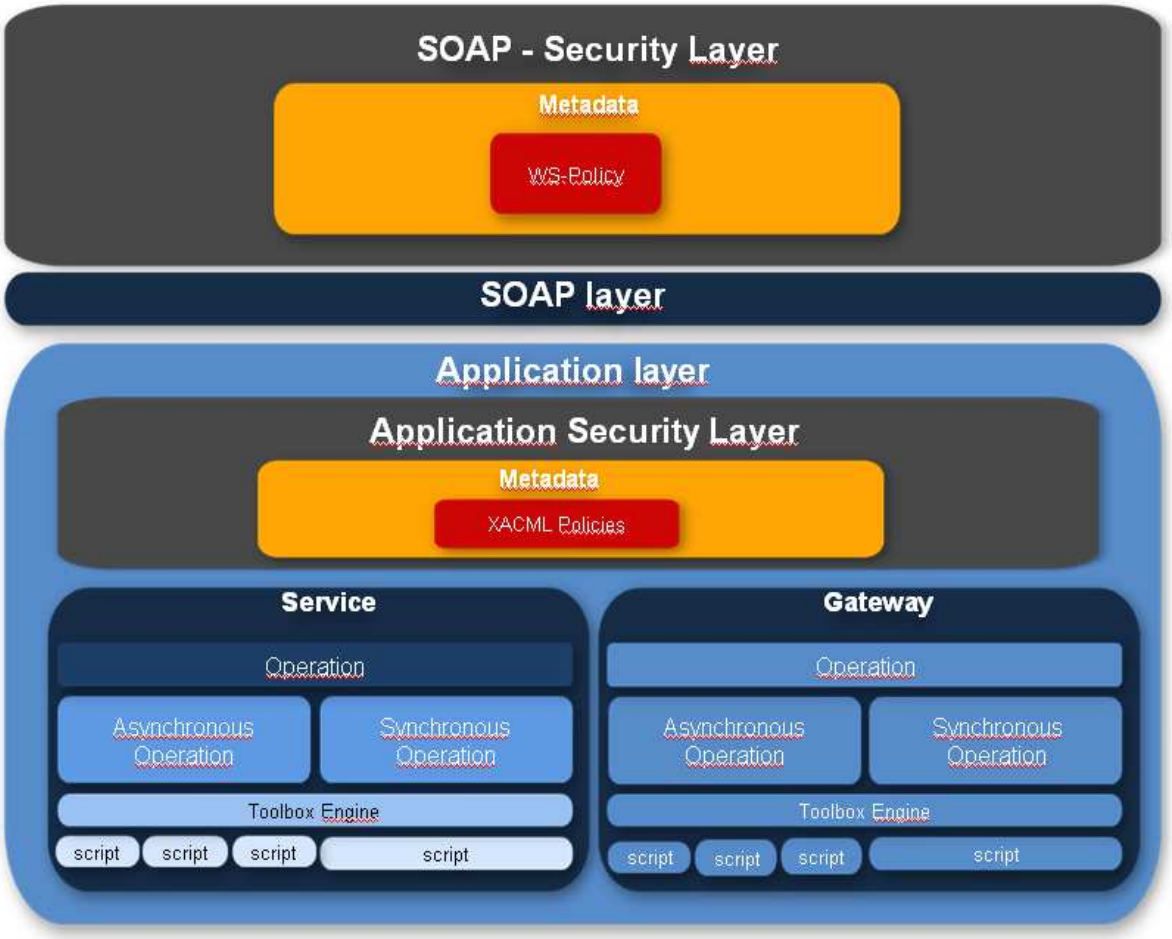


Figure 1 Toolbox Security Stack Layer

In the next sections an overview of the Toolbox software is given, then the standalone and the gateway approaches are discussed.

2.1.1. The Toolbox Software

This section provides an overview of the Toolbox run-time environment (Toolbox RE) which is the environment wrapped by the Toolbox Security layer.
The SSE allows a service provider to go to the SSE Portal and enter the details of its service interactively into the SSE database. This information, which includes the filename of the WSDL file and the parameters to define a service request, is sufficient for the SSE system and the embedded workflow tool to connect to that service over the Internet, without any development on the SSE Portal.

In case a service doesn't support the SOAP communication mode, the Service Provider can install the Toolbox run-time environment and use it as connection layer between the SSE and the service itself.
On the back side, the Toolbox supports several kind of modes to communicate with back-end services:

- File exchange
- FTP
- HTTP
- Java API call
- Java code
- Command invocation
- Database using JDBC 2.0
- SOAP

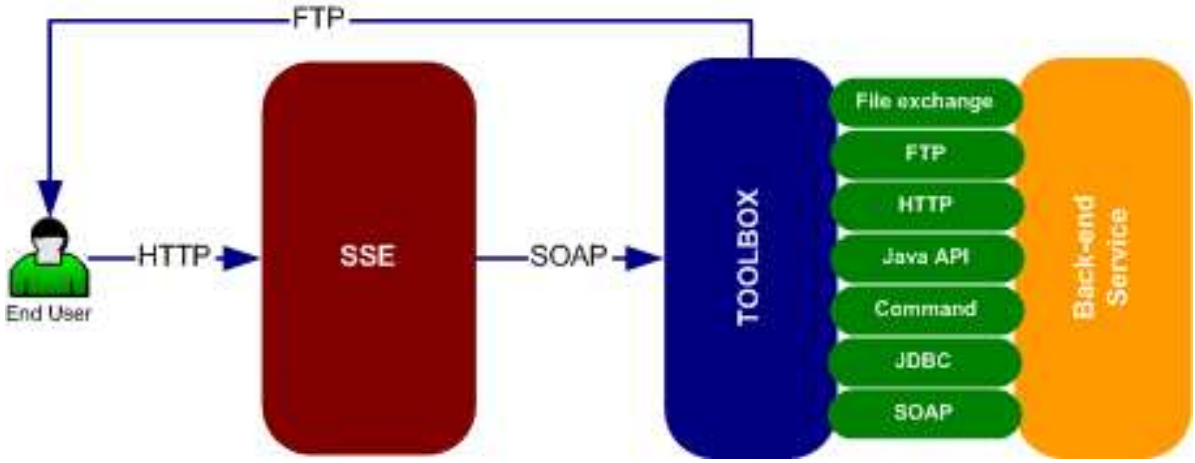


Figure 2 The SSE - TOOLBOX - Back-end service interaction

On the front side, the Toolbox supports both the synchronous as well the asynchronous communication mechanism described in the SSE ICD.

In the synchronous mechanism the SSE Portal is a SOAP client and the Toolbox acts only as a SOAP server. It only responds to SOAP requests. The XML response is sent back to the SSE Portal during the same HTTP exchange.

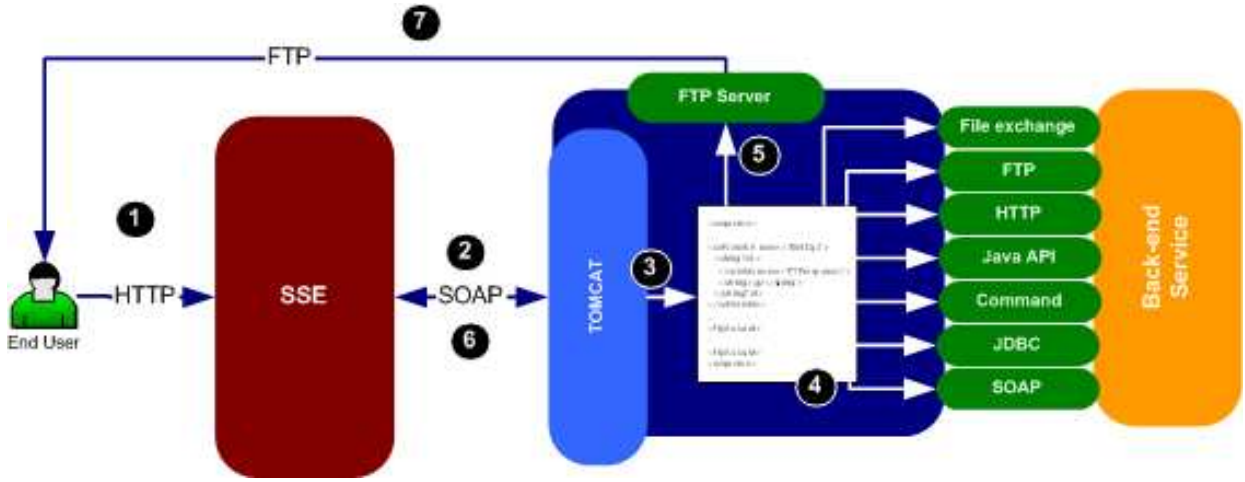


Figure 3 Synchronous Communication

In the asynchronous mechanism the SSE Portal is a SOAP client as well as a SOAP server. The service provider sends a SOAP message back to SSE when a result is ready. Thus the Toolbox implements a SOAP server as well as a SOAP client. The Toolbox uses the WS-Addressing protocol.

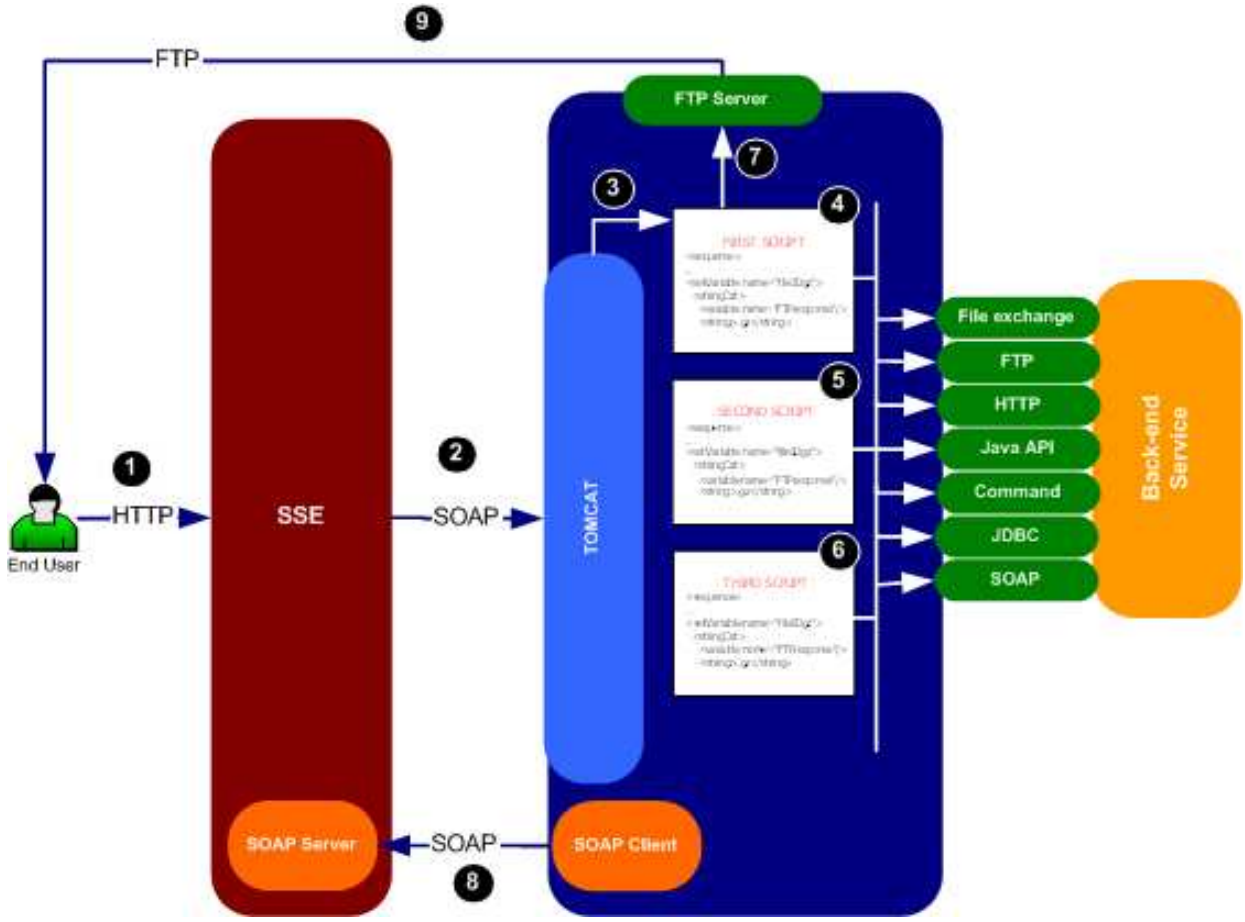


Figure 4 Asynchronous Communication

The Toolbox RE provides a mechanism to define the sequence of steps that are needed to complete the service each time a request arrives from the SSE. The approach used is based on XML scripting. Indeed the services deployed on the Toolbox are defined through a set of XML scripting files interpreted by an XML engine embedded in the Toolbox RE.
A service, to be plugged in the SSE, must implement the operations defined in one of the supported interface (SSE,EOLI,HMA and OGC catalogues). The same service can support more than one operation. For each supported operation the service provider has to define a set of XML scripting files. These files define the steps required to fulfil a request incoming from the SSE Portal.

The Toolbox uses XML Schemas to validate all the input and the output messages exchanged with the SSE Portal. Thus the user shall provide these schemas also and redefine them when necessary (e.g. SSE schemas).

2.1.2. Stand Alone Approach

Stand alone approach can be used to protect Web Services published on the Toolbox.
In this case the security layer is published directly on top of the SOAP interface of the secured service to check the incoming messages and to invoke directly the service operation if all the checks are successful.
This solution is supported by Synchronous and Asynchronous operations.

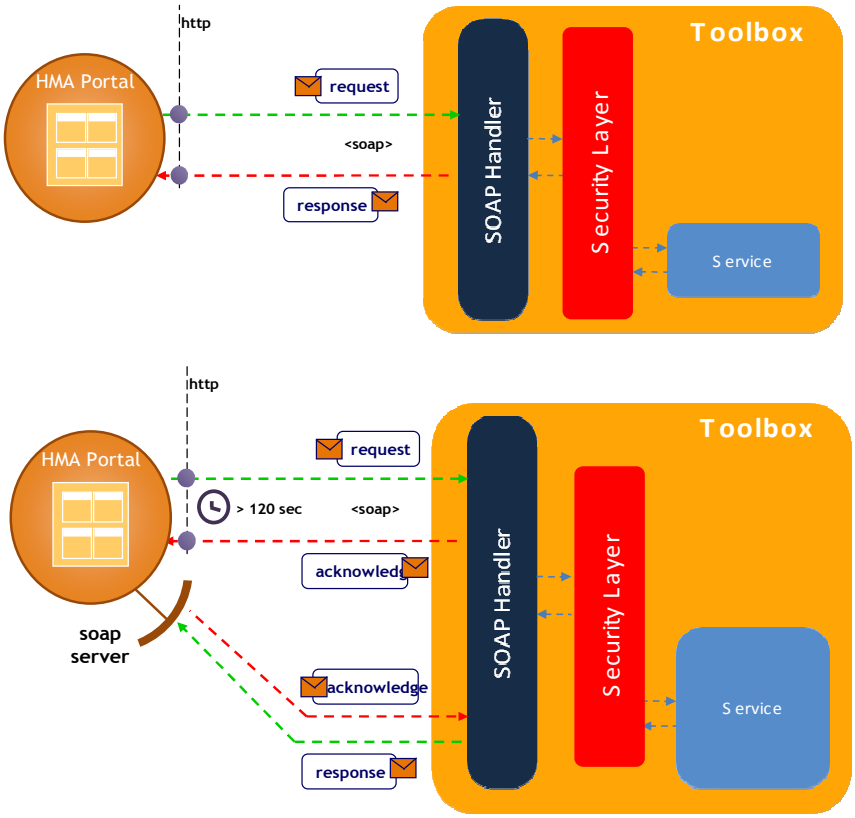


Figure 5 Stand alone approach

2.1.3. Gateway Approach

In the Gateway Approach the Toolbox acts as a gateway to protect end points already published as web services. It functions independently of the Web service it protects and acts as a proxy to the Web service clients. The gateway is the entry point of the system and acts as an enforcement point. All the messages will be directed to the gateway. For each secured service the enforcement point will check the incoming messages and will forward the message to the web service that have to be secured only if all the checks are successful.
The Gateway is basically a service with a predefined behaviour. Thus deploying a gateway is as deploying a specific service having as interfaces the same interfaces deployed on the service to be secured.
It is possible to create one or more services to be secured. Once a gateway service is added to a Toolbox, than it is possible to add security policies to it.
This solution foresee an upgrade of the Toolbox architecture. A new SOAP Server component has to be included in order to be able to secure asynchronous services.

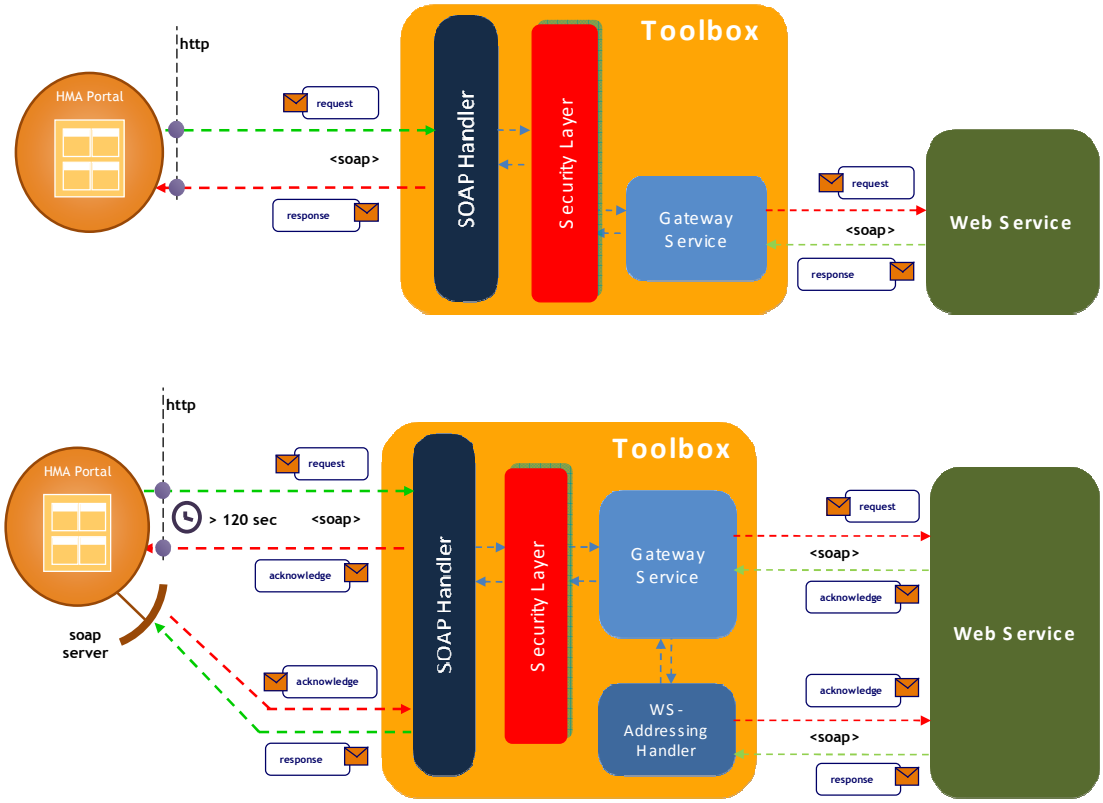


Figure 6 is the UML model of the Toolbox Security context; also the basic use cases are identified.

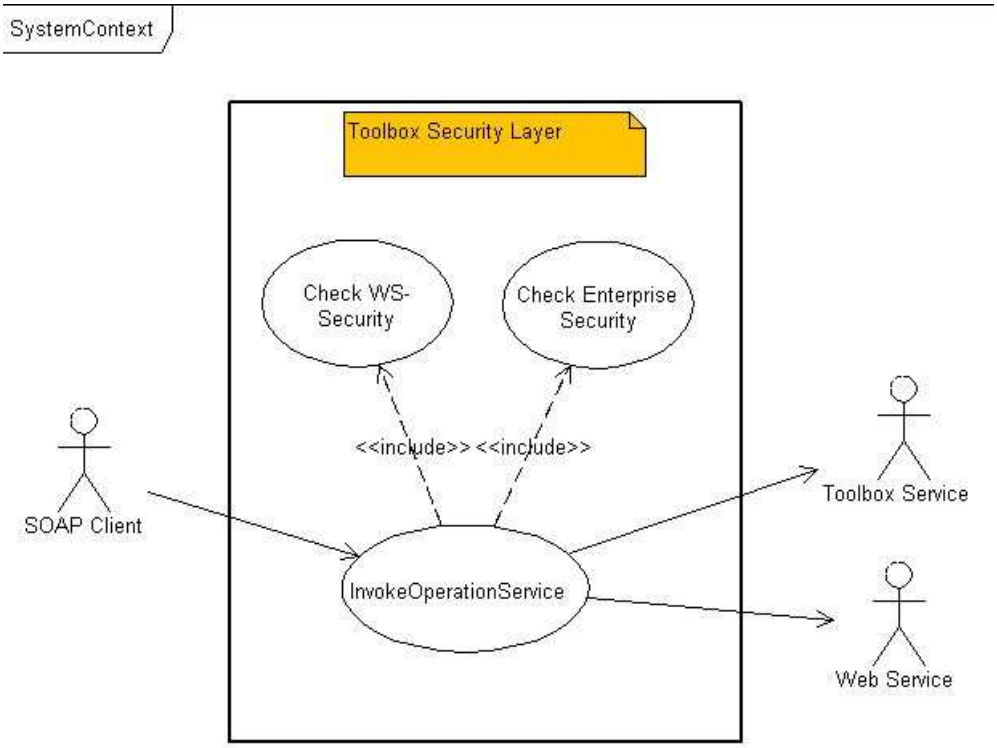


Figure 6 – Toolbox Security Context and Use Cases

The actors of the Toolbox Security are the SOAP client, the Toolbox service (stand alone approach) and the Web Service (gateway approach). The InvokeOperationService use case is initiated by the client which requests an operation of a given service, then checks about WS-Security and enterprise level security are enforced; if these checks succeed than the request is forwarded to the Toolbox or to Web Service, otherwise a fault is returned to the client.

3. Software architectural design

In this chapter the top-level structure of the Toolbox Security software design is described.

3.1. Overall Architecture

In Figure 7 the main components of the Toolbox Security software, the external components together with main relationships are illustrated by using a UML class diagram.

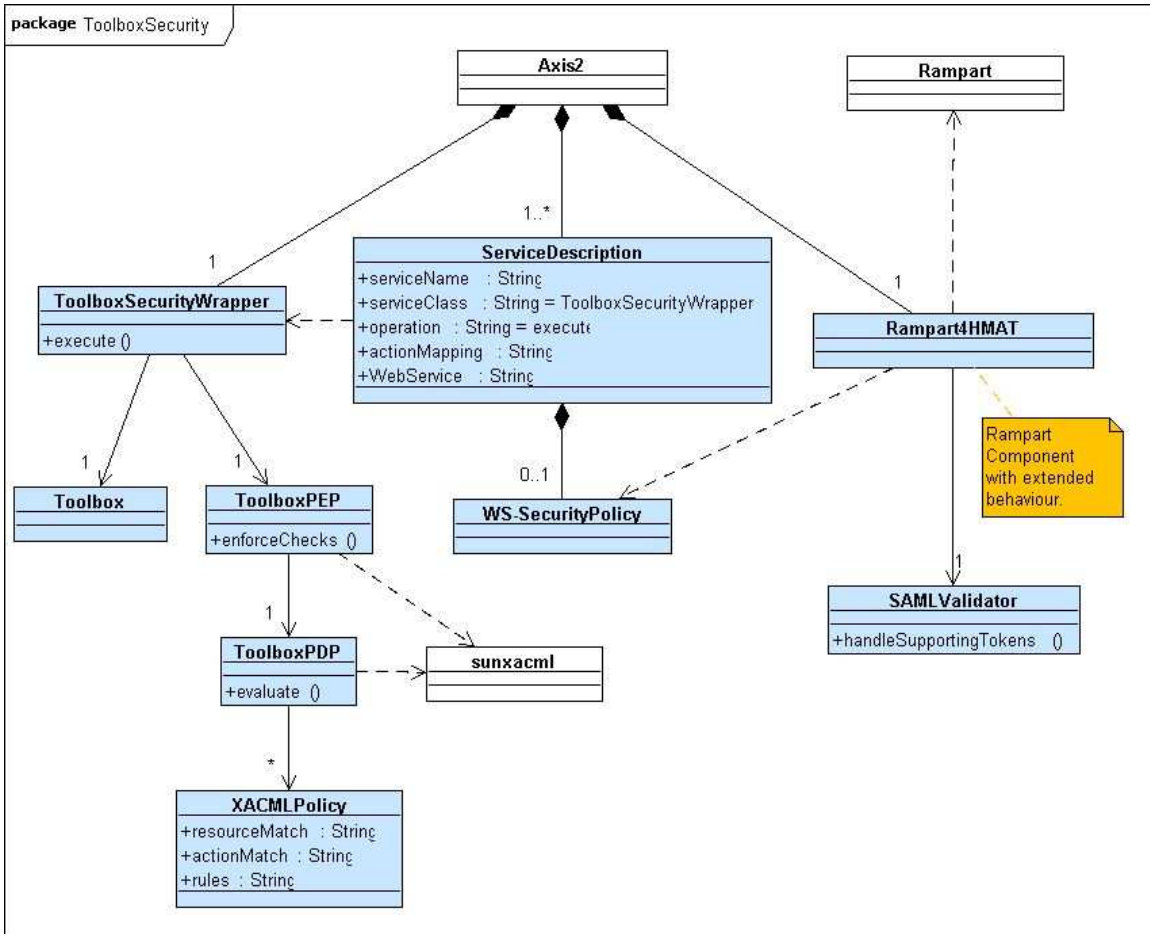


Figure 7 Toolbox Security Static Architecture

In Figure 7, *Rampart*, *Axis2* and *sunxacml* represent external components.

The Toolbox Security software has been integrated into the Axis2 framework component [IR6] which is a basic SOAP engine.

The basic Toolbox Security component which interact with Axis2 are Rampart4HMAT, ToolboxSecurityWrapper and ServiceDescription.

Rampart4HMAT component has been obtained by extending the behavior of the Rampart Axis2 module [IR5]; Rampart is an Axis2 extension implemented by the Apache Software Foundation and gives support for WS-Security policy checks. Rampart4HMAT fixes and extends Rampart behaviour to fulfil the WS-Security requirements specified in [NR3], in particular regarding the authorization security policies specification and validation.
The following table summarizes the support added in Rampart4HMAT, also covering the tracing to the requirements defined in [NR3]:

Extended/Implemented Support	Satisfied Requirements (ID)
SAMLToken WS-Security policy tag element	HMAT-RB-INT-020, HMAT-RB-INT-030
Signed and encrypted SAML token.	HMAT-RB-INT-070
Encryption/signature order of SAML token.	HMAT-RB-INT-090
SOAP fault with SOAP 1.1.	HMAT-RB-INT-010, HMAT-RB-INT-130
Support for SAML token with bearer confirmation method.	TBD

Table 3-1 Rampart4HAMT extended support and requirements

Rampart4HMAT uses SAMLValidator class to validate SAML token.

ToolboxSecurityWrapper is the service exposed to the client which wraps a given Toolbox or Web service; it is used to add authentication and authorization checks to the wrapped service. ToolboxSecurityWrapper has been developed as Axis2 service.

ServiceDescription is used to configure ToolboxSecurityWrapper service into Axis2 and to set the wrapped service; also it allows to attach WS-Security policy to a wrapped service.

XACMLPolicy, ToolboxPEP and ToolboxPDP implement the authentication phase; they use sunxacml, the SUN open source implementation of the OASIS XACML standard..

TBD: deployment diagram.

3.2. Software Item Component

3.2.1. Rampart4HMAT

3.2.1.1. Type

Rampart4HMAT extends the behavior of the Rampart Java component.

3.2.1.2. Purpose

The purpose of this component is to implement WS-Security check requirements upon any SOAP request covering the HMA-T requirements.

3.2.1.3. Function

Rampart4HMAT is an Axis2 module; it is invoked by Axis2 during SOAP request and response management.
The Rampart module introduces a couple of handlers:

- RampartReceiver,
- RampartSender.

The "RampartReceiver" handler intercepts the incoming message. Then Rampart validates the security of the incoming message, and checks whether it is in-line with the specified security policy. All security actions such as decryption of the message, validating the digital signature, validating the timestamp, and authenticating the user happens inside the Rampart module. "RampartSender" is the last handler in the outflow. The outgoing message is intercepted by this handler and Rampart takes the security actions. For example SOAP message can be encrypted, digitally signed, and security tokens are included according to the security policy.

3.2.1.4. Subordinates

Following are the Rampart sub-components which have been extended in Rampart4HMAT to cover the HMAT requirements:

- RampartEngine
- PolicyBasedResultValidator
- SamlTokenBuilder

3.2.2. SAMLValidator

3.2.2.1. Type

SAMLValidator is a new development implemented as Java class.

3.2.2.2. Purpose

This component perform validation about the SAML token.

3.2.2.3. Function

It is called by Rampart4HMAT during the validation of the incoming message. It has a main entry point method called *handleSupportingTokens* which takes in input the token to be validated.

3.2.3. ToolboxSecurityWrapper

3.2.3.1. Type

ToolboxSecurityWrapper is a new development implemented as Java class.

3.2.3.2. Purpose

Given an incoming SOAP request, it allows to execute authentication and authorization checks.

3.2.3.3. Function

ToolboxSecurityWrapper is an Axis2 service which wraps the actual services requested by the client; it is invoked by the client
The ToolboxSecurityWrapper service is deployed on Tomcat and is accessible via HTTP through the URL

https://<tomcat_host>:8080/ToolboxSecurity/services/<service_name>.

Actually ToolboxSecurityWrapper is invoked only if the WS-Security checks performed by Rampart4HMAT are successful.
ToolboxSecurityWrapper invokes the ToolboxPEP to check the XACML policies: only if the checks succeeds the client request is forwarded to the target service, otherwise a SOAP fault with information about the policy failed is returned to the client.

3.2.4. ToolboxPEP

3.2.4.1. Type

ToolboxPEP is a new development implemented as Java class.

3.2.4.2. Purpose

It implements a Policy Enforcement Point as recommended by the XACML specification.

3.2.4.3. Function

It has a main entry point method called *enforceChecks*, which takes in input the requested service, the SOAP action, the SAML token and the body of the SOAP request. It retrieves the available XACML policies and builds the XACML request to be passed to the ToolboxPDP. The XACML request owns all the actual parameters coming from the *enforceChecks* invocation: this allows to use XACML to check about information stored in the SAML token and/or in the SOAP body.

3.2.5. ToolboxPDP

3.2.5.1. Type

ToolboxPDP is a new development implemented as Java class.

3.2.5.2. Purpose

It implements a Policy Decision Point as recommended by the XACML specification.

3.2.5.3. Function

It has a main entry point method called *evaluate* which takes in input the XACML request, evaluates it against the available XACML policies and returns the result.

3.2.6. ServiceDescription

3.2.6.1. Type

ServiceDescription is implemented as XML.
It is stored in:

<TOMCAT_ROOT>/webapps/ToolboxSecurity/WEB-INF/services/ToolboxSecurityWrapper/META-INF/services.xml .

3.2.6.2. Purpose

It stores the Axis2 configuration information for the ToolboxSecurityWrapper and the wrapped services; also it provides a connection between the wrapped services and the WS-Security policy that need to be applied.

3.2.6.3. Subordinate

WS-SecurityPolicy

3.2.6.4. Data

For a given wrapped service the following information is available:

- ServiceName: the name of the Toolbox service that need to be wrapped
- ServiceClass (read only): represents the wrapper service
- Operation (read only): is the default operation of the wrapper service
- ActionMapping: the SOAP action used to invoke the wrapped service
- WS-Policy: the WS-Security policies that need to be applied for the wrapped service; see 3.2.7.
- WebService: the address of the Web Service that need to be protected. It is used to implement the gateway approach.

Next figure shows an XML implementation example of ServiceDescription:

```
<?xml version="1.0" encoding="UTF-8"?>
<serviceGroup>

    <service name="ServiceName" scope="application">
        <module ref="rampart"/>
        <module ref="addressing"/>
        <description>Service description</description>
        <parameter name="ServiceClass">com.intecs.service.ToolboxSecurityWrapper</parameter>
        <operation name="execute">
                <messageReceiver
                    class="org.apache.axis2.receivers.RawXMLINOutMessageReceiver"/>
                <actionMapping>SoapAction</actionMapping>
        </operation>

        <!--  WS-SECURITY POLICIES -->
        <wsp:Policy wsu:Id="SgnOnlyAnonymous">
        </wsp:Policy>

    </service>

</serviceGroup>
```

3.2.7. WS-SecurityPolicy

3.2.7.1. Type

WS-SecurityPolicy is implemented as XML. It is stored in:

<TOMCAT_ROOT>/webapps/ToolboxSecurity/WEBINF/services/ToolboxSecurityWrapper/META-INF/services.xml

inside the ServiceDescription.

3.2.7.2. Purpose

It allows to specify the WS-Security condition for a given wrapped service.

3.2.7.3. Data

The XML must be conformant to the WS-Security Policy standard specification [IR7].
Concerning reusability it is worth to noting that Toolbox Security, by using WS-SecurityPolicy, allows to define different WS-Security policies for the wrapped services.
The following is the WS-Security policy which allows to cover WS-Security requirements expressed in [NR3]:

```
<wsp:Policy wsu:Id="SgnOnlyAnonymous"
    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
    xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:sp="http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702">
    <wsp:All>
        <sp:AsymmetricBinding>
            <wsp:Policy>
                <sp:InitiatorToken>
                    <wsp:Policy>
                        <sp:X509Token sp:IncludeToken=
                            "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
                            <wsp:Policy>
                                <sp:RequireThumbprintReference/>
                                <sp:WssX509V3Token10/>
                            </wsp:Policy>
                        </sp:X509Token>
                    </wsp:Policy>
                </sp:InitiatorToken>
                <sp:RecipientToken>
                    <wsp:Policy>
                        <sp:X509Token sp:IncludeToken=
                            "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/Never">
                            <wsp:Policy>
                                <sp:RequireThumbprintReference/>
                                <sp:WssX509V3Token10/>
                            </wsp:Policy>
                        </sp:X509Token>
                    </wsp:Policy>
                </sp:RecipientToken>
                <sp:AlgorithmSuite>
                    <wsp:Policy>
                        <sp:Basic128/>
                    </wsp:Policy>
                </sp:AlgorithmSuite>
                <sp:Layout>
                    <wsp:Policy>
                        <sp:Strict/>
                    </wsp:Policy>
                </sp:Layout>
                <sp:IncludeTimestamp/>
                <sp:EncryptBeforeSigning/>
            </wsp:Policy>
        </sp:AsymmetricBinding>

        <sp:SignedEncryptedSupportingTokens>
            <wsp:Policy>
                <sp:SamlToken sp:IncludeToken=
                    "http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/IncludeToken/AlwaysToRecipient">
                    <sp:WssSamlV11Token11/>
                </sp:SamlToken>
            </wsp:Policy>
        </sp:SignedEncryptedSupportingTokens>

        <sp:SignedParts>
            <sp:Body/>
        </sp:SignedParts>

    </wsp:All>

</wsp:Policy>
```

Figure 8 WS-Security Policy

3.2.8. XACMLPolicy

3.2.8.1. Type

XACMLPolicy is implemented as XML file.
It is stored in:

<TOMCAT_ROOT>/webapps/ToolboxSecurity/WEB-INF/services/toolbox/policies/.

3.2.8.2. Purpose

It allows to specify an XACML policies for a given service. XACML policies can be built upon SAML token and SOAP body attributes values.

3.2.8.3. Data

The following information can be provided according to the XACML specification [IR7]:

- resourceMatch: identifies the wrapped service for which the policy applies; its value must be */ToolboxSecurity/services/<service_name>*,
- actionMatch: identifies the SOAP operation for which the policy applies,
- rules: identifies the rules that need to be checked.

3.2.9. Toolbox

3.2.9.1. Type

The Toolbox is implemented as Java class.

3.2.9.2. Purpose

The purpose of this class is to receive and handle all requests coming from the Toolbox Security Layer, i.e. authorized requests only. Several kind of requests may be sent to this class, obtaining different kind of behaviours.

3.2.9.3. Function

This class performs essentially the following tasks:

- On initialisation (executed only once before the first request is served) it loads its configuration containing, among other details, the list of the deployed services. Thus the Toolbox will contain a table of Service objects (see [NR2]) loaded during initialization.
- While running, the Toolbox is invoked from the ToolboxSecurityWrapper with the request to be served: the request is redirected to the selected service, i.e. the Service object.

4. Software Requirements Traceability Matrix

This section will be provided in the next version of the document.