HMA-S

Identity Mgt Demonstrator Technical Note

(D2000.3)

Prepared by:
Y. Coene (Spacebel)
M.R. Barone (Intecs)
T. Hoa Nguyen (Spacebel)
25/04/2014

Verified by:
M. Gilles
25/04/2014

Approved by:
Y. Coene
25/04/2014
Abstract

This document is the HMA-S Identity Management Services Demonstrator Technical Note. It contains a description of the interfaces and the user manual.

Keywords

HMA-S, ESRIN, HMA, Earth Observation, Identity Management, Demonstrator, OGC 07-118.

Contract

Contractual: ✓
Contract issuer: ESRIN / ASTRIUM-UK
Contract n°: 284000 (internal)

Classification

General Public: 
Industry Programme: ✓
Restricted Dispatching Programme: 
Confidential Programme: 

Configuration

Configured document: ✓
Non-configured document: 

References

Reference: HMA-S.SPB.D2000.3
Issue: 1 – 13/05/2013
Revision: 2 – 25/04/2014
Number of Pages: 115
### Internal Distribution

<table>
<thead>
<tr>
<th>Name</th>
<th>Dept.</th>
<th>Copies</th>
<th>Information</th>
<th>Action</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y. Coene</td>
<td>Space</td>
<td>1</td>
<td>Information</td>
<td>✓</td>
<td>For preparation</td>
</tr>
<tr>
<td>P. Jacques</td>
<td>Space</td>
<td>1</td>
<td></td>
<td>✓</td>
<td>For verification</td>
</tr>
<tr>
<td>M. Gilles</td>
<td>Space</td>
<td>1</td>
<td></td>
<td>✓</td>
<td>For information</td>
</tr>
<tr>
<td>T. Hoa Nguyen</td>
<td>Space</td>
<td>1</td>
<td></td>
<td>✓</td>
<td>For preparation</td>
</tr>
<tr>
<td>M.R. Barone</td>
<td>Intecs</td>
<td>1</td>
<td></td>
<td>✓</td>
<td>For preparation</td>
</tr>
<tr>
<td>S. Gianfranceschi</td>
<td>Intecs</td>
<td>1</td>
<td></td>
<td>✓</td>
<td>For preparation</td>
</tr>
<tr>
<td>Y. Coene</td>
<td>Space</td>
<td>1</td>
<td></td>
<td>✓</td>
<td>For approval</td>
</tr>
<tr>
<td>V. Demeuse</td>
<td>CM</td>
<td>1</td>
<td></td>
<td>✓</td>
<td>For configuration</td>
</tr>
</tbody>
</table>

### External Distribution

<table>
<thead>
<tr>
<th>Name</th>
<th>Dept.</th>
<th>Copies</th>
<th>Information</th>
<th>Action</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Philippe Mougaud</td>
<td>ESRIN</td>
<td>1</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Claudio Gizzi</td>
<td>ASTRIUM-UK</td>
<td>1</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Pier Giorgio Marchetti</td>
<td>ESRIN</td>
<td>1</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Andrea Della Vechia</td>
<td>ESRIN</td>
<td>1</td>
<td></td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
### Document Change Log

<table>
<thead>
<tr>
<th>Issue</th>
<th>Issue Date</th>
<th>Pages Affected</th>
<th>Relevant Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>13/05/2013</td>
<td>All</td>
<td>Initial draft version derived from the HMA-T Identity and Access Management Guidance Document, HMA-T-TN-0004-SPB Issue 1, Revision 0, 12/07/2010.</td>
</tr>
<tr>
<td>1.1</td>
<td>10/01/2014</td>
<td>All</td>
<td>Document updated according to implementation. Add installation procedure.</td>
</tr>
</tbody>
</table>
| 1.2   | 25/04/2014 | Section 4, B.2 | Use latest version of CIM EP means I15 output of T5 as protected service  
Add reference to the STS source code stored on the Google website. |
TABLE OF CONTENTS

TABLE OF CONTENTS .................................................................................................................. V

LIST OF FIGURES ........................................................................................................................ IX

LIST OF TABLES .......................................................................................................................... IX

HMA-S PROJECT - EXECUTIVE SUMMARY .............................................................................. X

1 INTRODUCTION ...................................................................................................................... 12
  1.1 Purpose of the Document ........................................................................................................ 12
  1.2 Scope of the Document ............................................................................................................ 12
  1.3 Readership of the Document .................................................................................................. 12
  1.4 Organisation of the Document ............................................................................................... 12
  1.5 Applicability of the Document ............................................................................................... 12
  1.6 Abbreviations and Acronyms ................................................................................................. 13
  1.7 Definitions ............................................................................................................................... 14

2 APPLICABLE AND REFERENCE DOCUMENTS ..................................................................... 15
  2.1 Applicable Documents ........................................................................................................... 15
  2.2 Reference Documents ............................................................................................................ 15

3 DEMONSTRATOR ARCHITECTURE ...................................................................................... 17
  3.1 Component Types .................................................................................................................. 17
    3.1.1 Mapping Component Types on Software Packages .......................................................... 18
  3.2 Interfaces ............................................................................................................................... 19
    3.2.1 IF-HMAS-Authenticate ..................................................................................................... 19
    3.2.2 IF-SSO-Authenticate ....................................................................................................... 20
    3.2.3 IF-XXX ............................................................................................................................ 20
  3.3 Persistent HMA-S Testbed ....................................................................................................... 21
    3.3.1 TestBed Software Component Instances ........................................................................ 21
    3.3.2 TestBed Software Configuration Items ............................................................................ 22
4 SOFTWARE INSTALLATION MANUAL................................................................. 32
  4.1 Introduction ............................................................................................ 32
  4.2 Prerequisites ....................................................................................... 32
  4.3 Install the Single VM (from Scratch) ..................................................... 33
      4.3.1 STEP 1: Install Linux................................................................. 33
      4.3.2 STEP 2: Install STS ................................................................. 33
      4.3.3 STEP 3: Install PEP ................................................................. 34
      4.3.4 STEP 4: Install CIM-EP Catalogue Client ......................... 36
      4.3.5 STEP 5: Configuration Customization .................................. 37
  4.4 Install VM1 ......................................................................................... 38
      4.4.1 Deploy OVF Template ............................................................... 38
      4.4.2 Configure Applications ............................................................ 38
  4.5 Install VM2 ......................................................................................... 41
      4.5.1 Deploy OVF Template ............................................................... 41
      4.5.2 Configure Applications ............................................................ 41
  4.6 Install VM3 ......................................................................................... 42
      4.6.1 Deploy OVF Template ............................................................... 42
      4.6.2 Configure Applications ............................................................ 42
      4.6.3 Configure existing CIM-EP Client ......................................... 44
  4.7 Configuration ...................................................................................... 46
      4.7.1 STS / LDAP Configuration ....................................................... 46
      4.7.2 PEP configuration ..................................................................... 47
      4.7.2.1 Setting PEP-Level Keystore ................................................ 48
      4.7.2.2 Setting Service-Level Keystores ...................................... 49
      4.7.3 CIM-EP Client Configuration ................................................ 49
  5 USER MANUAL ....................................................................................... 51
  5.1 Using the Online the Demonstrator via CIM-EP Client .................... 52
  5.2 Using the Online the Demonstrator via SOAP-UI .............................. 57
C.5  Manage a PEP Service ................................................................. 92
    C.5.1  Service Information ......................................................... 93
    C.5.2  Service Log ................................................................. 94
    C.5.3  Configure Service .......................................................... 94
    C.5.4  Service Instances ......................................................... 94
C.6  Change PEP Admin Password ..................................................... 95

ANNEX D: USER GUIDE – CIM EP CATALOGUE CLIENT ................................. 96
    D.1  Introduction .................................................................. 96
    D.2  Prerequisites ............................................................... 96
    D.3  Installation ................................................................ 96
    D.4  Installation Testing ...................................................... 96
    D.5  Configuration ............................................................. 97
        D.5.1  Catalogue Registration ........................................... 97
        D.5.2  SSO IDP Registration ............................................. 98
        D.5.3  Keystore Configuration .......................................... 99

ANNEX E: APACHE ANT SCRIPT TO CREATE KEYSTORES .......................... 101

ANNEX F: HMA ONLINE RESOURCES ..................................................... 114
**List of Figures**

Figure 1: Demonstrator Component Types and Interfaces................................................................. 17
Figure 2: Configuration where STS is IdP (Part 1).............................................................................. 25
Figure 3: Configuration where STS is IdP (Part 2).............................................................................. 26
Figure 4: Configuration where STS is not IdP (part 1)...................................................................... 27
Figure 5: Configuration with Multiplicity (part 1).............................................................................. 29
Figure 6: Configuration with Multiplicity (Part 2).............................................................................. 30
Figure 7 PEP Configuration - Java security ...................................................................................... 83
Figure 8 PEP Configuration - Launch............................................................................................... 84
Figure 9 PEP Configuration - Network............................................................................................ 85
Figure 10 PEP Configuration - Logging............................................................................................ 86
Figure 11 PEP Configuration - Application level keystore ................................................................ 87
Figure 12 PEP Configuration - Contact info...................................................................................... 88
Figure 13 PEP Service Creation - STEP 1......................................................................................... 89
Figure 14 PEP Service Creation - STEP 2......................................................................................... 90
Figure 15 PEP Service Creation - STEP 3 (optional). ....................................................................... 91
Figure 16 PEP Service Creation - STEP 4 (optional)........................................................................ 92
Figure 17 PEP Service Management............................................................................................... 93
Figure 18 PEP Service Information.................................................................................................. 94
Figure 19 PEP Service Instances...................................................................................................... 95
Figure 20: Catalogue Client home page – Demonstrator Client ......................................................... 97
Figure 21 CIM EP Client Setting Form page........................................................................................ 98

**List of Tables**

Table 1: Software Component Instance (Service)................................................................................. 22
Table 2: TestBed Software Configuration Items.................................................................................. 23
Table 3: Virtual Machines.................................................................................................................. 24
Table 4: Online Demonstrator Access Endpoints............................................................................... 51
Table 5: SAML Attributes value of Test Users applicable in the Demonstrator.................................. 52
Table 6: Test Input Data...................................................................................................................... 56
Table 7: Open source download URLs.............................................................................................. 114
Table 8: Online software..................................................................................................................... 114
HMA-S Project - Executive Summary

The state of play of the standardization of the ground segment interfaces for EO missions is described in [RD11] and [RD12]. The figure below, taken from [RD11] highlights the existing space standards, mainly from the European Cooperation for Space Standardisation (ECSS), covering the overall earth observation process, and the interfaces where the harmonization work within the HMA projects is focused.

Figure 1: Earth Observation Ground Segment Components [RD11]

The objective of “Heterogeneous Missions Accessibility” – HMA” is to establish harmonised access to heterogeneous earth observation (EO) missions’ data from multiple mission ground segments, including national missions and ESA Sentinel missions. In practice, the goal of HMA is to standardise the ground segment interfaces of the satellite missions for easier access to EO data. The HMA Architecture Working Group (AWG) has been coordinating the ground segment interface harmonisation activities initiated by the Ground Segment Coordination Body (GSCB). These activities, which were performed under ESA contracts such as HMA-I, HMA-T and most recently HMA-Follow On (HMA-FO) have produced interface specifications standardised through the OGC Consortium. The so-called “HMA Cookbook” [RD12] describes in detail the domains which have been subject to standardisation.

The ground segment interfaces covered by HMA-related projects and in various stages of standardisation at the Open Geospatial Consortium (OGC) include:

- Dataset (i.e. Product) metadata: OGC 06-080 and OGC 10-157,
- Catalogue access (datasets): OGC 06-131, OGC 10-189,
The HMA-S project is the continuation of the standardisation activities from HMA-T and HMA-FO. The HMA-S project aims to further advance the HMA standardisation activities and address in particular the following interfaces:

- Dataset metadata (Task 3),
- Catalogue access (datasets, dataset series and services) (Task 4 and 5),
- Feasibility analysis (Task 7),
- Ordering and product download (Task 4),
- Processing (Task 6),
- Identity Management (Task 2) [AD05].

The objectives of the HMA-S demonstrators are on one hand to re-use existing applications to optimise the effort and maintain a set of open-source reference implementations within the scope of HMA. On the other hand, the demonstrators need to be made available as a standalone version for independent download and use; but also integrated into the HMA-S Test Bed for online access.

In response to these two challenges, Astrium and Spacebel are proposing demonstrators relying on previous implementation and relevant to the ongoing projects or specification to implement.

The Identity Management, EO metadata and ebRIM CIM demonstrators will be based on previous developments from HMA-T for Identity Management and ERGO (Buddata) and SMAAD for the EO metadata and the CIM EP Demonstrators.

The OpenSearch extensions and Web Processing Service demonstrator will reuse MapShup and the DREAM Multi-Mission Feasibility Analysis components.
1 INTRODUCTION

1.1 Purpose of the Document

This document is the “Identity Management Demonstrator Services” Technical Note. It is prepared by Spacebel with support from Intecs as a deliverable of WP2300 of the HMA for Science (HMA-S) project. It is identified as HMA-S.SPB.D2000.3 and provides both a description of the native interfaces and the user manual for the demonstrator.

The main purpose of the present document is to provide guidance for understanding, deploying and configuring the Identity Management Services Demonstrator in the context of HMA-S. It provides references and links to documents and software packages.

1.2 Scope of the Document

This document is deliverable HMA-S.SPB.D2000.3 as identified in WP2300. It satisfies Task 2 I20.3.1 and I20.3.2 requirements in the Statement of Work (SOW) [AD03].

The present document provides the main concepts of the Identity Management Services Demonstrator of HMA-S; it explains the role of the different components as well as the ways they can be deployed and configured. The present document does not neither cover the interfaces nor the architecture of these components.

1.3 Readership of the Document

This document is intended to be read by the HMA-S project team and the ESA Technical Officer. The target audience also includes software architects, system integrators and system administrators involved.

1.4 Organisation of the Document

This document is organised as follows:
- Chapter 1 is the introduction to this document.
- Chapter 2 lists the applicable and reference documents.
- Chapter 3 describes the high level architecture of the Demonstrator and its main interfaces.
- Chapter 4 contains the installation instructions for the Demonstrator.
- Chapter 5 is the user manual for the online version of the Demonstrator.

1.5 Applicability of the Document

This document applies to Task 2 of the HMA-S project.
1.6 Abbreviations and Acronyms

- **ATS**: Abstract test Suite
- **CITE**: Compliance and Interoperability Testing and Evaluation
- **CTL**: Conformance Test Language
- **DAIL**: Data Access Integration Layer
- **EO**: Earth Observation
- **ESA**: European Space Agency
- **ETS**: Executable Test Suite
- **HMA**: Heterogeneous Missions Accessibility
- **HMA-FO**: HMA Follow-On
- **HMA-I**: HMA-Interoperability
- **HMA-S**: HMA for Science
- **HTTP**: HyperText Transport Protocol
- **HTTPS**: HTTP Secure
- **ICD**: Interface Control Document
- **IdP**: Identity Provider
- **PEP**: Policy Enforcement Point
- **PDP**: Policy Decision Point
- **RTS**: Request Security Token
- **RTSR**: Request Security Token Response
- **SAML**: Security Assertion Mark up Language
- **SP**: Service Provider
- **SSO**: Single Sign-On
- **STS**: Security Token Service
- **UM**: User Management
- **UM-SSO**: User Management – Single Sign-On
- **VM**: Virtual Machine
- **XACML**: eXtensible Access Control Markup Language
- **XML**: Extensible Markup Language
### 1.7 Definitions

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAML token</td>
<td>SAML assertions defined in the OGC 07-118 that are included in service SOAP request by the web service clients when accessing the protected Web services.</td>
</tr>
<tr>
<td>STS</td>
<td>A Security Token Service (STS) is primarily designed to issue tokens. It can be used to either convert a token from a certain format into a different format, or to convert tokens from one security domain into tokens of another security domain (source: [RD2]).</td>
</tr>
</tbody>
</table>
2 APPLICABLE AND REFERENCE DOCUMENTS

2.1 Applicable Documents

The following documents are applicable to the project. In the current document, these documents are referenced as listed below.

[AD01] HMA-S Project Management Plan, HMA-S.ASU.D100.1, Issue 1, Revision 0, 25/02/2013.
[AD03] Statement of Work LTDP-GSEG-EOPG-SW-12-0007, Heterogeneous Missions Accessibility for Science, Issue 1, Revision 0, 13/09/2012.
[AD05] HMA-SPB.D2000.1: OGC User Management Interfaces for EO Services. This document corresponds to the OGC 07-118r9, 28/06/2013 document modified as part of the WP 2100 of the HMA-S project.

2.2 Reference Documents

The following documents provide background reference. In the body of the text these documents are referenced as listed below.

[RD04] HMA-S.CTR.D5000.1: This document corresponds to the updated OGC 07-038r3 document modified as part of the WP 5100 of the HMA-S project to become I15 (ISO19115 Metadata) Extension Package of CS-W ebRIM Profile, OGC 13-084r1, 11/10/2013.
[RD07] ETS for OGC 07-118 http://portal.opengeospatial.org/?m=projects&a=view&project_id=309


3 DEMONSTRATOR ARCHITECTURE

3.1 Component Types

The figure below shows the Component Types and their interfaces as used in the Identity Management Services Demonstrator of HMA-S. The instantiation of these components will be explained further in chapter 3.4 of this document. The following conventions are used in the diagram below:

- Components E_y: identify external components not part of the downloadable HMA-S software.
- Components H_x.y: identify HMA-S Demonstrator components which are the output of HMA-S Task x.

![Diagram of Demonstrator Component Types and Interfaces](image)

The Identity Management Services Demonstrator of HMA-S complies with OGC 07-118 User Management Interfaces for Earth Observation Services, [AD05]. It is based on the following components (what follows is intentionally simplified in order to provide a quick overview for installation/administration of the system, detailed description may be found in [AD05]):

1. Security Token Service (H2.2): assesses the identity of requester and delivers SAML token for authenticated user; it serves Security Token Request (RST) and delivers signed and encrypted SAML token.

2. Policy Enforcement Point (H2.3): checks that a given service request issued by (or on behalf of) a given user is authorised for that user, according to defined access rules; this check is based on
the SAML token delivered by STS, which is attached in the service request; if authorised, then the PEP relays the request to the end-point SP.

To get an integrated system, two more components are needed:

3. **Client (H2.1):** allows a user to authenticate and to receive a SAML token by issuing a RST to STS; allows issuing a service request to service protected by a PEP, by attaching the SAML token in the request. The Demonstrator uses the CIM EP catalogue client output of HMA-S task 5 and common with task 3. This component is deployed at Spacebel and ESA (URLs can be found in in Table 8 of Annex F).

The OGC 07-118 document [AD05], which is the foundation of Identity Management Services Demonstrator, strongly relies on asymmetrical key encryption and signature. Several keys have to be exchanged and configured in the different components in order to set-up the system. This configuration depends on the role of the STS:

- if the STS is the identity provider (IdP), then the RST shall contain username and password, so that STS can authenticate user from its local user registry; this is the **RST with password case** (see [AD05]).

- if the STS is not the IdP, then the client relies on another IdP, the RST (issued after authentication on the other IdP) shall contain the username and a signature made by the client; from that request, that STS can authenticate the client as a trusted client; this is the **RST with signature case** (see [AD05]).

4. **Service (H2.4):** this is an example of a protected service (SOAP 1.1 assumed although the specification [AD05] does not have this limitation). This service is assumed to be the service provided by HMA-S task 5 [RD08] and is assumed to be already deployed and accessible.

### 3.1.1 Mapping Component Types on Software Packages

In the context of HMA-S, the software packages can be found, as open-source software. For the exact location where these components can be downloaded, see Table 7 of Annex F.

The following table indicates summary information on the three packages.

<table>
<thead>
<tr>
<th>Software Package</th>
<th>Version</th>
<th>Component Type</th>
<th>Purpose</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Token Service</td>
<td>2.5</td>
<td>H2.2 - STS</td>
<td>Authentication, Delivery of SAML tokens</td>
<td>The Security Token Service, previously called HMA Authentication Service is an open-source implementation (with Apache license) of the service with the same name defined in the HMA User Management Interfaces for Earth Observation specification OGC 07-118. It is able to generate compliant signed and encrypted SAML tokens including assertions with user attributes retrieved from an LDAP registry. The implementation uses ws-security and SAML as explained in the OGC 07-118 specification. The package includes XML Security library for encryption/signature. See [AD12] and [AD13].</td>
</tr>
<tr>
<td>PEP</td>
<td>2.0.2</td>
<td>H2.3 - PEP</td>
<td>Authorisation</td>
<td>In the HMA-T project the SSE Toolbox has been updated with a Security Layer allowing to</td>
</tr>
</tbody>
</table>

---

*Restricted Distribution*  
© Spacebel s.a.  
HMA-S.SPB.D2000.3  
1.2 - 25/04/2014
specify and check that only authenticated and authorized clients can request a given Service (see [AD14] and [AD15]). This Security Layer has been later packaged as a self-contained web application, named PEP. As defined in HMA User Management Interfaces for Earth Observation specification OGC 07-118, Authentication is supported by integrating WS-Security element into SOAP request, in particular by using SAML tokens. Authorization is enforced by defining policies expressed with XACML language; latter policies are also referred as enterprise level policies. The PEP tool allows to protect external SOAP services.

<table>
<thead>
<tr>
<th>CIM EP Client</th>
<th>1.0</th>
<th>H2.1 - Client</th>
<th>Catalogue Client</th>
</tr>
</thead>
<tbody>
<tr>
<td>The “CIM EP Catalogue Client” is a web application that allows a user to access a catalog service compliant with the protocol IF-HMAS-CIM-EP [RD04] on his internet browser. The current client version supports a Setting form so that the user can input a user ID and select a STS for retrieving SAML token. This feature is a step towards supporting integrating this client with a SSO solution, where the user authenticates himself on the SSO IDP and his SSO User Id is provided to the Client by the IDP on the fly.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CIM EP Service</th>
<th>3.0</th>
<th>H2.4 - Service</th>
<th>Catalogue Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIM Demonstrator (see D5000.2 document [RD08])</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 3.2 Interfaces

The following interfaces were depicted on the component diagram in the previous section.

#### 3.2.1 IF-HMAS-Authenticate

<table>
<thead>
<tr>
<th>Service Category</th>
<th>Identity Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference to specification</td>
<td>OGC 07-118r8, User Management for Earth Observation Services, version 1.1, prepared by HMA-S [AD05]</td>
</tr>
</tbody>
</table>
| Standard reference | - OASIS WS-Trust 1.3  
- OASIS SAML 1.1 |
| Description | This interface performs authentication by providing a SAML token in exchange of a (signed) RST (Request Security Token) request with a valid UM-SSO user identifier or password. The returned SAML token can be included in subsequent requests to services which require authentication, e.g. catalogue and ordering. |
| Format | Web services  
SOAP 1.1 over HTTP 1.1 |
Purpose

Comment

The Service Viewpoint of this HMA-compliant interface is described in detail in section 5.3.3 of [RD12]. It is assumed that this authentication service shares the user registry with the UM-SSO IDP.

3.2.2 IF-SSO-Authenticate

<table>
<thead>
<tr>
<th>Service Category</th>
<th>Identity Management</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference to specification</td>
<td>EO op UM-SSO Interface Control Document, SIE-EO-OP-UM-SSO-ICD-002, [RD03].</td>
</tr>
<tr>
<td>Standard reference</td>
<td>None.</td>
</tr>
<tr>
<td>Description</td>
<td>This interface performs authentication on the UM-SSO Identity Provider (IdP) at ESA. The UM-SSO IdP is part of the CDS CRAAS system at ESRIN and will be interfaced by the future ngEO system as well. Such authentication consists of an interaction between the UM-SSO Checkpoint which is part of the Web client and the IdP. Once authenticated, the Checkpoint passes UM-SSO user identifier and selected attributes in the HTTP header of HTTP requests when protected resources (Web pages) are accessed. The Web client can then detect from these attributes which authenticated user is accessing the Web pages. The interface is identical to the future IF-ngEO-UserAuthentication interface.</td>
</tr>
<tr>
<td>Format</td>
<td>Web services</td>
</tr>
<tr>
<td>Purpose</td>
<td>HTTP 1.1 GET</td>
</tr>
</tbody>
</table>

3.2.3 IF-XXX

The details of the interface IF-XXX are not important for the purpose of the Identity Mgt demonstration. The principle is valid for all SOAP-based and REST-based services as explained in the updated specification [AD05]. The particular example used in this demonstrator corresponds to the CIM catalog interface updated in the frame of HMA-S [RD04].

Note that the Demonstrator can be changed to protect another service. You should however take into account that the software packages implementing the client component H2.1 and Policy Enforcement Point H2.3 are limited to SOAP 1.1 services and require modifications to be used for REST-full services.
3.3 Persistent HMA-S Testbed

The Demonstrator components described in the current document can be downloaded individually by an interested user (See download links in Annex F), or can be accessed on-line as they are part of the persistent Testbed at ESRIN. The addresses where the components are accessible are included below:

### 3.3.1 TestBed Software Component Instances

<table>
<thead>
<tr>
<th>Software Instance</th>
<th>Configuration Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2.2 STS</td>
<td>Access URL: <a href="http://VM-S-IP/axis2/services/sts">http://VM-S-IP/axis2/services/sts</a></td>
</tr>
<tr>
<td></td>
<td>Identifier: urn:ceos:def:epr:spacebel:1.0:sts</td>
</tr>
<tr>
<td>H2.2 STS -1</td>
<td>Access URL: <a href="http://VM1-IP/axis2/services/sts">http://VM1-IP/axis2/services/sts</a></td>
</tr>
<tr>
<td></td>
<td>Identifier: urn:ceos:def:epr:gcm:1.0:sts-1</td>
</tr>
<tr>
<td></td>
<td>Main character: This STS represents a Identity management service at a site, e.g. a GCM like VITO, that supports a RST request for the SAML token for user &quot;TestUser1&quot;. To support this, the LDAP directory connected to this STS-1 instance stores only one record for the user TestUser1.</td>
</tr>
<tr>
<td>H2.2 STS -2</td>
<td>Access URL: <a href="http://VM2-IP/axis2/services/sts">http://VM2-IP/axis2/services/sts</a></td>
</tr>
<tr>
<td></td>
<td>Identifier: urn:ceos:def:epr:esa:1.0:sts-2</td>
</tr>
<tr>
<td></td>
<td>Main character: This STS represents a Identity management service at another site, e.g. at ESA, that supports a RST request for the SAML token for user &quot;TestUser2&quot;. To support this, the LDAP directory connected to this STS-2 instance stores only one record for the user TestUser2.</td>
</tr>
<tr>
<td>H2.3 PEP</td>
<td>Access URL:</td>
</tr>
</tbody>
</table>
### Table 1: Software Component Instance (Service)

<table>
<thead>
<tr>
<th>Software Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>identity-mgt-demonstrator-vm-v1-0.ovf</td>
<td>The OVF template that can be used to install the VMs without having to install the software from scratch.</td>
</tr>
<tr>
<td>sts-bin-v2.5.zip</td>
<td>Binary package of the HMA Security Token Service version 2.5</td>
</tr>
</tbody>
</table>

#### 3.3.2 TestBed Software Configuration Items

For users that are interested to install the Demonstrator at their premise, the following table provides the list of software configuration items that can be downloaded. These software items are the input for installation procedure provided in chapter 4.
<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEP.war</td>
<td>Binary package of the PEP full release version 2.0</td>
</tr>
<tr>
<td>PEP_patch_2.0.1.tar</td>
<td>Binary package of the PEP patch 2.0.1</td>
</tr>
<tr>
<td>PEP_patch_2.0.2.tar</td>
<td>Binary package of the PEP patch 2.0.2</td>
</tr>
<tr>
<td>policy_getRecords.xml</td>
<td>A sample XACML policy file</td>
</tr>
<tr>
<td>catalogueClient-bin-v1.0.zip</td>
<td>Binary package of CIM-EP Catalogue client.</td>
</tr>
<tr>
<td>sts.keystore</td>
<td>Self-signed keystore for the STS installed in the VM-B (VM base)</td>
</tr>
<tr>
<td>pep.keystore</td>
<td>Self-signed keystore for the PEP installed in the VM-B</td>
</tr>
<tr>
<td>sts-1.keystore</td>
<td>Self-signed keystore for the STS-1</td>
</tr>
<tr>
<td>sts-1-authentication-service.properties</td>
<td>Configuration file for the STS-1 instance</td>
</tr>
<tr>
<td>sts-2.keystore</td>
<td>Self-signed keystore for the STS-2</td>
</tr>
<tr>
<td>sts-2-authentication-service.properties</td>
<td>Configuration file for the STS-2 instance</td>
</tr>
<tr>
<td>sts-2-federated-sts.properties</td>
<td>STS federation configuration file for the STS-2 instance</td>
</tr>
<tr>
<td>pep-1.keystore</td>
<td>Self-signed keystore for the PEP-1</td>
</tr>
<tr>
<td>pep-2.keystore</td>
<td>Self-signed keystore for the PEP-2</td>
</tr>
</tbody>
</table>

Table 2: TestBed Software Configuration Items

*Note: See Annex E for the Ant build script that creates and configures the keystores ready to be used in the Demonstrator. Each keystore file is created by Ant target named "<keystore-filename>" and...*
preconfigured by Ant target named “config.<keystore-filename>”, where <keystore-filename> is the same with filename of each keystore item shown on the table above.

3.4 Demonstrator Configurations

In the present section, we provide summary information on the different ways to deploy the Identity Management Services Demonstrator of HMA-S. A strong emphasis is put on configuration of keystores (location and purpose of different private / public keys), which is the cornerstone to achieve the integration of the different components.

3.4.1 Virtual Machines

The following table describes the machines used in the Demonstrator. Note that not all these machines have to be deployed, but depending on each Demonstrator configuration. The following table describes the VMs in details. Note that VM-1, VM-2, VM-3 contain the same software than VM-S. They just differ at the HMA-S component configuration level. VM-3 can also be a reuse of the VM provided by HMA-S task 3.

<table>
<thead>
<tr>
<th>Purpose</th>
<th>VM-S</th>
<th>VM-1</th>
<th>VM-2</th>
<th>VM-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>A VM on which all software components are installed. The software components are integrated together. This VM can be used for Demonstrator where the Client, the STS and the PEP runs on the same machine. This VM is used to create the OVF export (identity-mgt-demonstrator-vm-v1-0.ovf). Represents a STS/IDP instance at a GCM site, for example at VITO. Represents a STS/IDP instance at another site, for example at ESA</td>
<td>Represent a Service Client</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software instance (see 3.3.1 for description of these software instances)</td>
<td>STS</td>
<td>STS-1</td>
<td>STS-2</td>
<td>CIM-EP-Client</td>
</tr>
<tr>
<td></td>
<td>PEP</td>
<td>PEP-1</td>
<td>PEP-2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CIM-EP client</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Users (included in the LDAP directory on the VM)</td>
<td>TestUser1</td>
<td>TestUser1</td>
<td>TestUser2</td>
<td>Not Applicable</td>
</tr>
<tr>
<td></td>
<td>TestUser2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Virtual Machines
3.4.2 DC1 - Demonstrator Configuration where STS is IdP

The following diagram summarizes the deployment architecture without Web SSO. In this case, the STS is the IdP and the STS receives *RST with password*.

The RST is a SOAP request on HTTPS.

The LDAP user registry is accessed by STS for authentication and for retrieval of attributes to be put in the SAML token.

Two keystores are required.
- The **STS keystore** contains:
  - STS private key: used to sign SAML token to be delivered,
  - Relying Party public key: used to encrypt the SAML token to be delivered; the relying party is the entity that runs the PEP that shall consume the SAML token.

*Figure 2: Configuration where STS is IdP (Part 1)*
- The **PEP keystore** contains:
  - Relying Party private key: used to decrypt received SAML token,
  - STS public key: used to verify signature of the received SAML token.

Note that we assume here that each component has one unique keystore. This is just for simplification; this constraint is not mandatory provided that each component is able to retrieve the expected key for each purpose in some keystore. It will be explained that the software package used for the PEP use several keystores.

As explained above, the present architecture covers only the first use case of OGC 07-118. It shall be enhanced so that the “front-end” STS (i.e. federating) is able to delegate authentication and building of SAML token to another STS (i.e. federated). A second STS is then required for that purpose. Note that, in the scope of the present activity (demonstrator), it is not foreseen to use more than one federated STS, although OGC 07-118 does not limit the number of STS.
3.4.3 DC2 - Demonstrator Configuration where STS is not IdP

The following diagram summarizes the deployment architecture where STS is not the IdP, i.e. the STS does not authenticate the user but trusts that the requesting client has made authentication checks through another IdP, which is based for example on a Web-SSO system like Shibboleth [RD01]; in this case the STS receives RST with signature.

The RST is a SOAP request on HTTPS.

The LDAP user registry is accessed by STS just for the retrieval of the attributes to be put in the SAML token (no password-based authentication).

![Configuration Diagram](image)

Figure 4: Configuration where STS is not IdP (part 1)

The second part of this configuration is identical to the configuration shown on Figure 3.

Three keystores are required:
- The **Client keystore** contains:
  - Client private key : used to sign the RST sent to STS
- The **STS keystore** contains:
  - STS private key : used to sign SAML token to be delivered,
Relying Party’s public key: used to encrypt the SAML token to be delivered; the relying party is the entity that runs the PEP that shall consume the SAML token,

Trusted client private key(s): used to verify signature of received RST; there is one key for each client that is trusted by STS.

- The **PEP keystore** contains:
  - PEP private key: used to decrypt received SAML token,
  - STS public key: used to verify signature of the received SAML token.

Note that this configuration is a superset of the configuration seen in section 3.4.2. This means that the architecture of the present section allows also to accept RST with password, provided that LDAP user registry has entries with password attribute. Such architecture allows having a multi-purpose STS that is able to afford both users authenticated by STS (RST with password) and users authenticated by an external IdP through a trusted client (RST with signature).

As explained before, we assume here that each component has one unique keystore. This is just for simplification; this constraint is not mandatory. It will be explained that the software package used for the PEP relies on several keystores.

To deploy this DC2 configuration, you can install only the VM-S machine.

### 3.4.4 DC3 - Demonstrator Configuration with Multiplicity and With DelegateTo

The purpose of this configuration is to demonstrate the case where multiple instances of the H2.2 - STS and multiple instances of the H2.3 - PEP are deployed together and interconnected by means of the AppliesTo parameter (in the RST request) and the Issuer attribute (in the SAML Token).

See section 6.4.1.1 of and section 6.4.2 of [AD05] for more description about these two features.

This configuration also demonstrates the use case with the STS delegation where the STS-2 is delegating RST requests to the STS-1 (Delegated STS).

It can be highlighted from this demonstrator configuration the following main points:

- One Client works with multiple STS instances: the same H2.1 Client component is configured to work with two STS instances. For SAML tokens of user TestUser1, it requests the STS instance STS-1. For tokens of user TestUser2, it requests the STS instance STS-2. Note that the Client can request SAML token of TestUser1 by sending a RST request with Delegation to the STS-1.

- One STS instance works with multiple PEP instances: The Client instructs the STS instance to return the SAML token that is to be used in Service requests that address either the PEP-1 or PEP-2.

- Multiple STS instances work with the same PEP: SAML tokens returned by both STS-1 and STS-2 can be used in the Service requests that address either the PEP-1 or PEP-2.
Figure 5: Configuration with Multiplicity (part 1)
As shown on the Figures above, we can notice that there are two STS instances and two PEP instances deployed in the installation. The following highlights the main features in the configuration that allows to connect the software components together:
The keystore of STS-1 instance includes a self-signed key pair (private and public keys) under alias string "urn:ceos:def:epr:gcmm:1.0:sts-1". This alias string uniquely identifies this STS instance in the installation. The alias string is then set to the property “SAML_ASSERTION_ISSUER” in the configuration file of the STS-1 so that the string is set to the SAML attribute “Issuer” in SAML tokens returned by this STS-1 instance. On the keystore of PEP-1 and PEP-2, the alias string is used as alias for the public key of the STS-1.

The keystore of STS-1 includes the public key of PEP-1 under alias "urn:ceos:def:epr:gcmm:1.0:pep-1" and public key of PEP-2 under alias "urn:ceos:def:epr:esa:1.0:pep-2". On processing a RST request, the STS-1 instance retrieves one of these two public keys by the value of parameter "AppliesTo" in the RST request, to encrypt the SAML token.

The keystore of STS-1 stores also the public key of the CIM-EP client under alias "urn:ceos:def:epr:spacebel:1.0:cim-ep-client". This alias value is then included in the STS configuration parameter “CLIENT_CERTIFICATE_ALIASES”. The purpose of this is to make the STS instance trusts the CIM-EP client when processing RST requests with signature, e.g. case with SSO integration (where the STS is not the IDP).

In the same manner, the keystore of STS-2 includes a self-signed key pair under alias "urn:ceos:def:epr:esa:1.0:sts-2", which uniquely identifies the STS-2 instance. The STS configuration property “SAML_ASSERTION_ISSUER” is set to this alias value. The keystore includes public keys of the PEP instances and the CIM-EP instance under the same aliases as used in the keystore of the STS-1. There is not constraint for this similarity, but for simplification reason in the Demonstrator setup.

The keystore of PEP-1 instance includes a self-signed key pair under alias "urn:ceos:def:epr:gcmm:1.0:pep-1". The keystore includes public key of the STS-1 instance under alias "urn:ceos:def:epr:gcmm:1.0:sts-1", which is the same value that is set to the STS configuration property “SAML_ASSERTION_ISSUER” of the STS-1, as described above. For more detail, the PEP instance retrieves this STS public key from its keystore by looking for alias that matches the SAML attribute “Issuer” in the SAML token. It then uses the STS public key to validate the STS instance’s signature. With the same purpose, the keystore includes public key of STS-2 under alias "urn:ceos:def:epr:esa:1.0:sts-2".

Similarly, the keystore of PEP-2 instance includes a self-signed key pair under alias "urn:ceos:def:epr:esa:1.0:pep-2". It includes public key of STS-1 instance under alias "urn:ceos:def:epr:gcmm:1.0:sts-1" and of STS-2 instance under alias “urn:ceos:def:epr:esa:1.0:sts-2"

In this Demonstrator configuration, we intend to depict the possible use case where the identity management services are inter-cooperated between two different sites, for example ESA and a GCM such as VITO. Specifically, the STS-2 and PEP-2 represent an STS and PEP installation at ESA site. The STS-1 and PEP-1 represent one at the GCM site.

The LDAP directory connected with the STS-1 does not include TestUser2 but only TestUser1, and the LDAP directory of the STS-2 does not include TestUser1 but only TestUser2. Note that this setup is specific for this Demonstrator configuration, and is not related to any constraint of the software as the directories can includes any number of user entries.

To deploy this DC3 configuration, you need to install three machines VM-1, VM-2 and VM-3.
4 SOFTWARE INSTALLATION MANUAL

The purpose of the present section is to provide procedure that allows to deploy one of the Demonstrator configurations as described in chapter 3.4. The procedure reference to a set of different software packages that implement the architecture described in OGC 07-118 ([AD05]). It provides also installation and configuration information to make these packages fit together.

4.1 Introduction

All the afore-mentioned packages contain an installation guide, covering prerequisites and procedures.

The present section focuses on higher-level topics, at the level of integrating these components to set-up the HMA-S Identity Management Services Demonstrator.

Actually, the HMA Security Token Service encompasses the LDAP directory installation; this is optional since any pre-existing LDAP v3 user registry can be used. So, there are up to four components to be installed:

1. LDAP directory (part of HMA Security Token Service),
2. STS (part of HMA Security Token Service),
3. PEP implementation,
4. CIM EP client

Note that the CIM EP server is assumed to be already deployed. The CIM EP Client may also already exist on the CIM EP server. The installation of this client is duplicated here in case the only interactions of interest for the user is the interaction between the client and the STS.

These packages are all installed on a single Virtual Machine (VM) so that the OVF template is created (identity-mgt-demonstrator-vm-v1-0.ovf as defined in section 3.3.2). The subsection “Install single VM” below in this section provides the procedure to install this machine from the binary code of each software component – install from scratch.

The software components may be installed in any order, but it is advised to follow the order given below. Because the interdependencies between components (URL, keys,...), the integrated configuration require coordinated changes to the configuration on each of the installed components.

These components may be distributed on any numbers of machines. However, in an operational environment, it is expected that LDAP directory runs on the same machine than STS for security reasons. If this is not the case, then security threats about LDAP access shall be assessed and countermeasures shall be taken, like the use of LDAPS.

The machines VM-1, VM-2, and VM-3 are to be installed from the OVF template mentioned above. The subsections “Install VM-1”, “Install VM-2”, and “Install VM-3” below provide the procedure to install each of these VMs.

4.2 Prerequisites

A Host machine is needed to host the virtual machines for the installation. In the scope of the Task, we have not precisely determined the minimum configuration for this Host. Theoretically, a Host that can run three virtual machines (VM) with Redhat Enterprise Linux 6.3 (64 bits) operating system, each VM requires:

- 50 GB of disk space
- 1 GB of memory

The following is the Host server used at Spacebel to implement and test installation procedure for the Demonstrator Configurations.

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processors</td>
<td>2x Intel Xeon E5-2640 Processor (2.50GHz, 6C, 16M Cache, 7.2 GT/s QPI, 95W TDP, Turbo, HT)</td>
</tr>
<tr>
<td>Physical RAM</td>
<td>64GB Memory for 2 CPU (8x8GB Dual Rank LV RDIMMs) 1333MHz, Using 1333MHz DIMMs</td>
</tr>
<tr>
<td>Hard disk</td>
<td>Internal SSD Module with 2GB SD Card + Raid system with 6x 600GB SAS 10k 2.5&quot; HD Hot Plug</td>
</tr>
<tr>
<td>CD/DVD drive</td>
<td>DVD-ROM, Second Ethernet card</td>
</tr>
<tr>
<td>Hypervisor</td>
<td>VMware ESXi version 5.1.0</td>
</tr>
<tr>
<td>Name / IP Address</td>
<td>DAIL-SSE / 72.19.1.10</td>
</tr>
</tbody>
</table>

A client PC to run the vSphere Client version 5.1.0 application and an account to connect to the host (via the vSphere client) with sufficient privileges to create/modify/remove virtual machines onto the Host.

The software files as defined in the section 3.3.2 "TestBed Software Configuration Items" in this document are available.

4.3 Install the Single VM (from Scratch)

4.3.1 STEP 1: Install Linux

Follow the procedure provided in Annex A.1 to create the VM and have it up and running on the Host. This IP address of this VM will be referred as “VM_IP” in this chapter.

4.3.2 STEP 2: Install STS

Perform the steps provided in Annex B.1 to configure the OpenLDAP as follows:

- At Step 1 "Install OpenLDAP", skip the commands that install these RPM packages of the software (because they are already installed by the Kickstart provided Annex A.1). Executes only the two commands to register the OpenLDAP service into the operating boot process and have it started.
- At Step 2, note down the LDAP suffix and the password of the OpenLDAP admin to be used in the next steps below.

Perform the steps provided in Annex B.2 to install the JDK.

Perform the steps provided in Annex B.2.3 to install the Tomcat server onto directory /apps.
Perform the steps provided in Annex B.4 to install the Axis2 application.

Perform the steps provided in Annex B.5 to install the HMA Security Token Service (STS), as follows:

- At step that updates the STS configuration file (authentication-service.properties) in section “Install HMA Security Token Service”, update only the following properties and keep default values for the rest:
  - LDAPSearchContext: update to apply the LDAP suffix of the OpenLDAP.
  - LDAPPrincipal: update to apply the LDAP suffix of the OpenLDAP.
  - LDAPCredentials: update to apply the admin password of the OpenLDAP.
  - KEystore_LOCATION: set to /apps/conf/keystores/sts.keystore. This keystore file is to be installed in the next steps below.
  - CLIENT_CERTIFICATE_ALIASES: set to ”urn:ceos:def:epr:spacebel:1.0:cim-ep-client”.
  - LOCAL_STS_URN: to be put in comments (add # at the beginning of the line) to not apply this property to configure the STS as a federating STS, and not federated STS.

- Keep the file federated_sts.properties unchanged.

- At step to update the log4j configuration file, set /apps/log/sts/sts.log. Note to create the directory /apps/log/sts if it does not yet exist.

Copy keystore file sts.keystore (see section 3.3.2 for detail of this software file) to directory /apps/conf/keystores/

4.3.3 STEP 3: Install PEP

In the Demonstrator, PEP version 2.0 and its two existing patches 2.0.1 and 2.0.2 are installed.

Perform procedures provided in Annex C.2 to install the software, taking into account the following notes:

- Use the Tomcat server that is already installed for the STS.
- Use the JDK that is already installed for the STS. This JDK is put at directory /usr/java/jdk1.7.0_45

After installing the PEP.war file, make sure that the Tomcat server is started.

Wait until the directory PEP is created under $TOMCAT_HOME/webapps/, then do the following to install the PEP patches:

- Expand file PEP_patch_2.0.1.tar to directory $TOMCAT_HOME/webapps
- Expand file PEP_patch_2.0.2.tar to directory $TOMCAT_HOME/webapps

Perform procedures provided in Annex C.3 to configure the PEP, as follows:

- Net Configuration:
  - PEP Endpoint address: set to the IP address of the current VM
  - Port Number: set to 80
  - SSL Port number: set to 443
- General Security Configuration:
  - PEP level keystore: upload the file pep.keystore (defined in section 3.3.2)
- Keystore password: set to "changeit"
- IdP key alias: set to "urn:ceos:def:epr:spacebel:1.0:pep"

Log Settings:
- Logging Level: INFO
- Log File Size: 5000

Contact Information: fill with your information or can be left empty.

The next step is to create the PEP service that will protect the H2.4 Service (CIM I15 Catalogue). But due to a limitation of PEP that it cannot parse correctly the WSDL of the I15 catalogue when multiple SOAP binding are defined in the WSDL, a preliminary step is needed to update the CIM I15 WSDL file as described hereafter:

- Copy and rename the updated WSDL file at $TOMCAT_HOME/webapps/PEP/i15.wsdl
- Edit the i15.wsdl file to remove the XML element "wsdl:port name="CswPort1.2". This is to work around the problem of the current PEP software, that it cannot parse correctly the WSDL of the I15 catalogue when multiple SOAP binding are defined in the WSDL.

Perform the procedure provided in Annex C.4 – Create a PEP service, to create the PEP service with the following input:

The PEP home page URL is http://VM_IP/PEP.

At step "New Pep Service",
- Service WSDL: http://VM_IP/PEP/i15.wsdl
- Service Name: Secured-I15.
- Select checkbox "Forward message with incoming security token"

At step "Security Configuration",
- Keystore: the keystore file pep.keystore
- Alias: urn:ceos:def:epr:spacebel:1.0:pep
- Password: changeit

At step "Authentication Checks", keep default values.

At step "Authorisation Checks",
- select checkbox "Custom step for the DAIL PEP" (to have it checked)
- Policies directory: /apps/conf/policies

When finish, click button "Create Service".

When prompted by the PEP to select PEP operations,
- Unselect checkbox "Token mandatory"
- Select checkbox "csw.getRecords"
Click button “Create PEP Operations”, then click button OK to complete the creation for the PEP service “Secured-I15”. The endpoint of this PEP service is http://VM_IP:80/PEP/services/Secured-I15.

Create directory /apps/conf/policies/catalogue on the machine and then copy to this directory the file policy_getRecords.xml as defined in section 3.3.2.

For more information, the file policy_getRecords.xml includes a sample policy set that is analysed by the PEP to authorises catalogue requests from the catalogue client. With this policy set, only requests for users of that the SAML attribute “UserProfile” is Scientific or GMES. All other cases are denied. Thus, only requests with SAML tokens of the TestUser2 are authorized. All requests of TestUser1 and anonymous users are denied.

4.3.4 STEP 4: Install CIM-EP Catalogue Client

Perform the procedure provided in Annex D to install the catalogue client as follows:

At step “Installation” use the Tomcat server that is already installed for the STS.

At step “Catalogue Registration”, replace the existing content of file service.xml by the following:

```xml
<services xmlns="http://www.spacebel.be/catalogue/client">
  <service>
    <id>1</id>
    <name>Secured I15 Catalogue (Task 2)</name>
    <icd>ogc-13-084r1</icd>
    <operation name="Search">
      <xsl>I15_Catalogue_Search.XSL</xsl>
      <binding>
        <soap action="http://www.opengis.net/cat/csw/2.0.2/requests#GetRecords">http://VM_IP:80/PEP/services/Secured-I15</soap>
      </binding>
    </operation>
    <operation name="Present">
      <xsl>I15_Catalogue_Present.XSL</xsl>
      <binding>
        <httpget>115-HTTP-ADDRESS</httpget>
      </binding>
    </operation>
  </service>
</services>
```

Where,
- **VM_IP** is replaced by the IP address (or full domain name) of the single machine VM.
- **115-HTTP-ADDRESS** is replaced by the HTTP access address of the CIM EP (I15) Catalogue server as described in [RD08], e.g. http://hma-s-catalogue.spacebel.be/hmas/httpservice

At step “SSO IDP Registration” and step “Keystore configuration”, apply the following property for content of the file idp.properties:

sts=STS (Spacebel)
And the following text for file um.properties:

```
clientKeystoreLocation=/apps/conf/keystores/cim-ep-client.keystore
clientKeystorePassword=changeit
clientCertificatePassword=changeit
#serviceId = relying parties alias
1=urn:ceos:def:epr:spacebel:1.0:pep
#sts reference (in file idp.properties) = sts endpoints
sts=http://localhost:80/axis2/services/sts
```

and the following text for file delegated-idp.properties:

```
# STS delegation is not applied
na= Not Applicable
```

And copy the keystore cim-ep-client.keystore to directory /apps/conf/keystores/ on the machine.

Lastly, to support HTTPs connection, import the SSL certificate used by the Tomcat server into the JDK trust store. Execute the following command to import the certificate:

```
```

Note: when prompted for the keystore password by the above commands, enter changeit. When prompted “Trust this certificate? [no]”, enter yes followed by Enter to have the import completed.

**4.3.5 STEP 5: Configuration Customization**

The Single VM, as the result of previous steps (STEP 1 to 4 above) is pre-configured with the default configuration that connects together all software components on the machine. At this installation state, the VM is ready to be used for the Demonstrator configuration DC1 and DC2.

With this default configuration:

- The PEP service "Secured CIM-EP Catalogue" is registered on the Client as a CIM EP Catalogue service.
- This PEP service is connected to the pre-existing CIM I15 Catalogue service. A service request sent by the Client goes through this PEP service to the CIM live endpoint. By this way, the PEP software protects this live endpoint.
- The STS and PEP instances are inter-connected via the keys configuration done in the two keystore files: sts.keystore and pep.keystore. In the sts.keystore, we import the public certificate of the PEP and in the pep.keystore, we import the public key of the STS. The sts.keystore includes also the public of the CIM-EP client as a trusted client (case of RST request with signature).

- The SSL certificate of the Tomcat server is imported into the truststore of the Java JRE on the machine. This is to implement the HTTPS connections between the CIM-EP client and the STS.

To customize the default configuration said above, see section 4.7 “Configuration” below for guidelines and the user manual of the software components in Annex B, C, and D.

4.4 Install VM1

4.4.1 Deploy OVF Template

Perform the procedure provided in Annex A.2 to install the VM from OVF template (file identity-mgt-demonstrator-vm-v1-0.ovf, defined in section 3.3.2).

4.4.2 Configure Applications

In this section, we perform a number of reconfigurations for the software instances on this VM to satisfy the constraints of the Demonstrator configuration DC3.

In brief, we need to configure the STS-1 so that:

- Its keystore includes the public key in the pair with the private key in the keystore of the service “Secured-I15” on both PEP-1 and PEP-2. This is to have the SAML tokens it returns are decryptable by the PEP services.

- Its keystore includes the public keys of the CIM-EP client. This is to make the STS trust the CIM-EP client (for use case RST with signature).

- A SAML token returned by the STS has the Issuer attribute valued by the identifier string of the STS-1 (“urn:ceos:def:epr:gcm:1.0:sts-1”)

And configure the PEP-1 so that:

- It has a keystore (PEP level keystore).

- This pep-level keystore includes the public keys of the STS-1 and STS-2. This is to make the PEP-1 accept the SAML tokens returned by the STS-1 and STS-2.

- Its service “Secured-I15” has a keystore – keystore at service level.

- This service-level keystore includes the private key in pair with the public key that are included in the STS keystore said above. Again, this is to have the PEP service be able to decrypt the SAML tokens from both STS-1 and STS-2 instances.

The following describes in detail the steps to configure the VM:

Step 1: Configure the STS
Replace content of the file /apps/apache-tomcat-6.0.37/webapps/axis2/WEB-INF/services/sts.aar/authentication-service.properties on the VM by the content of the file sts-1-authentication-service.properties.

Doing the replacement above, we:

- Apply “urn:ceos:def:epr:gcm:1.0:sts-1” to the property SAML_ASSERTION_ISSUER. This in turns sets the Issuer attribute in the SAML token returned by this STS to “urn:ceos:def:epr:gcm:1.0:sts-1”.
- Set property “LOCAL_STS_URN” to the identification string above so that this STS supports a RST request sent by the STS-2 in the Demonstration configuration DC3. For more information, the STS-1 is configured to be a delegated STS and the STS-2 is configured to be a delegating STS.
- Apply the keystore file sts-1.keystore.

Step 2: Configure the PEP

Step 2.1: Configure PEP level keystore:

Sign in the PEP home page (http://VM1_IP/PEP) under user admin / admin, then select menu Start → Config. This will open a new browser window named “PEP Configuration”.

Select tab “Net Configuration” on the window “PEP Configuration” to replace existing IP address in field “PEP Endpoint address”, e.g. 91.183.191.163, by the IP address of the current VM1 machine.

Select tab “General Security Configuration”, then fill the form as follows:

- PEP level keystore: browse and select the keystore file pep-1.keystore.
- IdP Key Alias: urn:ceos:def:epr:gcm:1.0:pep-1
- Keep the rest unchanged.

When finish, click button “Save Configuration” (on the bottom right of the window), button Ok on the new dialogue window “Save Configuration”.

Step 2.2: Configure Service Secured-I15:

On the PEP home page (in STEP 2.1), select menu Start → PEP services → Secured-I15 → Delete Service. Click button “Yes” on the dialogue window “Delete Service” to confirm deletion of the existing Secured-15 service. Then click button Ok on the dialogue window "Delete PEP Service".

Edit the file i15.wsdl on the machine located at directory /apps/apache-tomcat-6.0.37/webapps/PEP to replace “hma-s-catalogue.spacebel.be” by the IP address of the Catalogue server installed by procedure of [RD08].

Then perform the procedure provided in Annex C.4 – Create a PEP service, to create a new PEP service with the following input:

The PEP home page URL is http://VM1_IP/PEP.

At step ”New Pep Service”,

- Service WSDL: http://localhost/PEP/i15.wsdl
- Service Name: Secured-I15.
- Select checkbox “Forward message with incoming security token”

At step “Security Configuration”,
- KeyStore: upload the keystore file pep-1.keystore.
- Alias: urn:ceos:def:epr:gcm:1.0:pep-1
- Password: changeit

At step “Authentication Checks”, keep default values.
At step “Authorisation Checks”,
- select checkbox “Custom step for the DAIL PEP” (to have it checked)
- Policies directory: /apps/conf/policies

When finish, click button “Create Service”.

When prompted by the PEP to select PEP operations,
- Unselect checkbox “Token mandatory”
- Select checkbox “csw.getRecords”

Click button “Create PEP Operations”, then click button OK to complete the creation for the PEP service “Secured-I15”. The endpoint of this PEP service is http://VM1_IP:80/PEP/services/Secured-I15.

In this step, we have:

- Apply the PEP level keytore and the Service level keystore using the the keystore file pep-1.keystore. This keytore includes a self-signed key pair for the PEP, the public keys of the STS-1 and STS-2. See section 3.3.2 for more detail of this software item.
- The Secured-I15 service on this PEP is accessible at URL http://VM1_IP/PEP/services/Secured-I15, is connected to the CIM catalogue server installed by procedure of [RD08]

Step 3: Exclude the CIM-EP Client
On the machine VM1, we don't need the CIM-EP client.
As root on the VM, run the following commands:
Root>service tomcat stop
Root>mkdir /apps/bk
Root>mv /apps/apache-tomcat-6.0.37/webapps/catalogueClient /apps/bk
Root>service tomcat start

Step 4: Configure LDAP Directory
We need to remove from the LDAP directory the entry for user TestUser2.
Run the following command as root on the VM to do that:

```
ldapdelete -D "cn=ldapRoot,dc=spacebel,dc=be" -w secret
"cn=TestUser2,ou=Users,dc=spacebel,dc=be"
```

### 4.5 Install VM2

#### 4.5.1 Deploy OVF Template

Perform the procedure provided in Annex A.2 to install the VM from OVF template (file identity-mgt-demonstrator-vm-v1-0.ovf, defined in section 3.3.2).

#### 4.5.2 Configure Applications

In the same way that we configure the software on the VM1, we need to configure the software instances on this VM to satisfy the constraints of the Demonstrator configuration DC3.

**Step 1: Configure the STS**

Replace content of the file `/apps/apache-tomcat-6.0.37/webapps/axis2/WEB-INF/services/sts.aar/authentication-service.properties` on the VM by the content of the file `sts-2/authentication-service.properties`.

Replace content of the file `/apps/apache-tomcat-6.0.37/webapps/axis2/WEB-INF/services/sts.aar/federated-sts.properties` by the content of file `sts-2-federated-sts.properties`

Replace string “VM1_IP” in the file `/apps/apache-tomcat-6.0.37/webapps/axis2/WEB-INF/services/sts.aar/federated-sts.properties` by the IP address of the VM1 machine.

Doing the replacement above, we:

- Apply “urn:ceos:def:epr:esa:1.0:sts-2” to the property SAML_ASSERTION_ISSUER. This in turns sets the Issuer attribute in the SAML token returned by this STS to “urn:ceos:def:epr:esa:1.0:sts-2”.
- Comment out the property “LOCAL_STS_URN” and configure the file federated-sts.properties to make the STS-2 support delegating RST requests to the STS-1
- Apply the keystore file `sts-2.keystore`.

**Step 2: Configure the PEP**

Repeat the same configuration as for the VM1 but with the following inputs:

**PEP Net Configuration:**
- The IP address of the VM2 machine.

**PEP level keystore:**
- PEP level keystore: `pep-2.keystore`.
- IdP Key Alias: `urn:ceos:def:epr:esa:1.0:pep-2`
Service Secured-I15:
- The PEP home page http://VM2_IP/PEP
- Keystore: pep-2.keystore

Step 3: Exclude CIM-EP-Client
Repeat the same reconfiguration as for the VM1 to exclude the CIM-EP client from the VM.

Step 4: Configure LDAP Directory
We need to remove from the LDAP directory the entry for user TestUser1.
Run the following command as root on the VM to do that:
ldapdelete -D "cn=ldapRoot,dc=spacebel,dc=be" -w secret "cn=TestUser1,ou=Users,dc=spacebel,dc=be"

4.6 Install VM3
In this technical note document, the machine VM3 is designed to run the CIM-EP catalogue client. It has the same function of the machine installed by procedure in [RD06].

If for some reason, you don't want to install the machine as per procedure in [RD06], but still want a CIM-EP catalogue client (for using the Demonstrator), then perform the procedures provided in two following subsections: 4.6.1 and 4.6.2.

In case you already have the machine installed by procedure in [RD06], you only need to perform the procedure in section 4.6.3.

4.6.1 Deploy OVF Template
Perform the procedure provided in Annex A.2 to install the VM from OVF template (identity-mgt-demonstrator-vm-v1-0.ovf, defined in section 3.3.2).

4.6.2 Configure Applications
In this section, we configure the VM to satisfy the constraints of the Demonstration configuration DC3.

Step 1: Replace content of file /apps/apache-tomcat-6.0.37/webapps/catalogueClient/WEB-INF/classes/service.xml by the following

```
<services xmlns="http://www.spacebel.be/catalogue/client">
  <service>
    <id>1</id>
    <name>Secured CIM-EP Catalogue 1</name>
    <icd>ogc-13-084r1</icd>
    <operation name="Search">
      <xsl>I15_Catalogue_Search.XSL</xsl>
    </operation>
  </service>
</services>
```
<binding>
  <soap action="http://www.opengis.net/cat/csw/2.0.2/requests#GetRecords">http://VM1-IP:80/PEP/services/Secured-I15</soap>
</binding>
</service>
<service>
  <id>2</id>
  <name>Secured CIM-EP Catalogue 2 (Task 2)</name>
  <icd>ogc-13-084r1</icd>
  <operation name="Search">
    <xsl>I15_Catalogue_Search.XSL</xsl>
    <binding>
      <soap action="http://www.opengis.net/cat/csw/2.0.2/requests#GetRecords">http://VM2-IP:80/PEP/services/Secured-I15</soap>
    </binding>
  </operation>
  <operation name="Present">
    <xsl>I15_Catalogue_Present.XSL</xsl>
    <binding>
      <httpget>I15-HTTP-ADDRESS</httpget>
    </binding>
  </operation>
  <aoiRequired>false</aoiRequired>
</service>
</services>

Note to replace VM1_IP and VM2_IP by the IP addresses of the VM1 and VM2 machines. And replace I15-HTTP-ADDRESS by the http address of the CIM EP (I15) catalogue service as installed by procedure in [RD08].

Step 2: Replace the content of file /apps/apache-tomcat-6.0.37/webapps/catalogueClient/WEB-INF/classes/um.properties by the following:

clientKeystoreLocation=/apps/conf/keystores/cim-ep-client.keystore
clientKeystorePassword=changeit
clientCertificatePassword=changeit
#serviceId = relying parties alias
1=urn:ceos:def:epr:gcm:1.0:pep-1
2=urn:ceos:def:epr:esa:1.0:pep-2
#sts reference (in file idp.properties) = sts endpoints
sts-1=http://VM1_IP:80/axis2/services/sts
sts-2=http://VM2_IP:80/axis2/services/sts

Note to replace VM1_IP and VM2_IP by the IP addresses of the VM1 and VM2 machines.

Step 3: replace content of the file /apps/apache-tomcat-6.0.37/webapps/catalogueClient/WEB-INF/classes/idp.properties by the following:

sts-1=STS-1(GCM SSO IDP)
sts-2=STS-2(ESA SSO IDP)

Step 4: Replace content of file /apps/apache-tomcat-6.0.37/webapps/catalogueClient/WEB-INF/classes/delegated-idp.properties by the following:

na= Not Applicable
urn:\ceos\def\epr\gcm\1.0\:sts-1 = STS-1 (GCM)

Step 5: Remove STS and PEP
As root on the VM, run the following commands:
Root>service tomcat stop
Root>rm -Rf /apps/apache-tomcat-6.0.37/webapps/PEP.war
Root>rm -Rf /apps/apache-tomcat-6.0.37/webapps/PEP
Root>rm -Rf /apps/apache-tomcat-6.0.37/webapps/axis2.war
Root>rm -Rf /apps/apache-tomcat-6.0.37/webapps/axis2
Root>service tomcat start

4.6.3 Configure existing CIM-EP Client

This section is to configure the CIM-EP client on the machine as outcome of the procedure in [RD06] in order to integrate it with the STS/PEP applications on the machine VM-1 and VM-2.

Step 1: Insert to the file $TOMCAT_HOME/webapps/catalogueClient/WEB-INF/classes/service.xml the following two XML elements "service":

```xml
<service>
  <id>A_NEW_ID1</id>
  <name>Secured CIM-EP Catalogue 1</name>
  <icd>ogc-13-084r1</icd>
  <operation name="Search">
    <xsl>115_Catalogue_Search.XSL</xsl>
  </operation>
  <binding>
```
<soap action="http://www.opengis.net/cat/csw/2.0.2/requests#GetRecords">http://VM1-IP:80/PEP/services/Secured-I15</soap>
</binding>
</operation>
<operation name="Present">
<xsl>II5_Catalogue_Present.XSL</xsl>
<binding>
<httpget>CIM-EP-HTTP-ADDRESS</httpget>
</binding>
</operation>
<aoiRequired>false</aoiRequired>
</service>
<service>
<id>A_NEW_ID2</id>
<name>Secured CIM-EP Catalogue 2 (Task 2)</name>
<icd>ogc-13-084r1</icd>
<operation name="Search">
<xsl>II5_Catalogue_Search.XSL</xsl>
<binding>
<soap action="http://www.opengis.net/cat/csw/2.0.2/requests#GetRecords">http://VM2-IP:80/PEP/services/Secured-I15</soap>
</binding>
</operation>
<operation name="Present">
<xsl>II5_Catalogue_Present.XSL</xsl>
<binding>
<httpget>CIM-EP-HTTP-ADDRESS</httpget>
</binding>
</operation>
<aoiRequired>false</aoiRequired>
</service>

Note to replace:
- **VM1_IP** and **VM2_IP** by the IP addresses of the VM1 and VM2 machines.
- **CIM-EP-HTTP-ADDRESS** by the http address of the CIM EP (I15) catalogue service as installed by procedure in [RD08].
- **A_NEW_ID1** and **A_NEW_ID2** by identification numbers that are not yet used for other service elements in the service.xml file.

Step 2: Replace the content of file /apps/apache-tomcat-6.0.37/webapps/catalogueClient/WEB-INF/classes/um.properties by the following:

```properties
clientKeystoreLocation=/apps/conf/keystores/cim-ep-client.keystore
clientKeystorePassword=changeit
clientCertificatePassword=changeit
```
#serviceId = relying parties alias
A_NEW_ID1=urn:ceos:def:epr:gcm:1.0:pep-1
A_NEW_ID1=urn:ceos:def:epr:esa:1.0:pep-2

#sts reference (in file idp.properties) = sts endpoints
sts-1=http://VM1_IP:80/axis2/services/sts
sts-2=http://VM2_IP:80/axis2/services/sts

Note to replace VM1_IP and VM2_IP by the IP addresses of the VM1 and VM2 machines and A_NEW_ID1, A_NEW_ID2 with the values defined in Step 1 above.

Step 3: replace content of the file /apps/apache-tomcat-6.0.37/webapps/catalogueClient/WEB-INF/classes/idp.properties by the following:

sts-1=STS-1(GCM SSO IDP)
sts-2=STS-2(ESA SSO IDP)

Step 4: Replace content of file /apps/apache-tomcat-6.0.37/webapps/catalogueClient/WEB-INF/classes/delegated-idp.properties by the following:

na= Not Applicable
urn\:ceos\:def\:epr\:gcm\:1.0\:sts-1 = STS-1 (GCM)

Step 5: Copy file cim-ep-client.keystore as defined in 3.3.2 to directory =/apps/conf/keystores/ on the machine as outcome of the procedure in [RD06].

4.7 Configuration

4.7.1 STS / LDAP Configuration

The configuration of STS and LDAP is explained in the user guide of HMA Security Token Service. To make the link with the keys configuration shown in section 3.4, the following statements hold.

1. The STS keystore is configured by defining KEYSTORE_LOCATION and KEYSTORE_PASSWORD parameters.

2. The STS private key, used for SAML token signature, is configured by defining AUTHENTICATION_CERTIFICATE_ALIAS and AUTHENTICATION_CERTIFICATE_PASSWORD parameters.

3. The Relying Party's public key, used to encrypt the SAML token, is configured by defining AUTHORISATION_CERTIFICATE_ALIAS parameter; this is actually the public key of the PEP that shall consume the SAML token (see section 4.7.2.2). This alias shall been defined when importing the certificate from the PEP component in the STS keystore.

4. If the STS shall not accept RST with signature from trusted clients (see 3.4.2), then the value assigned to CLIENT_CERTIFICATE_ALIASES shall be empty.
5. Otherwise, if the STS accepts RST with signature from trusted clients (see 3.4.3), then the CLIENT_CERTIFICATE_ALIASES shall contain a comma-separated list of all the aliases to get the public keys of the trusted clients (these shall be defined when importing the certificate from trusted clients in the STS keystore).

Here are the steps to create a new STS keystore and to configure the STS accordingly:

1. **Create STS keystore**

   Execute the following command:
   
   ```
   keytool -genkey -alias sts_alias -keyalg RSA -validity number_of_dates -keystore sts_keystore_filename
   ```
   
   enter new keystore password, certificate password and answer the questions.

   → this produces a local file `sts_keystore_filename`

   From this, the following STS parameters can be set: KEYSTORE_LOCATION, KEystore_PASSWORD, AUTHENTICATION_CERTIFICATE_ALIAS and AUTHENTICATION_CERTIFICATE_PASSWORD.

2. **Export certificate from PEP keystore**

   2.1 Place you in a writable directory where you can access PEP service-level keystore (see 4.7.2.2), named in the following `pep_keystore_filename`.

   2.2 Execute the following command, where `pep_alias` is the alias of the PEP certificate:
   
   ```
   keytool -export -rfc -alias pep_alias -file pep.cer -keystore pep_keystore_filename
   ```
   
   and enter keystore password associated to `pep_keystore_filename`

   → this produces a local file `pep.cer` (the certificate with PEP public key);

3. **Import certificate in STS keystore**

   Execute the command
   
   ```
   keytool -import -alias pep_alias -file pep.cer -keystore sts_keystore_filename
   ```
   
   enter keystore password of STS keystore. From this, the AUTHORIZATION_CERTIFICATE_ALIAS parameter can be set (`pep_alias`).

4.7.2 **PEP configuration**

   The PEP uses actually several keystores, which are defined at two levels:

   - **PEP-level keystore:**
1. one and only one keystore, whatever the number of registered services,
2. configured in "PEP configuration" page (see section C.3.3 for details)
3. stores public key of STS, as "trusted certificate entries",
4. used to check signatures of SAML tokens.

- **Service-level keystore(s):**
  - one keystore for each registered service,
  - configured in each "service configuration" page (see section C.4.2 for details),
  - stores private key of authorisation service, as "key entries",
  - used to decrypt SAML tokens.

There is one PEP-level keystore to set-up once for all and multiple service-level keystores to set-up for each service that has to be secured.
This is explained in the following subsections.

### 4.7.2.1 Setting PEP-Level Keystore

The set-up of PEP-level keystore has three main steps:

1. export certificate from STS keystore;
2. create a new keystore and import this certificate;
3. upload this new keystore as PEP-level keystore.

Note that these steps apply for operational environment. In a test environment, where security standards can be relaxed, the procedure can be simplified by replacing steps 1 and 2 by getting a copy of STS keystore file, with associated aliases and passwords. If such configuration is chosen, note that, due to a limitation in current version of the PEP, the STS keystore shall have certificate password identical to keystore password (see warning in 4.7.2.2).

Here are the three detailed steps, using keytool command (note that other keystore management tools can be used):

#### 1. export certificate from STS keystore

1.1 place you in a writable directory where you can access STS keystore (named in the following `sts_keystore_filename`),
1.2 execute the following command

```
keytool -export -rfc -alias sts_alias -file sts.cer -keystore sts_keystore_filename
```

and enter keystore password associated to `authentication_keystore_filename`
→ this produces a local file sts.cer (the certificate with STS public key);

2. create a new keystore and import this certificate
   execute the following command
   keytool -import -alias sts_alias -file sts.cer -keystore 
   pep_keystore_filename
   enter new keystore password and answer the questions.
   → this produces a local file pep_keystore_filename

3. upload this new keystore as PEP-level keystore
   The keystore can be configured by accessing the PEP configuration page, as explained in section C.3.3. Save the configuration when uploaded the keystore and provided the requested info.

4.7.2.2 Setting Service-Level Keystores

A keystore, containing the private key of PEP, has to be set-up on each registered service. In an architecture with one single STS, the same keystore shall be used for each service.

To create a new service-level keystore, the following command shall be executed:
keytool -genkey -keyalg RSA -alias pep_alias -keystore 
service_keystore_filename

enter new keystore password and answer the questions.
IMPORTANT WARNING: Due to a limitation in current version of PEP, the certificate password shall be the same as the keystore password.
→ this produces a local file service_keystore_filename

For each service registered in the PEP, the service-level keystore shall be set-up by accessing the service configuration page, as explained in section C.4.2. Save the configuration once uploaded the keystore and provided the requested alias and password.

Note that the public key of the service-level keystore shall be exported in a certificate and imported in the STS keystore (see 4.7.1). From that point, the STS shall be able to encrypt the SAML token so that only the PEP can decrypt it.

4.7.3 CIM-EP Client Configuration

Generally, the Client must be configured for the following before it can be used:
- The catalogue services that users can select to send a Service request, e.g. a Catalogue search request, to.
- The STS endpoints from that the Client requests users’ SAML token to insert in a Service request.
- The SSL certificate that is for HTTPs connections between the Client and the STS instance. We have to import the SSL certificate (public key) used by the Tomcat server where the STS instance runs to the truststore of the Java JRE used by the Client.

The procedures to perform these configuration work are described in detail in Annex D.5
5 USER MANUAL

This chapter provides guidelines to use the Online HMA-S Identity Management Demonstrator installation located at ESA. This installation applies the Demonstrator configuration DC3 as described in section 3.4.4 and accessible online via the following endpoints:

<table>
<thead>
<tr>
<th>Client</th>
<th>Endpoint</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>STS / IDP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STS-1</td>
<td><a href="https://VM1-IP/axis2/services/sts">https://VM1-IP/axis2/services/sts</a></td>
<td>URL where the STS-1 receives the RST requests in SOAP 1.1 format.</td>
</tr>
<tr>
<td>STS-2</td>
<td><a href="https://VM2-IP/axis2/services/sts">https://VM2-IP/axis2/services/sts</a></td>
<td>URL where the STS-2 receives the RST requests in SOAP 1.1 format.</td>
</tr>
<tr>
<td>Services</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Secured-I15</td>
<td><a href="http://VM1-IP/PEP/services/Secured-I15">http://VM1-IP/PEP/services/Secured-I15</a></td>
<td>URL where the PEP-1 receives the Catalogue service requests in SOAP 1.1 format that in turn address the CIM EP catalogue service that it protects – the Non-Secured CIM-EP service listed below in this table</td>
</tr>
<tr>
<td>Catalogue 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Secured-I15</td>
<td><a href="http://VM2-IP/PEP/services/Secured-I15">http://VM2-IP/PEP/services/Secured-I15</a></td>
<td>URL where the PEP-2 receives the Catalogue service requests in SOAP 1.1 format that in turn address the CIM EP catalogue service that it protects – the Non-Secured-I15 Catalogue listed below in this table</td>
</tr>
<tr>
<td>Catalogue 2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Online Demonstrator Access Endpoints

The following table shows the SAML attributes values corresponding the two test users setup in the Demonstrator:

<table>
<thead>
<tr>
<th>SAML Token Attribute Name</th>
<th>TestUser1</th>
<th>TestUser2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>TestUser1</td>
<td>TestUser2</td>
</tr>
<tr>
<td>c</td>
<td>BE</td>
<td>BE</td>
</tr>
<tr>
<td>o</td>
<td>Spacebel</td>
<td>Spacebel</td>
</tr>
<tr>
<td>ProjectName</td>
<td>Project1</td>
<td>Project2</td>
</tr>
</tbody>
</table>
In the following subsection we describe how to use Demonstrator via the CIM-EP client web interface. In the subsection that follows, we describe a few tests that show how to use the Demonstrator from the SOAP-UI client. This latter case allows to view the messages in raw XML format (on the SOAP-UI client GUI) exchanged between the Client and the STS, and between the Client and the PEP.

### 5.1 Using the Online the Demonstrator via CIM-EP Client

We propose to use HMA-S Catalogue Client to access the Demonstrator by performing the following steps in sequence using the values def

**STEP1:** Open the home page of the Client on your Internet browser.

*Note: Firefox version 20.1 is successfully used.*

**STEP2:** Click on link (on the top-right) Setting. Fill the form with information as provided on the columns 1,2 and 3 on Table 6, for each use case. Click Save button to save the settings.

**STEP3:** Select the Catalogue service as defined in the column 4 on the table said above. Click button Search.

**STEP4:** Observer the result for each use case following the description from the last column on the table said above.
<table>
<thead>
<tr>
<th>SSO User ID</th>
<th>SSO ID</th>
<th>Delegated SSO ID (for STS Delegation)</th>
<th>Catalogue Service</th>
<th>Is SAML token available (authenticated)</th>
<th>Authorisation</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>U01.01</td>
<td>TestUser1</td>
<td>STS-1</td>
<td>Secured-I15 Catalogue 1</td>
<td>Yes</td>
<td>Denied</td>
<td>TestUser1 is available in the LDAP directory used by the STS-1. Thus the Client succeeds to retrieve a SAML token for the user from the STS instance. The Search request is denied because the TestUser1 has SAML attribute “UserProfile” valued with “Commercial”. Note: In the Demonstrator, only requests from user TestUser2 are authorized to access the protected service.</td>
</tr>
<tr>
<td>U01.02</td>
<td>TestUser1</td>
<td>STS-2</td>
<td>Secured-I15 Catalogue 1</td>
<td>No</td>
<td>Denied</td>
<td>TestUser1 is not available in the LDAP directory used by the STS-2. Thus the Client fails to retrieve a SAML token for the user from the STS-2 instance. On the Client page, you should see the following error report after clicking button Search: “Authentication failed”. Note: the Client does not submit the Search request to the protected catalogue. It stops at the step to obtain the SAML token when encountered the failure.</td>
</tr>
<tr>
<td>U01.03</td>
<td>TestUser1</td>
<td>STS-2</td>
<td>STS-1</td>
<td>Secured-I15 Catalogue 1</td>
<td>Yes</td>
<td>Denied</td>
</tr>
</tbody>
</table>

Note: In the Demonstrator, only requests from user TestUser2 are authorized to access the protected service.
Only service request with SAML token of TestUser2 is authorised to the protected catalogue instance. The Client does not ask for SAML token as no user is applicable. However it does issue the Search request to the protected catalogue. And the request is denied by the PEP. The Client receives the following error report from the PEP:

“Policy enforcement restricts access. Authorization failed: Authorization denied
Policy exception report

The Client receives error report “Authentication failed” from the STS

TestUser3 is not available on the LDAP directory of the STS-1.
The Client receives error report “Authentication failed” from the STS

TestUser3 is not available on the LDAP directory of the STS-2.
The Client receives error report “Authentication failed” from the STS

TestUser3 is not available on the LDAP directory of the STS-1.
The Client receives error report “The specified request failed. Authentication failed” from the STS
<table>
<thead>
<tr>
<th>Case</th>
<th>User</th>
<th>STS</th>
<th>Security</th>
<th>Catalogue</th>
<th>Result</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>U02.01</td>
<td>TestUser2</td>
<td>STS-1</td>
<td>Not Applicable</td>
<td>Secured-I15 Catalogue 1</td>
<td>No</td>
<td>Denied</td>
</tr>
<tr>
<td>U02.02</td>
<td>TestUser2</td>
<td>STS-2</td>
<td>Not Applicable</td>
<td>Secured-I15 Catalogue 1</td>
<td>Yes</td>
<td>Authorised</td>
</tr>
<tr>
<td>U02.03</td>
<td>TestUser2</td>
<td>STS-1</td>
<td>Not Applicable</td>
<td>Secured-I15 Catalogue 1</td>
<td>No</td>
<td>Denied</td>
</tr>
<tr>
<td>U02.04</td>
<td>TestUser2</td>
<td>STS-2</td>
<td>STS-1</td>
<td>Secured-I15 Catalogue 1</td>
<td>No</td>
<td>Denied</td>
</tr>
<tr>
<td>U11.01</td>
<td>TestUser1</td>
<td>STS-1</td>
<td>Not Applicable</td>
<td>Secured-I15 Catalogue 1</td>
<td>Yes</td>
<td>Denied</td>
</tr>
<tr>
<td>U11.02</td>
<td>TestUser2</td>
<td>STS-2</td>
<td>Not Applicable</td>
<td>Secured-I15 Catalogue 1</td>
<td>Yes</td>
<td>Authorised</td>
</tr>
</tbody>
</table>
instances are consumable by the same PEP instance (PEP-1).

<table>
<thead>
<tr>
<th>U11.03</th>
<th>TestUser1</th>
<th>STS-1</th>
<th>Not Applicable</th>
<th>Secured-I15 Catalogue 1</th>
<th>Yes</th>
<th>Denied</th>
</tr>
</thead>
<tbody>
<tr>
<td>U11.04</td>
<td>TestUser1</td>
<td>STS-2</td>
<td>STS-1</td>
<td>Secured-I15 Catalogue 2</td>
<td>Yes</td>
<td>Denied</td>
</tr>
</tbody>
</table>

This case and the case just above present the use case where SAML tokens obtained from the STS instance (STS-1) are consumable by difference PEP instances (PEP-1, and PEP-2).

Table 6: Test Input Data
5.2 Using the Online the Demonstrator via SOAP-UI

You can use SOAP-UI client to submit a RST request or Service request to the appropriate endpoints supported by the Demonstrator. Note that, only request in SOAP 1.1 format is supported.

The current document, we describe a few tests in which we send RST requests with user name and password to the STS instances in the Demonstrator. These tests demonstrate the use cases where the STS instances work as an IDP.

Note: the use cases described in previous section are for the cases where STS instances work with external IDP.

STEP1: Install the SOAP UI on your test PC.

*Note: SOAP UI version 3.5 is successfully tested. In the following steps, the procedure are applicable for SOAP-UI 3.5. Other versions of the tool may have slight difference.*

STEP2: Create a SOAP-UI project to connect to the STS-2

2.1: Select menu File → New SOAP UI Project

2.2: Fill the project form as follows:

- Project Name: STS

Note to replace VM-2-IP by the IP address of the machine VM-1.

2.3: Click OK. This will create in the tool a new project shown on the left panel.
STEP 3: Perform test cases

3.1: Right click on menu “RequestSecurityToken” and select menu “New Request”.

3.2: Enter request name “rst-without-AppliesTo”

3.3: Replace the request content by the following:

```xml
    xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
    <SOAP-ENV:Header/>
    <SOAP-ENV:Body>
        <wst:RequestSecurityToken>
            <wst:TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1</wst:TokenType>
            <wst:RequestType>http://docs.oasis-open.org/wss/200512/Issue</wst:RequestType>
            <wsse:UsernameToken>
                <wsse:Username>TestUser2</wsse:Username>
                <wsse:Password>TestUser2</wsse:Password>
            </wsse:UsernameToken>
        </wst:RequestSecurityToken>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.4: Click on icon green on the top-left of the request window to send the RST request to the STS-1 instance.

The STS-1 returns the SAML token for user TestUser1 as shown in the above screenshot.

3.5: Modify the request to change password to “abc”. Click the green icon to submit the request. The STS-1 returns error report.
3.6: Right click on the menu on the left “rst-without-AppliesTo” and select Clone Request.

3.7: Enter request name “rst-with-AppliesTo”

3.8: Replace the request by the following:

```xml
    xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
    xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512/"
  <SOAP-ENV:Header/>
  <SOAP-ENV:Body>
    <wst:RequestSecurityToken>
      <wst:TokenType>
        http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1
      </wst:TokenType>
      <wst:RequestType>
        http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
      </wst:RequestType>
      <wsp:AppliesTo
        <EndPointReference
          xmlns="http://www.w3.org/2005/08/addressing">
          <Address>urn:ceos:def:epr:gcm:1.0:pep-1</Address>
        </EndPointReference>
      </wsp:AppliesTo>
      <wsse:UsernameToken>
        <wsse:Username>TestUser2</wsse:Username>
        <wsse:Password>TestUser2</wsse:Password>
      </wsse:UsernameToken>
    </wst:RequestSecurityToken>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

3.9: Click the green icon on the request page to issue the RST request. The client receives the SAML token
3.10: Change AppliesTo to "urn:ceos:def:epr:gcm:1.0:abc". Submit the request. The Client shows error report

3.11: Repeat 3.6 and 3.10 to create a new request named "rst-with-ApppliesTo-and-with-DelegateTo".

3.12: Replace the request content by the following

```xml
<SOAP-ENV:Body>
<br:RequestSecurityTokenType xmlns:br="http://docs.oasis-open.org/ebxml-msg/brs/2005/07/brs-securetrust-1.0.xsd">
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```
3.13: Click the green icon to submit the request. The Client receives the SAML token as the RST request is destined to the STS-1 and the user TestUser1 is authenticated by this STS.

3.14: Replace TestUser1 by TestUser2 in the request. Issue the request. The Client shows error report.
3.15: On the request page of STEP 3.13 above. Replace the IP address in the request address by the IP address of the machine VM-1. This is to prepare for sending the RST request with DelegateTo parameter to the STS-1 service, which is a deleted STS.

3.16: Click the green icon to submit the RST request to the STS-1. The STS-1 returns the SAML token in clear as defined in [AD05], see the Figure 6.
Annex A: LINUX INSTALLATION

A.1 Create VM from Kickstart File

The file below is the Kickstart file used to setup the Red Hat Enterprise Linux 6.3 (64bit) installation on the Virtual Machine for the Demonstrator. This version of the Operating System is compliant with the applicable EOP-G Baseline [AD04].

```bash
Using username "root".
Last login: Fri Apr 19 10:58:05 2013 from h-notejlm.spb.spacebel.be
Install from kickstart file hma-s on 2013-03-26
[root@hma-s ~] # more /etc/redhat-release
Red Hat Enterprise Linux Server release 6.3 (Santiago)
[root@hma-s ~] # uname -im
x86 64 x86_64
[root@hma-s ~] #

# Kickstart file automatically generated by anaconda.
#version=DEVEL
install
cdrom
lang en_US.UTF-8
keyboard be-latin1
network --onboot yes --device eth0 --bootproto dhcp --noipv6
rootpw --iscrypted $1$Ne/ICc9q8noHCWTAGzHNI0eUlhqgkq/
# System services
services --disabled="autofs,sendmail,rhnsd,cups,netfs,nfslock,kudzu,crond,portmap"
# Reboot after installation
reboot
firewall --port=22:tcp --service=http --service=ssh --service=smtp
authconfig --enableshadow --enablemd5
selinux --disabled
timezone --utc Europe/Brussels
bootloader --location=mbr --driveorder=sd0 --append="crashkernel=auto rhgb rhgb quiet"
quiet" --password=$1$VVPB9Dhs$JsCpc1EZx.1ugExvatawI0
pciE9x.lugExvatawIO
# The following is the partition information you requested
# Note that any partitions you deleted are not expressed
# here so unless you clear all partitions first, this is
# not guaranteed to work
#clearpart --linux --drives=sd0

#part /boot --fstype=ext3 --size=150
#part swap --size=3000
#part pv.008003 --grow --size=1
#volgroup vg_root --pesize=32768 pv.008003
#logvol /apps --fstype=ext4 --name=lv_apps --vgname=vg_root --size=8192
#logvol /home --fstype=ext4 --name=lv_home --vgname=vg_root --size=2048
#logvol /opt --fstype=ext4 --name=lv_opt --vgname=vg_root --size=4096
#logvol / --fstype=ext4 --name=lv_root --vgname=vg_root --size=11264
#logvol /var --fstype=ext4 --name=lv_var --vgname=vg_root --size=4096
%packages
```
@Base
@Core
@base
@core
device-mapper-multipath
fipscheck
neon
openldap-clients-2.4.23-26.el6
openldap-servers-2.4.23-26.el6
sgpio
subversion
-amtu
-bluez-utils
-jwhois
-rp-pppoe
-rsh
-stunnel
-telnet
-vnc-server
-words
-ypbind

%end

%post --nochroot --logfile /mnt/sysimage/root/ks-post.log

echo "######################################################################"
echo "# Running Post Configuration   #"
echo "######################################################################"

# prevent future yum updates pulling down & install new kernels (and breaking VMware &
video drivers).
echo "exclude=kernel*" >> /mnt/sysimage/etc/yum.conf
# custome /etc/issue banner
echo "Warning! This is a private system. Unauthorized access to or use of the system is
strictly prohibited." >> /mnt/sysimage/etc/issue

%end

A.2 Create VM from OVF Template

This section describes steps to deploy a VM onto Host using the exported OVF template.

STEP 1: Login into the VMware ESXi server using the vSphere Client application
STEP 2: Select the host node and then click on menu File -> Deploy OVF Template
STEP 3: Click button Browser and select the file .ovf
STEP 4: Click button Next, then Next. Enter a name for the VM node registered on Host
STEP 5: Click button Next. Select the storage to store the virtual machine files on the host.
STEP 6: Click button Next. Select “Thin Provision” as the Disk Format.
STEP 7: Click button Next. Select the network mapping for OVF template.
STEP 8: Click button Next. Click button Finish to start deploying the selected OVF template.
STEP 9: Power on the server.
STEP 10: Select tab “Console” on the vSphere client.
STEP 11: When prompted for login via the Console on STEP 10, login with root user, password "tonton". Note to change this password to a new password.

STEP 12: On the terminal, as result of previous step, run the following command:

```
#system-config-network
```

STEP 13: Press Enter (to select “Device configuration”), then Enter to go to the window “Network Configuration”.

![Network Configuration](image)

Enter the IP address of the server for “Static IP”; The Netmask and IP address of the default gateway of the network where the server is connected to.

When complete, select button “Ok”, then button “Save”.

STEP 14: Select menu “DNS Configuration”. Then Enter to go to the window “DNS Configuration”

![DNS Configuration](image)

STEP 15: Enter the full qualified domain name of the server for “Hostname”. For example hma-s.spacebel.be

STEP 16: Enter proper Search values.

STEP 17: Select button Ok, then button “Save&Quit”.

STEP 18: Reboot the server.

STEP 19: If the machine is not reachable via ssh, perform the following fix:

1) Open following file which holds record of the NIC templates

   `vi /etc/udev/rules.d/70-persistent-net.rules`

   The above file should contain similar records as below:

   ```
   # PCI device 0x15ad:0x07b0 (e1000)
   ```
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="00:27:3e:ae:00:1a",
ATTR{type}=="1", KERNEL=="eth\", NAME="eth0"

# PCI device 0x15ad:0x07b0 (e1000)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="00:27:3e:ae:00:1b",
ATTR{type}=="1", KERNEL=="eth\", NAME="eth1"
2) Remove previous eth entries and change the last entry NAME to eth0
# PCI device 0x15ad:0x07b0 (e1000)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*",
ATTR{address}=="00:27:3e:ae:00:1b",
ATTR{type}=="1", KERNEL=="eth\", NAME="eth0"
3) Save the file.
4) Open following file and modify the mac address
/etc/sysconfig/network-scripts/ifcfg-eth0
5) Save the file.
6) Reload udev configuration for the changes we made to reflect
start_udev
7) Restart network service
service network restart
Annex B: USER GUIDE - SECURITY TOKEN SERVICE

B.1 Introduction

The Security Token Service (STS) is a Web service that implements, for the SOAP binding, the authentication use cases 1, 2, 3 and 4 defined in the OGC 07-118 version 1.1. The implementation has used the following software and libraries:

- LDAP version 3 directory service
- OpenSAML library version 1.1
- Axis2 version 1.4 (web archive distribution package)
- Apache Tomcat (6.0 or higher)
- Java JDK (or JRE) version 7 or higher.
- Ant build script tool version 1.6.2 or higher.

In this document, you can find the manuals for:

- installing the Security Token Service.
- integrate the service with authorization service(s).

B.2 Install

The HMA Security Token Service can be installed using different methods depending on the availability of the hardware and software. The simplest method is to install the service onto an existing server that already has:

- a tomcat server installed with Axis2 web application that allows web service access (SOAP/HTTP(S)) to the Security Token Service.
- a connection to an LDAP version 3 directory service to authenticate users' credentials and retrieve users' information.

In this case, the installation procedure is as follows:

- Deploy the Security Token Service by using an Axis2 deployable archive file (sts.aar from the binary package sts-bin-v2.5.zip).
- Reconfigure the LDAP directory to support the data schema/attributes required by the Security Token Service (see the section "Configure Security Token Service" below).

The binary package sts-bin-v2.5.zip can be obtained from the Google code website by checking out the source code from https://hma-security-token-service.googlecode.com/svn/tags/release-v2.5 to a local directory of a machine with Ant version 1.6 or higher and Java 1.7 installed. Then go to the local directory to execute command “ant dist”. This command generates under the local directory a directory named “dist” that includes the sts-bin-v2.5.zip package file.
The following sections describe a complete procedure that allows to install the Security Token Service on an empty machine with only operating system software installed. The procedure has been tested successfully on the following system and software files:

System:
- An Intel Xenon PC with 1.7 GHz CPU, 1 GB RAM, 50 GB, with Red Hat Enterprise Linux Server release 6.3 installed.

Software:

<table>
<thead>
<tr>
<th>Software</th>
<th>Installer binary package file</th>
<th>Project website and/or location where the software binary package can be obtained</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenLDAP version 2.4.23-26</td>
<td>openldap-2.4.23-26.el6.x86_64.rpm openldap-servers-2.4.23-26.el6.x86_64.rpm openldap-clients-2.4.23-26.el6.x86_64.rpm</td>
<td><a href="http://www.openldap.org">http://www.openldap.org</a> The RPM package is available on the software CDROM &quot;Red Hat Enterprise Linux Server release 6.3&quot;.</td>
</tr>
<tr>
<td>Oracle/Sun JDK version 1.7.0_45</td>
<td>jdk-7u45-linux-x64.rpm</td>
<td><a href="http://www.oracle.com/technetwork/java">http://www.oracle.com/technetwork/java</a></td>
</tr>
<tr>
<td>Apache Ant version 1.6.2</td>
<td>apache-ant-1.6.2-bin.zip</td>
<td><a href="http://archive.apache.org/dist/ant/binaries/apache-ant-1.6.2-bin.zip">http://archive.apache.org/dist/ant/binaries/apache-ant-1.6.2-bin.zip</a></td>
</tr>
<tr>
<td>Apache Axis2 version 1.4</td>
<td>axis2-1.4.zip</td>
<td><a href="http://archive.apache.org/dist/ws/axis2/1_4/axis2-1.4.zip">http://archive.apache.org/dist/ws/axis2/1_4/axis2-1.4.zip</a></td>
</tr>
<tr>
<td>Security Token Service version 2.5</td>
<td>sts-bin-v2.5.zip</td>
<td><a href="https://drive.google.com/file/d/0B4WHm-d_5aovLVNJUW16RmJ1UGc/view?usp=sharing">https://drive.google.com/file/d/0B4WHm-d_5aovLVNJUW16RmJ1UGc/view?usp=sharing</a></td>
</tr>
</tbody>
</table>

The procedure includes:
- Install OpenLDAP
- Install Java JDK
- Install Apache Tomcat
- Install Axis2 web application
- Install HMA Security Token Service
- Test the installation.

Note: the procedure's steps should be executed step by step and in the order they appear in this document. In the steps' description, $software is used to reference to a local directory on the system where the software binary package files listed on the table above are put.

**B.2.1 Install OpenLDAP**

**Step 1: Install OpenLDAP**

If the software is not yet installed during the OS system installation, log in to the system under root and go to directory $software to execute the following command:

```
root> rpm –ihv openldap-2.4.23-26.el6.x86_64.rpm
root> rpm –ihv openldap-servers-2.4.23-26.el6.x86_64.rpm
root> rpm –ihv openldap-clients-2.4.23-26.el6.x86_64.rpm
```

Follow the command instruction to have the software installed completely.

Run the following command to have the LDAP service started together with the system boot:

```
root> chkconfig slapd on
```

Start the directory service for the first time for test:

```
root> service slapd start
```

The last command should print a log "Starting slapd: [ OK ]" if the installation is successful.

Note: the current step installs OpenLDAP software files to directory /etc/openldap.

**Step 2: Configure the LDAP directory service**

The present step allows to specify the LDAP directory service, including the specific attributes needed in the LDAP directory, by means of LDAP schema file(s). The schema file(s) are to be defined according to the needs of the application using the LDAP directory, since OGC 07-118 version 0.1.0 does not specify a list of attributes (contrarily to version 0.0.4). Note that the/inetOrgPerson schema (RFC 2798) is already defined in OpenLDAP; provided that/inetOrgPersonuser attributes are sufficient, the definition of a new schema is not required.

The following instructions include the hmaOpenLDAP.schema into the software installation, as an example of extending LDAP schema for new attributes and object types that are not yet defined/available in RFC 2798. This hmaOpenLDAP schema registers a new object "HMAUser" that has sufficient attributes to fulfill the LDAP-SAML mapping, which produces a SAML token as per the example on table 2 of OGC -07-118r9 Issue 1.1.

For more information about OpenLDAP extending schema, see http://www.openldap.org/doc/admin24/schema.html

As root on the system:
- copy file hmaOpenLDAP.schema (from package sts-bin-v2.5.zip, at path sts.aar/ldap) to directory /etc/openldap/schema
- copy file slapd.conf (from package sts-bin-v2.5.zip, at path sts.aar/ldap) to directory /etc/openldap.
- edit file slapd.conf (copied above) to apply your system information, as follows:
  - change suffix from "dc=spacebel,dc=be" to a new LDAP suffix that fits your case.
  - change rootdn from "cn=ldapRoot,dc=spacebel,dc=be" to another value of your system, where
    ldapRoot is the directory admin account name and "dc=spacebel,dc=be" must be set to the new
    LDAP suffix said above.
  - change rootpw from secret to a new password for the directory admin account.

When finish, go to directory /etc/openldap to run the following commands to apply the change made above:
Stop the slapd service if it is running:
root> service slapd stop
Apply the configuration:
root> mv /etc/openldap/slapd.d /etc/openldap/slapd.d-bk
root> mkdir /etc/openldap/slapd.d
root> slaptest -f /etc/openldap/slapd.conf -F /etc/openldap/slapd.d
root> chown -R ldap:ldap /var/lib/ldap
root> chown -R ldap:ldap /etc/openldap/slapd.d
Start slapd service for testing:
root> service slapd start
The last command should print "Starting slapd: [ OK ]" if the configuration is successful.

Step 3: Initiate the HMA user directory tree
This step is to create the base entries for appending the HMA user data and to insert initial user entries (for
testing purposes):
Copy file testUsers.ldif (from package sts-bin-v2.5.zip, at path sts.aar/ldap) to directory /etc/openldap.
Go to directory /etc/openldap to run the following command:
ladvadd -x -D "cn=ldapRoot,dc=spacebel,dc=be" -w secret -f testUsers.ldif
Note to replace the values in the bold texts with the information for your system: "cn=ldapRoot,dc=spacebel,dc=be" is replaced by the rootdn applied in Step 2 above; "secret" is replaced by the rootpw applied in Step 2 above.
For more information, refer to the product documentation at http://www.openldap.org/doc/admin24/index.html

B.2.2 Install JDK

Sign in the installing machine under root user.
Change command prompt to directory $softwareDir, and run the following command:
root>rpm -Uvh jdk-7u7-linux-x64.rpm
Run this command to test the installation
root> java -version.
The last command should prints "java version "1.7.0_45"..." if the installation is successful.
B.2.3 Install Tomcat

Step 1: install software file

As root, expand the zip file $softwareDir/apache-tomcat-6.0.37.tar.gz to directory on the machine /apps/. This will create a new directory "apache-tomcat-6.0.37" under the /apps directory.

This new directory "apache-tomcat-6.0.37" is referenced later on as $TOMCAT_HOME.

Step 2: Configure the default HTTP ports

Go to directory $TOMCAT_HOME to run the following commands to configure HTTP access port

root>sed -i.bk "s/8080/80/g" conf/server.xml
root>sed -i "s/8443/443/g" conf/server.xml

Step 3: Configure SSL keys HTTPs connection

And run the following commands to create a self-signed key to be used for HTTPs connections supported by this Tomcat server.

root>mkdir -p /apps/conf/keystores
root>/usr/java/default/bin/keytool -genkey -alias tomcat -keyalg RSA -validity 3650 -keystore /apps/conf/tomcat.keystore

Answer the question prompted by the last command, to have the key generated. When prompted for the password, enter "changeit".

You can apply other password instead. However in that case, you need to set it to the tomcat server.xml configuration.

Edit the conf/server.xml file to apply the keystore generated above, as follows:

Replace the following text in the file:

<!--
<Connector port="443" protocol="HTTP/1.1" SSLEnabled="true"
maxThreads="150" scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS" />
-->

by the following new text:

<Connector port="443" protocol="HTTP/1.1" SSLEnabled="true" maxThreads="150"
scheme="https" secure="true"
clientAuth="false" sslProtocol="TLS" keystoreFile="/apps/conf/keystores/tomcat.keystore"
keystorePass="changeit" />

Step 4: Register tomcat service to the system
Create new file /etc/init.d/tomcat with the following content:

```bash
# This is the init script for starting up the Tomcat server
#
# chkconfig: 345 91 10
# description: Starts and stops the Tomcat daemon.
#
# Source function library.
. /etc/rc.d/init.d/functions

# Get config.
. /etc/sysconfig/network

# Check that networking is up.
[ "${NETWORKING}" = "no" ] && exit 0

tomcat=/apps/apache-tomcat-6.0.37
startup=$tomcat/bin/startup.sh
shutdown=$tomcat/bin/shutdown.sh
start(){
    echo -n "$Starting Tomcat service: "
    $startup
    RETVAL=$?
    echo
}

stop(){
    echo -n "Stopping Tomcat service: "
    $shutdown
    RETVAL=$?
    echo
}

restart(){
    stop
    sleep 30
    start
}

# See how we were called.
case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
```
status)
# This doesn't work ;)
    status tomcat
    ;;
    restart)
    restart
    ;;
*)
    echo "$Usage: $0 {start|stop|status|restart}"
    exit 1
esac
    exit 0

### B.2.4 Install Axis2

Copy the file `$softwareDir/axis2.war` to the Tomcat deployment directory `$TOMCAT_HOME/webapps`

Edit file `$TOMCAT_HOME/webapps/axis2/WEB-INF/conf/axis2.xml` to set "true" value for the parameter "disableSOAP12".

```xml
<parameter name="disableSOAP12" locked="true">true</parameter>
```

### B.2.5 Install HMA Security Token Service

Change the command prompt to the directory `$softwareDir` to run the following commands:

- `unzip sts-bin-v2.5.zip`

A directory named "sts.aar" will appear under the directory `$softwareDir`.

Copy the directory "sts.aar" to the Axis2 service home directory `$TOMCAT_HOME/webapps/axis2/WEB-INF/services/

**authentication-service.properties** file

Update the file `$TOMCAT_HOME/webapps/axis2/WEB-INF/services/sts.aar/authentication-service.properties` to have the values of your system, as described in the following table:

<table>
<thead>
<tr>
<th>LDAP_URL</th>
<th>URL to connect to the LDAP directory, e.g, ldap://localhost:389 (see the ldap configuration in section &quot;Install LDAP&quot;).</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDAPSearchContext</td>
<td></td>
</tr>
</tbody>
</table>
The root position where ldap searching action is started from, e.g., dc=spacebel,dc=be (see the ldap configuration in section "Install LDAP").

**LDAPPrincipal**
The string used to connect to the LDAP under the admin user, e.g. cn=ldapRoot,dc=spacebel,dc=be

**LDAPCredentials**
The password of the admin user, e.g. secret

**KEYSTORE_LOCATION**
The location of the keystore file. The Security Token Service include a keystore for test and demonstration purpose. This demonstration keystore is located at: $TOMCAT_HOME/webapps/axis2/WEB-INF/services/sts.aar.aar/keystore/demokeystore. The keystore file contains a pair of public and private keys with alias 'authenticate'. See the following section for more information about the SSL keys used by the Security Token Service.

**KEYSTORE_PASSWORD**
Keystore password

**AUTHENTICATION_CERTIFICATE_ALIAS**
Alias of certificate containing the private key of the STS, used to sign SAML tokens

**AUTHENTICATION_CERTIFICATE_PASSWORD**
Password associated to AUTHENTICATION_CERTIFICATE_ALIAS

**AUTHORISATION_CERTIFICATE_ALIAS**
Alias of certificate containing the public key of the authorisation service, used to encrypt the SAML token (see 3.1)

**CLIENT_CERTIFICATE_ALIASES**
Coma-separated list with aliases of certificate containing the public keys of trusted STS clients, used to verify RST signature (used only for RST with signature) (see 3.2); if no client is trusted for issuing RST with signature, then the value shall be empty

**REGISTRATION_STATE_ATTRIBUTE_NAME**
Optional; if set, then an extra check is made on RST before issuing SAML token: the specified user shall have a LDAP attribute named as the value of REGISTRATION_STATE_ATTRIBUTE_NAME and this attribute shall have the same value as REGISTRATION_STATE_ATTRIBUTE_VALUE; if this checks fails, then the RST fails

**REGISTRATION_STATE_ATTRIBUTE_VALUE**
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISSUER_NAME</td>
<td>Name of the SAML token issuer</td>
</tr>
<tr>
<td>SAML_TOKEN_EXPIRY_PERIOD</td>
<td>The period during which the SAML token is still valid.</td>
</tr>
<tr>
<td>SAML_ASSERTION_ID_PREFIX</td>
<td>Prefix of the SAML assertion identifier.</td>
</tr>
<tr>
<td>SAML_ASSERTION_ELEMENT</td>
<td>Optional, used for SAML token backward compatibility with OGC 07-118 v0.0.4 or before; it impacts the SAML token returned in <code>&lt;wst:RequestedSecurityToken&gt;</code> of RSTR; if set to True, then the SAML token is an element <code>&lt;Assertion xmlns=&quot;http://earth.esa.int/um/eop/saml&quot;&gt;</code>, parent of a <code>&lt;xenc:EncryptedData&gt;</code> element, with &quot;Content&quot; encryption type; otherwise, the SAML token is directly the <code>&lt;xenc:EncryptedData&gt;</code> element, with &quot;Element&quot; encryption type; this is the format defined in OGC 07-118 v0.0.5 or after.</td>
</tr>
<tr>
<td>SAML_OLD_SIGNATURE</td>
<td>Optional, used for SAML token backward compatibility with OGC 07-118 v0.0.5 or before; if set to True, then the canonicalization method algorithm used in signature is &quot;<a href="http://www.w3.org/TR/2001/REC-xml-c14n-20010315">http://www.w3.org/TR/2001/REC-xml-c14n-20010315</a>&quot;, the signature's URI reference is empty and the certificate is attached to the token; otherwise, the canonicalization method algorithm used in signature is &quot;<a href="http://www.w3.org/2001/10/xml-exc-c14n#">http://www.w3.org/2001/10/xml-exc-c14n#</a>&quot;, the signature's URI reference refers to SAML Assertion and no certificate is attached; this is the format defined in OGC 07-118 v0.0.6 or after.</td>
</tr>
<tr>
<td>SAML_TOKEN_ENCRYPTION_ACTIVE</td>
<td>Optional, flag indicating whether the SAML token shall be delivered encrypted or not. The value True shall be put in any operational system supposed to comply with OGC 07-118. The value False is mainly dedicated to testing purpose. If missing, then True is assumed.</td>
</tr>
<tr>
<td>LOG4J_CONFIG_LOCATION</td>
<td></td>
</tr>
</tbody>
</table>
The location of the Log4j configuration file, e.g.
$TOMCAT_HOME/webapps/axis2/WEB-INF/services/sts.aar.aar/authentication-service-log4j.properties

LOCAL_STS_URN

Optional. used for STS federation only. It is an arbitrary URN identifying the present STS instance. If set, the STS as the ability to be federated, i.e. to receive RST from another STS. Any RST, containing an element <DelegateTo> having a URN content equal to LOCAL_STS_URN is considered as coming from a federating STS; in that case the SAML token is returned ensigned and unencrypted. Note that this URN shall be declared also in the federated_sts.properties file of the federating STS (see below)

federated_sts.properties file

If the STS is a federating STS, i.e. it shall be able to forward RST to other STS, then the following file shall be filled in:

$TOMCAT_HOME/webapps/axis2/WEB-INF/services/sts.aar.aar/federated_sts.properties

(if the STS is not federating, then the file can be empty or even missing). The purpose of this file is to map the STS URN present in <DelegateTo> element of the RST, if any, to the actual URL of the corresponding federated STS. The format is a set of lines with equalities

delegated_sts_urn = delegated_sts_url

Not that the colon character in the urn shall be escaped with a backslash character. For instance,

urn:\ceos\:def\:epr\:spacebel\:1.0\:federated-sts = http://localhost:8080/axis2/services/sts

authentication-service-log4j.properties file

Update the parameter "log4j.appender.logfile.File" in the file /stsHome/apache-tomcat-5.5.27/webapps/axis2/WEB-INF/services/sts.aar.aar/authentication-service-log4j.properties to point to the service log file, e.g.: /var/log/sts.log

B.2.6 Test

This section describes steps to make the first test to verify if the installation done in the sections above succeeded.

Step1: prepare test data
Connect to the LDAP directory server to import the test user profile using the input data from the file
/authenticatorServiceHome/apache-tomcat-5.5.27/webapps/axis2/WEB-INF/services/authentication_v2.4.aar/ldap/testUsers.ldif

If the import succeeds, there are two test users in the LDAP directory with credentials
"TestUser1/TestUser1 and TestUser2/TestUser2".

Step 2: make a SOAP call to the service operation "RequestSecurityToken".

Using a SOAP client, such as TCP monitor to send the following SOAP message to the service at
http://IP:PORT/axis2/services/sts

Note to replace the IP and PORT with your system values.

SOAP request message:

```xml
<soapenv:Envelope
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <wst:RequestSecurityToken
      xmlns:wst="http://docs.oasis-open.org/wss/sx/sx-wssecurity-secext-1.0.xsd"
      xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-wssecurity-secext-1.0.xsd">
      <wst:TokenType>
        http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1
      </wst:TokenType>
      <wst:RequestType>
        http://docs.oasis-open.org/wss/sx/sx-ws-security-secext-1.0.xsd">
          <wsse:UsernameToken
            xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-wssecurity-secext-1.0.xsd"/>
            <wsse:Username>TestUser1</wsse:Username>
            <wsse:Password>TestUser1</wsse:Password>
          </wsse:UsernameToken>
        </wst:RequestSecurityToken>
      </soapenv:Body>
    </soapenv:Envelope>
```

The service should return a SOAP response message that includes the SAML token representing the user TestUser1.
B.3 Configure Security Token Service

B.3.1 Integrate to Authorisation Services

This section describes steps to integrate the Security Token Service into an authorization service, e.g. a Policy Enforcement Point - PEP (as the consumer of this Security Token Service).

The SAML token returned by this STS is in encryption and it will be decrypted by the authorisation service during the authorisation processing. To make the encryption and decryption works each other, the Security Token Service needs to use the certificate that is in pair with the private key of the authorization service.

The following steps will make the two services integrated:

Step 1: Export the certificate of the authorisation service (public key only).

The following Java keytool command can be used:

```
keytool -export -rfc -alias pep-alias -file pep.cert -keystore pep_keystore_filename
```

Step 2: Import the certificate (obtained in step 1) into the Security Token Service's keystore under an alias with name equals to the value of the property "AUTHORISATION_CERTIFICATE_ALIAS" (see configuration file $TOMCAT_HOME/webapps/axis2/WEB-INF/services/sts.aar/authentication-service.properties).

The following Java keytool command can be used:

```
keytool -import -alias pep-alias -file pep.cert -keystore sts_keystore_filename
```

Step 3: Restart STS

B.3.2 Register a trusted client on STS

This section describes steps to register a trusted client, so it can issue valid RST with signature to STS. This is the case for a STS that is not the IDP server. For this, a RST request does not include user password but the user id and the signature of the trusted client. For the latter purpose, the STS shall verify the signature thanks to the registered public key of the trusted client in its keystore. The registration of a trusted client on STS therefore boils down to register the public key of trusted client in the STS keystore. This is done with the following steps:

Step 1: Export the certificate of the trusted client (public key only).

The following Java keytool command can be used:

```
keytool -export -rfc -alias <client-alias> -file client.cert -keystore client_keystore_filename
```

Step 2: Import the certificate (obtained in step 1) into the STS keystore under a given alias A

The following Java keytool command can be used:

```
keytool -import -alias <client-alias> -file client.cert -keystore sts_keystore_filename
```

Step 3: Add alias <client-alias> in the coma-separated list defined in the property "CLIENT_CERTIFICATE_ALIASES" (see configuration file $TOMCAT_HOME/webapps/axis2/WEB-INF/services/sts.aar/authentication-service.properties)

Step 4: Restart STS
To unregister a trusted client on STS, the following steps shall be done:

Step 1: Remove the alias corresponding to the client from the coma-separated list defined in the property "CLIENT_CERTIFICATE_ALIASES" (see configuration file $TOMCAT_HOME/webapps/axis2/WEB-INF/services/sts.aar/authentication-service.properties)

Step 2: Remove the corresponding certificate from STS keystore (this step is optional)

The following Java keytool command can be used:

    keytool -delete -alias <client.alias> -keystore sts.keystore_filename

Step 3: Restart STS

B.3.3 Change LDAP Directory Service

To change to a new LDAP directory service (used by the Security Token Service), update the following parameters in the software configuration file ($TOMCAT_HOME/webapps/axis2/WEB-INF/services/sts.aar/authentication-service.properties):

- LDAPURL
- LDAPSearchContext
- LDAPPrincipal
- LDAPCredentials

B.3.4 Change Keys (Keystore)

The default configuration of the Security Token Service applies the following keys and certificates (stored in the file $TOMCAT_HOME/webapps/axis2/WEB-INF/services/sts.aar/keystore/default.keystore):

- Authentication certificate (via alias defined by property AUTHENTICATION_CERTIFICATE_ALIAS): this certificate is included in a SAML token (used to verify the digital signature)
- Authentication private key (via alias defined by property AUTHENTICATION_CERTIFICATE_ALIAS): this key is used to sign SAML tokens.
- Authorisation certificate (via alias defined by property AUTHORISATION_CERTIFICATE_ALIAS): this certificate is used to encrypt/cipher SAML tokens.

Those keys and certificates are generated using the following Ant script function (that uses Java JDK 1.7.x keystore utility):

<target name="createDemoKeystore">
    <mkdir dir="${build.dir}/${aar.file}/keystore"/>
    <property name="file" value="${build.dir}/${aar.file}/keystore/default.keystore"/>
</target>
<property name="cert.file" value="${build.dir}/${aar.file}/keystore/default.cert"/>
<property name="alias" value="urn:ceos:def:epr:spacebel:1.0:sts"/>
<property name="pass" value="changeit"/>

<!-- valid for 10 years-->
<property name="validity" value="3650"/>

<delete file="${file}"/>
<genkey alias="${alias}" storepass="${pass}" keyalg="RSA" keystore="${file}" validity="${validity}">
  <dname>
    <param name="CN" value="HMA-S"/>
    <param name="OU" value="Space"/>
    <param name="O" value="Spacebel"/>
    <param name="C" value="BE"/>
  </dname>
</genkey>
<exec executable="keytool">
  <arg value="-export"/>
  <arg value="-alias"/>
  <arg value="${alias}"/>
  <arg value="-rfc"/>
  <arg value="-file"/>
  <arg value="${cert.file}"/>
  <arg value="-keystore"/>
  <arg value="${file}"/>
  <arg value="-storepass"/>
  <arg value="${pass}"/>
</exec>
<echo message="Generated keystore at ${file}"/>
</target>

To change the default keystore configuration, follows the following steps:

1. update the Security Token Service configuration file for the properites defining alias to retrieve the required keys and certificate mentioned above.
2. update the configuration for the properties defining the keystore password and the private key password.
3. update the configuration for the property defining the location of the keystore file.

B.3.5 Configure User Data Included in SAML Token

The service allows to change names of the SAML attributes (in the SAML tokens) by updating the mapping file ($TOMCAT_HOME/webapps/axis2/WEB-INF/services/sts.aar/saml-ldap-attributes-mapping.properties). This file uses Java properties file's format to define the mapping as follows:
- The property name is the name of LDAP attributes of the user profile.
- The property value is the name of SAML attributes of the SAML token.

The LDAP attribute name present in the first association shall be an identifier of the user. This attribute will be used by the STS to build up the SAML token, in order to set the NameIdentifier element of the subject of NamesIdentifier in AuthenticationStatement and AttributeStatement.
Annex C: USER GUIDE - POLICY ENFORCEMENT POINT (PEP)

C.1 Introduction

The Policy Enforcement Point (PEP) is a Web application allowing to wrap a WEB service with a security layer thus granting access only to authenticated (and possibly authorized) users.

Setting the security layer has no impact on the service implementation.

C.2 Install the PEP

This section provides the requirements and the procedure to install the PEP.

C.2.1 Installation requirements

No special requirements is set on the hardware needed for installing the PEP, unless those dictated by the software packages it depends on, and which are listed in the following paragraph.

The software required for installing the PEP is the following:

- Java 6 (http://www.oracle.com/technetwork/java/javase/overview/index.html)

The installation of these packages can be found on the linked sites; the procedure follows the line guides described in the previous annex.

C.2.2 Installation procedure

The PEP can be simply deployed on an host by copying its war file (PEP.war) under the /webapps folder of the tomcat instance. Tomcat will automatically deploy the application. Alternatively, you can deploy it by using the graphical user interface provided by the Tomcat Manager (see the tomcat documentation).

Once deployed, the java environment needs to be configured; indeed, the PEP installation uses a dedicated java library from the Bouncy Castle organization (http://www.bouncycastle.org): this library provides cryptography extensions to the Java default installation. To enable the bouncy castle capabilities update the "java.security" file which can be found in JRE's lib/security directory as follows:

- add the following line: security.provider.X=org.bouncycastle.jce.provider.BouncyCastleProvider as the last line of the file, where X has to be replaced by the progressive number according to the current security providers already listed.

The following figure shows the “java.security” file with the bouncy castle line added.
# Configure the PEP

The configuration of the PEP is made using its GUI.

- The reference browser for accessing the PEP GUI is Mozilla Firefox.
- The URL to access the PEP GUI, is \texttt{http://host:port/PEP/} where “host” and “port” shall be provided according to the tomcat configuration (see the related documentation).

The initial page of the PEP GUI provides a login form to enter the administrator credentials. By default, the credentials of the administrator are:

- Username: admin
- Password: admin

The credentials are provided in the file “adminUsers.xml” at the following path:

\texttt{$CATALINA\_HOME/webapps/PEP/WEB-INF/xml}

where \texttt{$CATALINA\_HOME} is the installation directory of tomcat on the deployment machine.

The configuration windows of the PEP can be launched by selecting the "Start" button and clicking on the “PEP Services Manager” \texttt{\rightarrow} “Config” entry:
Additionally, the "PEP Configuration" window is automatically displayed the first time the user access the PEP GUI.

The PEP Configuration provides the following entries (see figure in the next sub section):

- Net Configuration
- Log Settings
- General Security Configuration
- Contact Information

### C.3.1 Net Configuration

The "Net Configuration" tab allows to set the following parameters (the figure shows the default "Net Configuration" parameters):
Figure 9  PEP Configuration - Network

Endpoint Configuration

- **PEP Endpoint address**: It indicates the host name or IP address that has to be used by the client to connect to the PEP. It can be the address of the machine if it is directly available or the address of a firewall that protects the PEP host.

- **Port Number**: It indicates the port number to be used by the client to connect to the PEP.

- **SSL Port Number**: It indicates the port number to be used by the client to connect to the PEP in a secure way.

Proxy Settings

Note: These parameters have to be filled only if the local network uses a proxy to route outgoing HTTP messages.

- **Host name**: The host name the proxy is running on.

- **Port Number**: The port number (of the proxy host) the proxy is listening on.

C.3.2 Log Settings

The "Log Settings" tab allows to set the following parameters:
- **Logging Level**: The level of the log messages to be visualized in the PEP Log window.
- **Log Directory**: The directory on the local machine where the PEP logs shall be stored.
- **Log File Size**: The size (KB) of the log files.

![PEP Configuration - Logging](image)

**Figure 10  PEP Configuration - Logging**

### C.3.3 General Security Configuration

The "General Security Configuration" allows to set the following parameters:

- **PEP level keystore**: It allows to load the keystore containing the public certificates of the trusted Identity Providers. The certificates are used to check the signature of the security token in the input messages.
- **Keystore password**: It is the password to access the PEP level keystore
- **IdP key alias**: It is used only in the following special case: the token received in input shall be re-signed and re-encrypted. This is the alias of the private key used for signing.
C.3.4 Contact Information

The "Contact Information" tab allows to set the company name and reference person to contact in case of issue reporting.
C.4 Create a PEP Service

In order to protect a deployed service, select from the Start button in the left bottom area of the PEP Panel the entry "PEP Services Manager → Create PEP Services → From WSDL"

The displayed window provides four tabs:

- New PEP Service
- Security Configuration
- Authentication Checks
- Authorization Checks

**Note:** Once a new PEP service has been created and configured, it can be accessed at the following URL: http://<host>:<port>/PEP/services/newPEPservice where newPEPservice is the name chosen for the new service.

C.4.1 New PEP Service

The "New PEP Service" allows to enter the following parameters:

- **Service WSDL:** It is the URL of the service to be secured
- **Service Name:** It is the (user) name for the new PEP service
Additionally, it is possible to indicate the handling of the incoming SAML token: by default, the PEP removes the SAML token from the incoming request before forwarding it to the service endpoint. It is possible to modify this behavior, by selecting one of the possible choices (self-explanatory):

- **Forward message with incoming security token**
- **Forward message with security token re-encrypted**: This choice allows to re-encrypt the SAML token in input using a public key (certificate) contained in the service level keystore; the key is indicated through its alias
- **Forward message with security token decrypted**

![Image](image_url)

*Figure 13  PEP Service Creation - STEP 1*

### C.4.2 Service Security Configuration

The "Security Configuration" tab allows to set the parameters for the security layer of the service:

- **Keystore**: It is the keystore to be set at SERVICE level, containing the private key used to decrypt the security token in the input message
- (Service Keystore) **Alias**: It is the alias which identifies the private key in the keystore
- (Service Keystore) **Password**: It is the password to access the keystore. Notice that, if the private key is protected with a password, the keystore password and the private key password shall be the same

![Service KeyStore](image)

*Figure 14 PEP Service Creation - STEP 2*

### C.4.3 Authentication Checks

The "Authentication Checks" tab allows to specify additional actions/checks to be performed on the SAML token in input, in addition to the signature verification:

- **Send the token for persistence to configured SAML Bean**: It allows to send the token to a configured service to be persisted in its database; the identifier of the persisted token, returned by the service, is added to the SOAP header. This is a custom step implemented for the DAIL. The parameters for accessing the service are defined in a property files, named “saml.properties”
• **Check the token validity time:** It allows to check that the time range specified in the SAML token (as starting time and ending time for the token usage) is satisfied: for testing purposes, this check can be disabled.

• **Check the token issuer:** It allows to check that the issuer of the token matches the provided value.

![Image](image_url)

*Figure 15 PEP Service Creation - STEP 3 (optional)*

### C.4.4 Authorization Checks

The "Authorization Checks" tab allows to enforce access rules, if needed, on the incoming requests. Three alternative options are available:

- **Standard (Geo)XACML authorization:** It allows to provide an authorization file written in XACML (or GeoXACML) containing the policy rules to be enforced.

- **Custom step for the DAIL PEP:** Step specific for enforcing GMES access rules on the main DAIL PEP.
• **Custom step for the CDS PEP**: Step specific for enforcing GMES access rules on the CDS PEP

![Figure 16 PEP Service Creation - STEP 4 (optional)](image)

**C.5 Manage a PEP Service**

The PEP GUI allow to trace the activity of a PEP service. By clicking on Start → PEP Services, the list of the PEP services created and configured is provided. Once selected the PEP service to be analyzed, the following entry options are shown to the user:

- Service Information
- Service Log
- Configure Service
- Service Instances
- Delete Service
C.5.1 Service Information

The Service Information entry allows visualizing general information about the service which is secured:

- **Service Operations**: It lists the operations available on the secured service, the associated SOAP Action and other information on the type of the input and output elements

- **WSDL**: It is the WSDL of the secured service

- **Schema**: It is the W3C Schema imported by the WSDL of the secured service
**C.5.2 Service Log**

The Service Log entry allows visualizing the high-level log messages of the PEP service.

**C.5.3 Configure Service**

The Configure Service entry allows changing the configuration of the PEP service. This window is very similar to that described in section C.4, with the only exception of the missing WSDL and service name entries.

**C.5.4 Service Instances**

The Service Instances entry allows visualizing the instances processed by the PEP service. For each instance processed by the PEP service, the following information is provided:

- an incremental instance identifier
- the requested operation
- the arrival date
- the final status of the processing
Additionally, two icons are provided so that, by clicking on them, the request in input/the response in output can be displayed.

![Figure 19 PEP Service Instances](image)

### C.6 Change PEP Admin Password

From a default installation, the account to sign in the PEP administration pages is admin / admin. To change the default password to a new password, edit the following files on the PEP machine:

- `$PEP/WEB-INF/classes/it/intecs/pisa/pep/rest/conf/pepUsers.xml`
- `$PEP/WEB-INF/xml/adminUsers.xml`

To replace the string admin in "password="admin"" by a new password.

Where `$PEP` is the expanded directory of the PEP application, e.g. `/apps/apache-tomcat-6.0.37/webapps/PEP`

Note: you need to apply the same new password on the two files said above.

Restart the tomcat server to take the change into effect.

D.1 Introduction

The “CIM EP Catalogue Client” referenced in this document is an open source catalogue client that can connect to several catalogue servers. This catalogue client also supports several catalogue interface: the interface IF-HMAS-CIM-EP [RD04] interface is one of these supported interfaces. Within the context of HMA-S, it is also used as a EOP O&M Catalog client.

In the context of HMA-S, this software package can be found, as open-source software, on the google code repository at http://code.google.com/p/hma-catalog-client/. The following subsections provide the procedure to install and configure the client to connect to a CIM EP server.

D.2 Prerequisites

See “Prerequisites” of the catalog client installation in section 4 of [RD06].

D.3 Installation

The installation of the open source catalogue client is described in section 4 of [RD06].

D.4 Installation Testing

After finishing the steps above, start the Tomcat, and then access the link: http://$SERVER_NAME:$SERVER_PORT/catalogueClient1 through your browser. The installation is successful if the Client home page is shown as picture below:

---

1 For example http://hma-s:8080/catalogueClient/.
D.5 Configuration

D.5.1 Catalogue Registration

An XML file called “services.xml” located at $TOMCAT_HOME/webapps/catalogueClient/WEB-INF/classes describes all information about catalogue services accessible via the Client software.

A description of the different fields belonging to that configuration file and an example of such configuration file can be found in [RD06].

To demonstrate the user management capabilities some additional configuration is required as explained hereafter.

The following steps should be followed while registering a new service:

- Stop Tomcat
- Update the “services.xml” as described in [RD06].
- Add a new property into the Java properties file $TOMCAT_HOME/webapps/catalogueClient/WEB-INF/classes/um.properties, as follows:
  o Property key: is the service identifier (value of element Id in the service.xml file)
  o Property value: is the URN string that is set to the parameter AppliesTo in RST requests.
- Start Tomcat.

The following text is an example of such um.properties file:
D.5.2 SSO IDP Registration

The current CIM EP client supports to get a SAML token from a STS and have the SAML token included in Service requests to the Catalogue server. The client provides a Setting form page so that a user can store two following information into the browser session via the Setting form page: a User ID, STS endpoint, and a delegated STS endpoint.

![Figure 21 CIM EP Client Setting Form page](image)

The user field the text field “SSO User ID” to enter the user ID and select an STS endpoint from the dropdown list “SSO IDP”, as shown on the Figure above. The user can select a delegated endpoint from the dropdown list “Delegated SSO IDP” to request the STS (as selected in list SSO IDP) to delegate a RST request by means of parameter “DelegateTo” as specified in the specification [RD07].

On processing a Service request from the user, the Client firstly constructs a RST request with the SSO User ID and its signature, like in the case with SSO integration, and sends it to the STS endpoint for the SAML token. This SAML token is inserted in the Service request to be submitted to the Catalogue server endpoint. For more information, the feature said above is an initial step towards supporting an integration with a SSO IDP in a later version of CIM EP client software.

Registering a SSO IDP endpoint on the Client can be done by editing properties included in the following Java properties files:

- `$TOMCAT_HOME/webapps/catalogueClient/WEB-INF/classes/idp.properties`
- `$TOMCAT_HOME/webapps/catalogueClient/WEB-INF/classes/um.properties`
- `$TOMCAT_HOME/webapps/catalogueClient/WEB-INF/classes/delegated-idp.properties`

The following describes the properties in detail:
Each property in file idp.properties defines configuration information for each STS endpoint, as follows:

- The property key: is a string that matches to a property key in the file um.properties.
- The property value is the label for the STS endpoint as shown on the dropdown list “SSO IDP”).

Each property in file idp.properties must have a corresponding property in file um.properties with the same key strings. The value of the property in file um.properties is the STS access endpoint. Here is an example of these properties.

Properties in file idp.properties:
sts-1=STS-1 (GCM)
sts-2=STS-2 (ESA)

And their corresponding properties in file um.properties:
sts-1=http://hma-ssts-1.spacebel.be:80/axis2/services/sts
sts-2=http://hma-ssts-2.spacebel.be:80/axis2/services/sts

For each STS endpoint, make sure that the public key of the Client is registered on the STS side (see Annex B.3.2 in this document for more detail).

Each property in file delegated-idp.properties defines configuration information for each delegated STS endpoint, as follows:

- The property key: is the alias string of the STS registered as a federated STS of a STS defined in the file idp.properties. This property key is the same with a property key in the file “federated-sts.properties” on the STS configuration. See Appendix B.2.4 for more detail of the file federated-sts.properties.
- The property value is the label (shown on the dropdown list “Delegated SSO IDP”) for the delegated STS.

Note: the property association “na = Not Applicable” in the file delegated-idp.properties indicates that none of delegated STS is applied for RST requests.

### D.5.3 Keystore Configuration

The Client needs a keystore providing keys used to construct RST request case with signature. The keystore information is defined by a set of Java properties in file
$TOMCAT_HOME/webapps/catalogueClient/WEB-INF/classes/um.properties. The following table describes these properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clientKeystoreLocation</td>
<td>Path of the keystore file used for the Client.</td>
</tr>
<tr>
<td>clientKeystorePassword</td>
<td>Password of the keystore.</td>
</tr>
<tr>
<td>clientCertificateAlias</td>
<td>Alias for the private key stored in the keystore. The Client use this key to sign RST requests. Note: the corresponding public key must be registered on the STS side in order the Client is</td>
</tr>
</tbody>
</table>

trusted by the STS (see Annex B.3.2 for more detail).

Here below is an example of these properties

```java
clientKeystoreLocation=/apps/conf/keystores/cim-ep-client.keystore
clientKeystorePassword=changeit
clientCertificatePassword=changeit
```

Note: in case you don't have yet a keystore for the Client, you can generate one including a self-signed key pair typed "RSA" by using the following Java JDK keytool command:

```bash
```
Annex E: Apache Ant Script to Create Keystores

This Annex provides an Ant build script to create the keystores files as listed in section 3.3.2. Each keystore file is created by Ant target named “<keystore-filename>” and preconfigured for the Demonstrator installation by Ant target named “config.<keystore-filename>”, where <keystore-filename> is filename of the keystore file.

```
<project name="IdentityMgmtDemonstrator" default="help" basedir=".">

    <!-- project properties -->
    <property name="build.dir" value="build"/>
    <property name="dist.dir" value="dist"/>

    <target name="clean">
        <delete dir="${build.dir}"/>
        <delete dir="${dist.dir}"/>
    </target>

    <!-- Keystore management -->
    <target name="keystores.all">
        <antcall target="pep.keystore"/>
        <antcall target="pep-1.keystore"/>
        <antcall target="pep-2.keystore"/>
        <antcall target="sts.keystore"/>
        <antcall target="sts-1.keystore"/>
        <antcall target="sts-2.keystore"/>
        <antcall target="cim-ep-client.keystore"/>
        <antcall target="config.pep.keystore"/>
        <antcall target="config.pep-1.keystore"/>
        <antcall target="config.pep-2.keystore"/>
        <antcall target="config.sts.keystore"/>
        <antcall target="config.sts-1.keystore"/>
        <antcall target="config.sts-2.keystore"/>
    </target>

    <target name="cim-ep-client.keystore">
        <mkdir dir="${dist.dir}"/>
        <property name="file" value="${dist.dir}/cim-ep-client.keystore"/>
        <property name="cert.file" value="${dist.dir}/cim-ep-client.cert"/>
        <property name="alias" value="urn:ceos:def:epr:spacebel:1.0:cim-ep-client"/>
        <property name="pass" value="changeit"/>
    </target>
```

```
<!-- valid for 10 years-->

<property name="validity" value="3650"/>

<delete file="${file}"/>
<genkey alias="${alias}" storepass="${pass}" keyalg="RSA"
keystore="${file}" validity="${validity}">
<param name="CN" value="HMA-S"/>
<param name="OU" value="Space"/>
<param name="O" value="Spacebel"/>
<param name="C" value="BE"/>
</dname>
</genkey>
<exec executable="keytool">
<arg value="-export"/>
<arg value="-alias"/>
<arg value="${alias}"/>
<arg value="-rfc"/>
<arg value="-file"/>
<arg value="${cert.file}"/>
<arg value="-keystore"/>
<arg value="${file}"/>
<arg value="-storepass"/>
<arg value="${pass}"/>
</exec>
<echo message="Generated keystore at ${file}"/>
</target>
<target name="sts.keystore">
<mkdir dir="${dist.dir}"/>
<property name="file" value="${dist.dir}/sts.keystore"/>
<property name="cert.file" value="${dist.dir}/sts.cert"/>
<property name="alias" value="urn:ceos:def:epr:spacebel:1.0:sts"/>
<property name="pass" value="changeit"/>

<!-- valid for 10 years-->
<property name="validity" value="3650"/>
<delete file="${file}"/>
<genkey alias="${alias}" storepass="${pass}" keyalg="RSA" keystore="${file}" validity="${validity}">
  <dname>
    <param name="CN" value="HMA-S"/>
    <param name="OU" value="Space"/>
    <param name="O" value="Spacebel"/>
    <param name="C" value="BE"/>
  </dname>
</genkey>
<exec executable="keytool">
  <arg value="-export"/>
  <arg value="-alias"/>
  <arg value="${alias}"/>
  <arg value="-rfc"/>
  <arg value="-file"/>
  <arg value="${cert.file}"/>
  <arg value="-keystore"/>
  <arg value="${file}"/>
  <arg value="-storepass"/>
  <arg value="${pass}"/>
</exec>
<echo message="Generated keystore at ${file}"/>
</target>
<target name="sts-1.keystore">
  <mkdir dir="${dist.dir}"/>
  <property name="file" value="${dist.dir}/sts-1.keystore"/>
  <property name="cert.file" value="${dist.dir}/sts-1.cert"/>
  <property name="alias" value="urn:ceos:def:epr:gcm:1.0:sts-1"/>
  <property name="pass" value="changeit"/>
  <!-- valid for 10 years -->
  <property name="validity" value="3650"/>
  <delete file="${file}"/>
  <genkey alias="${alias}" storepass="${pass}" keyalg="RSA" keystore="${file}" validity="${validity}">
</target>
<dname>
  <param name="CN" value="HMA-S"/>
  <param name="OU" value="Space"/>
  <param name="O" value="Spacebel"/>
  <param name="C" value="BE"/>
</dname>

<exec executable="keytool">
  <arg value="-export"/>
  <arg value="-alias"/>
  <arg value="${alias}"/>
  <arg value="-rfc"/>
  <arg value="-file"/>
  <arg value="${cert.file}"/>
  <arg value="-keystore"/>
  <arg value="${file}"/>
  <arg value="-storepass"/>
  <arg value="${pass}"/>
</exec>

<echo message="Generated keystore at ${file}"/>

<target name="sts-2.keystore">
  <mkdir dir="${dist.dir}"/>
  <property name="file" value="${dist.dir}/sts-2.keystore"/>
  <property name="cert.file" value="${dist.dir}/sts-2.cert"/>
  <property name="alias" value="urn:ceos:def:epr:esa:1.0:sts-2"/>
  <property name="pass" value="changeit"/>

  <!-- valid for 10 years -->
  <property name="validity" value="3650"/>

  <delete file="${file}"/>
  <genkey alias="${alias}" storepass="${pass}" keyalg="RSA"
         keystore="${file}" validity="${validity}">
    <dname>
      <param name="CN" value="HMA-S"/>
      <param name="OU" value="Space"/>
    </dname>
  </genkey>
</target>
<param name="O" value="Spacebel"/>
<param name="C" value="BE"/>
</pathname>
</genkey>
<exec executable="keytool">
    <arg value="-export"/>
    <arg value="-alias"/>
    <arg value="${alias}"/>
    <arg value="-rfc"/>
    <arg value="-file"/>
    <arg value="${cert.file}"/>
    <arg value="-keystore"/>
    <arg value="${file}"/>
    <arg value="-storepass"/>
    <arg value="${pass}"/>
</exec>
<echo message="Generated keystore at ${file}"/>
</target>
<target name="pep.keystore">
    <mkdir dir="${dist.dir}"/>
    <property name="file" value="${dist.dir}/pep.keystore"/>
    <property name="cert.file" value="${dist.dir}/pep.cert"/>
    <property name="alias" value="urn:ceos:def:epr:spacebel:1.0:pep"/>
    <property name="pass" value="changeit"/>

    <!-- valid for 10 years-->
    <property name="validity" value="3650"/>

    <delete file="${file}"/>
    <genkey alias="${alias}" storepass="${pass}" keyalg="RSA" keystore="${file}" validity="${validity}"
        <dname>
            <param name="CN" value="HMA-S"/>
            <param name="OU" value="Space"/>
            <param name="O" value="Spacebel"/>
            <param name="C" value="BE"/>
        </dname>
    </genkey>
</target>
<exec executable="keytool">
   <arg value="-export"/>
   <arg value="-alias"/>
   <arg value="${alias}"/>
   <arg value="-rfc"/>
   <arg value="-file"/>
   <arg value="${cert.file}"/>
   <arg value="-keystore"/>
   <arg value="${file}"/>
   <arg value="-storepass"/>
   <arg value="${pass}"/>
</exec>
<echo message="Generated keystore at ${file}"/>
</target>

<target name="pep-1.keystore">
   <mkdir dir="${dist.dir}"/>
   <property name="file" value="${dist.dir}/pep-1.keystore"/>
   <property name="cert.file" value="${dist.dir}/pep-1.cert"/>
   <property name="alias" value="urn:ceos:def:epr:gcm:1.0:pep-1"/>
   <property name="pass" value="changeit"/>
   <!-- valid for 10 years-->
   <property name="validity" value="3650"/>
   <delete file="${file}/"/>
   <genkey alias="${alias}" storepass="${pass}" keyalg="RSA" keystore="${file}" validity="${validity}"
            dname="">
      <param name="CN" value="HMA-S"/>
      <param name="OU" value="Space"/>
      <param name="O" value="Spacebel"/>
      <param name="C" value="BE"/>
   </dname>
</genkey>
<exec executable="keytool">
   <arg value="-export"/>
   <arg value="-alias"/>
   <arg value="${alias}"/>
   <arg value="-rfc"/>
   <arg value="-file"/>
   <arg value="${cert.file}"/>
   <arg value="-keystore"/>
   <arg value="${file}"/>
   <arg value="-storepass"/>
   <arg value="${pass}"/>
</exec>
<echo message="Generated keystore at ${file}"/>
<arg value="-export"/>
<arg value="-alias"/>
<arg value="$\{alias\}"/>
<arg value="-rfc"/>
<arg value="-file"/>
<arg value="$\{cert.file\}"/>
<arg value="-keystore"/>
<arg value="$\{file\}"/>
<arg value="-storepass"/>
<arg value="$\{pass\}"/>

</exec>
<echo message="Generated keystore at $\{file\}"/>
</target>
<target name="pep-2.keystore">
<mkdir dir="$\{dist.dir\}"/>
<property name="file" value="$\{dist.dir\}/pep-2.keystore"/>
<property name="cert.file" value="$\{dist.dir\}/pep-2.cert"/>
<property name="alias" value="urn:ceos:def:epr:esa:1.0:pep-2"/>
<property name="pass" value="changeit"/>

<!--[-- valid for 10 years -->]
<property name="validity" value="3650"/>

<delete file="$\{file\}"/>
<genkey alias="$\{alias\}" storepass="$\{pass\}" keyalg="RSA" keystore="$\{file\}" validity="$\{validity\}">
<dname>
  <param name="CN" value="HMA-S"/>
  <param name="OU" value="Space"/>
  <param name="O" value="Spacebel"/>
  <param name="C" value="BE"/>
</dname>
</genkey>
<exec executable="keytool">
  <arg value="-export"/>
  <arg value="-alias"/>
<arg value="${alias}"/>
<arg value="-rfc"/>
<arg value="-file"/>
<arg value="${cert.file}"/>
<arg value="-keystore"/>
<arg value="${file}"/>
<arg value="-storepass"/>
<arg value="${pass}"/>
</exec>
<echo message="Generated keystore at ${file}"/>
</target>
<target name="config.pep.keystore">
<exec executable="keytool">
  <arg value="-import"/>
  <arg value="-noprompt"/>
  <arg value="-trustcacerts"/>
  <arg value="-alias"/>
  <arg value="urn:ceos:def:epr:spacebel:1.0:sts"/>
  <arg value="-file"/>
  <arg value="${dist.dir}/sts.cert"/>
  <arg value="-keystore"/>
  <arg value="${dist.dir}/pep.keystore"/>
  <arg value="-storepass"/>
  <arg value="changeit"/>
</exec>
</target>
<target name="config.pep-1.keystore">
<exec executable="keytool">
  <arg value="-import"/>
  <arg value="-noprompt"/>
  <arg value="-trustcacerts"/>
  <arg value="-alias"/>
  <arg value="urn:ceos:def:epr:gcm:1.0:sts-1"/>
  <arg value="-file"/>
  <arg value="${dist.dir}/sts-1.cert"/>
  <arg value="-keystore"/>
<exec executable="keytool">
  <arg value="-import"/>
  <arg value="-noprompt"/>
  <arg value="-trustcacerts"/>
  <arg value="-alias"/>
  <arg value="urn:ceos:def:epr:gcm:1.0:sts-2"/>
  <arg value="-file"/>
  <arg value="${dist.dir}/sts-2.cert"/>
  <arg value="-keystore"/>
  <arg value="${dist.dir}/pep-1.keystore"/>
  <arg value="storepass"/>
  <arg value="changeit"/>
</exec>

<exec executable="keytool">
  <arg value="-import"/>
  <arg value="-noprompt"/>
  <arg value="-trustcacerts"/>
  <arg value="-alias"/>
  <arg value="urn:ceos:def:epr:gcm:1.0:sts-1"/>
  <arg value="-file"/>
  <arg value="${dist.dir}/sts-1.cert"/>
  <arg value="-keystore"/>
  <arg value="${dist.dir}/pep-2.keystore"/>
  <arg value="storepass"/>
  <arg value="changeit"/>
</exec>

<exec executable="keytool">
<arg value="-import"/>
<arg value="-noprompt"/>
<arg value="-trustcacerts"/>
<arg value="-alias"/>
<arg value="urn:ceos:def:epr:gcm:1.0:sts-2"/>
<arg value="-file"/>
<arg value="$\{dist.dir\}/sts-2.cert"/>
<arg value="-keystore"/>
<arg value="$\{dist.dir\}/pep-2.keystore"/>
<arg value="-storepass"/>
<arg value="changeit"/>
</exec>
</target>

<target name="config.sts.keystore">
<exec executable="keytool">
<arg value="-import"/>
<arg value="-noprompt"/>
<arg value="-trustcacerts"/>
<arg value="-alias"/>
<arg value="urn:ceos:def:epr:spacebel:1.0:pep"/>
<arg value="-file"/>
<arg value="$\{dist.dir\}/pep.cert"/>
<arg value="-keystore"/>
<arg value="$\{dist.dir\}/sts.keystore"/>
<arg value="-storepass"/>
<arg value="changeit"/>
</exec>
</target>

<exec executable="keytool">
<arg value="-import"/>
<arg value="-noprompt"/>
<arg value="-trustcacerts"/>
<arg value="-alias"/>
<arg value="urn:ceos:def:epr:spacebel:1.0:cim-ep-client"/>
<arg value="-file"/>
<arg value="${dist.dir}/cim-ep-client.cert"/>
<arg value="-keystore"/>
<arg value="${dist.dir}/sts.keystore"/>
<arg value="-storepass"/>
<arg value="changeit"/>
</exec>

</target>
<target name="config.sts-1.keystore">
   <exec executable="keytool">
      <arg value="-import"/>
      <arg value="-noprompt"/>
      <arg value="-trustcacerts"/>
      <arg value="-alias"/>
      <arg value="urn:ceos:def:epr:gcm:1.0:pep-1"/>
      <arg value="-file"/>
      <arg value="${dist.dir}/pep-1.cert"/>
      <arg value="-keystore"/>
      <arg value="${dist.dir}/sts-1.keystore"/>
      <arg value="-storepass"/>
      <arg value="changeit"/>
   </exec>

   <exec executable="keytool">
      <arg value="-import"/>
      <arg value="-noprompt"/>
      <arg value="-trustcacerts"/>
      <arg value="-alias"/>
      <arg value="urn:ceos:def:epr:gcm:1.0:pep-2"/>
      <arg value="-file"/>
      <arg value="${dist.dir}/pep-2.cert"/>
      <arg value="-keystore"/>
      <arg value="${dist.dir}/sts-1.keystore"/>
      <arg value="-storepass"/>
      <arg value="changeit"/>
   </exec>
</target>
<exec executable="keytool">
  <arg value="-import"/>
  <arg value="-noprompt"/>
  <arg value="-trustcacerts"/>
  <arg value="-alias"/>
  <arg value="urn:ceos:def:epr:spacebel:1.0:cim-ep-client"/>
  <arg value="-file"/>
  <arg value="${dist.dir}/cim-ep-client.cert"/>
  <arg value="-keystore"/>
  <arg value="${dist.dir}/sts-1.keystore"/>
  <arg value="-storepass"/>
  <arg value="changeit"/>
</exec>

</target>

<target name="config.sts-2.keystore">
  <exec executable="keytool">
    <arg value="-import"/>
    <arg value="-noprompt"/>
    <arg value="-trustcacerts"/>
    <arg value="-alias"/>
    <arg value="urn:ceos:def:epr:esa:1.0:pep-1"/>
    <arg value="-file"/>
    <arg value="${dist.dir}/pep-1.cert"/>
    <arg value="-keystore"/>
    <arg value="${dist.dir}/sts-2.keystore"/>
    <arg value="-storepass"/>
    <arg value="changeit"/>
  </exec>
</target>

<exec executable="keytool">
  <arg value="-import"/>
  <arg value="-noprompt"/>
  <arg value="-trustcacerts"/>
  <arg value="-alias"/>
</exec>
<exec filename="src/main/resources/parametrized部署脚本 文件" input="dist/pep-2-cert.truststore/keystore" output="dist/pep-2-cert.truststore/keystore"
                          storepass="changeit"/>

<exec filename="src/main/resources/parametrized部署脚本 文件" input="dist/pep-2-cert.truststore/keystore" output="dist/pep-2-cert.truststore/keystore"
                          storepass="changeit"/>

<exec executable="keytool">
    <arg value="-import"/>
    <arg value="-noprompt"/>
    <arg value="-trustcacerts"/>
    <arg value="-alias"/>
    <arg value="urn:ceos:def:epr:spacebel:1.0:cert-client"/>
    <arg value="-file"/>
    <arg value="$(dist.dir)/cim-ep-client-cert"/>
    <arg value="-keystore"/>
    <arg value="$(dist.dir)/sts-2.keystore"/>
    <arg value="-storepass"/>
    <arg value="changeit"/>
</exec>

</target>
</project>
Annex F: HMA ONLINE RESOURCES

This section provides the links available on internet to resources referenced in this document. Two types of resources are available:

- Open source software that can be downloaded for installation.
- Online software (generally web clients) that can directly be accessed with an internet browser.

The following table provides the open source download URLs of the components involved in this demonstrator.

<table>
<thead>
<tr>
<th>Id</th>
<th>Download URL</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2.1</td>
<td>Catalogue Client</td>
<td><a href="http://code.google.com/p/hma-catalog-client">http://code.google.com/p/hma-catalog-client</a></td>
</tr>
<tr>
<td>H2.2</td>
<td>STS</td>
<td><a href="http://code.google.com/p/hma-security-token-service">http://code.google.com/p/hma-security-token-service</a></td>
</tr>
<tr>
<td>H2.3</td>
<td>PEP</td>
<td>TBD</td>
</tr>
</tbody>
</table>

Table 7: Open source download URLs

The following table provides the Online URL of the components deployed at Spacebel and ESA.

<table>
<thead>
<tr>
<th>Id</th>
<th>Online URL</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2.1</td>
<td>Catalogue Client</td>
<td><a href="http://hma-s-catalogue.esa.int/catalogueClient/">http://hma-s-catalogue.esa.int/catalogueClient/</a></td>
</tr>
</tbody>
</table>

Table 8: Online software