Open Geospatial Consortium Inc.

Date: 2010-09-08

Reference number of this document: OGC 07-118r8

Version: 1.0

Category: OpenGIS® Best Practice

Editors: P. Denis, SPACEBEL s.a.

User Management Interfaces for Earth Observation Services

Warning

This document is not an OGC Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type:	OpenGIS® Best Practice
Document subtype:	Best Practice
Document stage:	Approved
Document language:	English

License Agreement

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modified Intellectual Property must include, in addition to the above copyright notice, a notice that the Intellectual Property includes modifications that have not been approved or adopted by LICENSOR.

THIS LICENSE IS A COPYRIGHT LICENSE ONLY, AND DOES NOT CONVEY ANY RIGHTS UNDER ANY PATENTS THAT MAY BE IN FORCE ANYWHERE IN THE WORLD.

THE INTELLECTUAL PROPERTY IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE DO NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE INTELLECTUAL PROPERTY WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE INTELLECTUAL PROPERTY WILL BE UNINTERRUPTED OR ERROR FREE. ANY USE OF THE INTELLECTUAL PROPERTY SHALL BE MADE ENTIRELY AT THE USER'S OWN RISK. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR ANY CONTRIBUTOR OF INTELLECTUAL PROPERTY RIGHTS TO THE INTELLECTUAL PROPERTY BE LIABLE FOR ANY CLAIM, OR ANY DIRECT, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM ANY ALLEGED INFRINGEMENT OR ANY LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR UNDER ANY OTHER LEGAL THEORY, ARISING OUT OF OR IN CONNECTION WITH THE IMPLEMENTATION, USE, COMMERCIALIZATION OR PERFORMANCE OF THIS INTELLECTUAL PROPERTY.

This license is effective until terminated. You may terminate it at any time by destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable

Contents

Figures

i. Preface

This document describes how user and identity management information may be included in the protocol specifications for OGC Services. The use cases addressed will make reference to EO (Earth Observation) services, for example catalogue access (EO Products Extension Package for ebRIM (ISO/TS 15000-3) Profile of CSW 2.0 [OGC 06-131]), ordering (Ordering Services for Earth Observation Products [OGC 06-141r2]) and programming (OpenGIS Sensor Planning Service Application Profile for EO Sensors [OGC 07-018r2]).

The document was initially produced during the ESA HMA (Heterogeneous Missions Accessibility) project and refined during the FEDEO (Federated Earth Observation) Pilot. It was further refined in the ESA EODAIL and HMA-T projects.

This document is not an OGC standard. This document describes how existing specifications from W3C and OASIS can be used in combination to pass identity information to OGC Web services.

ii. Submitting organisations

The following organisations will submit the original document or its revisions to the OGC™ Security Working Group.

- **Spacebel s.a.**
- **ESA – European Space Agency**
- **Intecs**
- **STFC**

The editors would like to acknowledge that this work is the result of collaboration and review of many organisations and would like to thank for the comments and contributions from:

- **Astrium**
- **Spot Image**
- **ASI**
- **CNES**
- **DLR**
- **Eumetsat**
- **EUSC**
- **MDA**
- **con terra**
- **Terradue**

- **Rhea System**
- **Oracle**
- **Siemens**

Note: this does not imply a complete endorsement by these organisations.

iii. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Current contributors:

Contact	Organisation	Email
Pierre Denis	Spacebel	Pierre.Denis@spacebel.be
Maria-Rosaria Barone	Intecs	mariarosaria.barone@intecs.it
Stefano Puri	Intecs	stefano.puri@intecs.it
Andrew Woolf	STFC	andrew.woolf@stfc.ac.uk

Previous contributors:

Contact	Organisation	Email
Rowena Smillie	Spacebel	Rowena.Smillie@spacebel.be
Alexandre Cucumel	Spacebel	-
Wouter Van de Weghe	Oracle	wouter.van.de.weghe@oracle.com

iv. Future work

- Formalisation of the WSDL interface using WS-Policy

v. Foreword

This Best Practice document, through the implementation profile, references several external standards and specifications as dependencies. These are indicated in section 3.1.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium Inc. shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

Introduction

This OGC Best Practice is complementary to a set of OGC Services and related standards and specifications that describe services for managing Earth Observation (EO) data products. These services include collection level, and product level catalogues, online-ordering for archived and to be acquired (future) products, on-line access to these EO products, etc. The application of the current Best Practice is not limited to the Earth Observation domain however, as this document can be considered as a model which could be extended to other OGC application domain and to other bindings beyond the SOAP and HTTP ones described in the following.

The intent of this Best Practice is to describe an identity management interface that can be implemented and supported by many data providers (satellite operators, data distributors …), most of whom have existing (and relatively complex) facilities for the management of their data and users. The proposed strategy is to specify a platform and provider independent interface using existing standards.

1 Scope

This proposed interface document describes the interfaces required to authenticate and authorise users in a federated system of OGC Web Services for Earth Observation. The document has been written with three high level scenarios in mind:

- The orchestration of OGC Web Services as it may occur when (e.g.) Sensor Planning Service, Web processing Service and Web Coverage Service are provided by several cooperating organizations.
- The system of systems of OGC Web Services as it may occur when several organisations may concur and cooperate in the provision of instances of the same service within a federated service provision. Several relevant use cases are proposed within the GEOSS AIP.
- The security and EO products market scenarios which have high level requirements related to the user authentication as well as to the authorisation to the use of the OGC Web Services over geospatial (e.g. area of interest) and/or temporal parameters.

The purpose of this document is to describe how:

HL-REQ010 To perform user authentication (and authorisation) for the use of existing OGC Web Services (i.e. without changes to published OGC standards).

HL-REQ020 To use OASIS and W3C already defined standards for authentication and authorisation of OGC Web Services.

HL-REQ030 To federate different user communities allowing cross authentication for the purpose of using OGC Web Services.

HL-REQ040 To perform authentication and authorisation across orchestrated OGC Web Services.

HL-REQ050 To perform authentication and authorisation across a “system of systems” based on OGC Web Services.

HL-REQ060 To map an authentication environment based on HTTP binding and Web-SSO (e.g. Shibboleth) with the one based on SOAP and SAML.

Hereafter a brief outline of the document content allows readers to jump directly to the topic of their interest:

- the authentication use cases with the use of the SOAP binding is addressed in the chapters 6 and 7;
- the mapping of an authentication environment based on HTTP with one based on SOAP as required by HL-REQ060 above is addressed in Chapter 8;
- security considerations linking selected threats and risks to proposed countermeasures are addressed in Chapter 9;
- the authorisation use case and the possible link with XACML and GEOXACML are addressed in Chapter 10.

2 Conformance

2.1 Conformance to base specifications

This present section describes the compliance testing required for an implementation of this Best Practice.

It is worth highlighting that this OGC document references and uses specifications (SAML, WS Security, XACML) that come from other organizational bodies (such as the Organization for the Advancement of Structured Information Standards - OASIS) for which the concept of "conformance testing" does not apply; consequently, it is not possible to recursively testing the conformance to the compound specifications.

2.2 Conformance classes

We assume that a unique "core" conformance class encompassing all of the specification clauses in the Best Practice is defined and assume that the "Abstract Test Suite" is made up of this unique conformance class ("the core"). This class defines test cases, which covers:

- Test Module Basic requirements
- Test Module Authorisation

These are detailed in the Abstract Test Suite (see Annex A).

3 References

3.1 Normative references

[NR1] W3C Recommendation January 1999, Namespaces In XML, http://www.w3.org/TR/2000/REC-xml-names

[NR2] W3C Recommendation 6 October 2000, Extensible Markup Language (XML) 1.0 (Second Edition), http://www.w3.org/TR/REC-xml

[NR3] W3C Recommendation 2 May 2001: XML Schema Part 0: Primer, http://www.w3.org/TR/2001/REC-xmlschema-0-20010502/

[NR4] W3C Recommendation 2 May 2001: XML Schema Part 1: Structures, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[NR5] W3C Recommendation 2 May 2001: XML Schema Part 2: Datatypes, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[NR6] W3C Simple Object Access Protocol (SOAP) Version 1.1 W3C Note 08 May 2000 , http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[NR7] WSDL, Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/wsdl

[NR8] IETF RFC 2119, Keywords for use in RFCs to Indicate Requirement Levels, http://rfc.net/rfc2119.html

[NR9] WS-Security, SOAP Message Security V1.1 http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

[NR10] SAML, Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1 http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf

[NR11] Web Services Security SAML Token Profile 1.1 http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf

[NR12] Secure Hash Standards (SHA-1) National Institute of Standards and Technology http://csrc.nist.gov/cryptval/shs.htm

[NR14] Glossary for the OASIS Security Assertion Markup Language (SAML) http://www.oasis-open.org/committees/security/docs/cs-sstc-glossary-01.pdf

[NR15] Java Cryptography Architecture API Specification & Reference

http://java.sun.com/j2se/1.5.0/docs/guide/security/CryptoSpec.html

[NR16] OGC 04-016r5, OWS Common Implementation Specification 2004/12/17

[NR17] XML encryption http://www.w3.org/TR/xmlenc-core/

[NR18] XML signature http://www.w3.org/TR/xmldsig-core/

[NR19] Apache XML Security http://santuario.apache.org/Java/index.html

[NR20] W3C Recommendation 4 September 2007, Web Services Policy 1.5 - Framework, http://www.w3.org/TR/ws-policy/

[NR21] OASIS eXtensible Access Control Markup Language (XACML) TC http://www.oasis-open.org/committees/xacml

[NR22] SOAP Version 1.2 Part 1: Messaging Framework (Second Edition), W3C Recommendation 27 April 2007, http://www.w3.org/TR/soap12-part1/

[NR23] OASIS WS-Trust 1.3 http://docs.oasis-open.org/ws-sx/ws-trust/v1.3/ws-trust.pdf

[NR24] OASIS WS-Security UsernameToken Profile 1.1 http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-UsernameTokenProfile.pdf

[NR25] OGC 07-026r2, Geospatial eXtensible Access Control Markup Language (GeoXACML), 1.0

[NR26] Web Services Federation Language (WS-Federation) Version 1.2 http://www.oasis-open.org/apps/group_public/download.php/31658/ws-federation-1.2-spec-cs-01.doc

3.2 Other references

[OR1] Shibboleth
http://shibboleth.internet2.edu/

4 Terms and definitions

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

4.1.
Authentication [NR14]
Verification that a potential partner in a conversation is capable of representing a person or organization

4.2.
circle of trust
A federation of Service Providers and identity providers within which Service Providers accept the authentication asserted by the identity provider.

4.3.
Claim
A statement made about a client, service or other resource (e.g. name, identity, key, group, privilege, capability, etc.).

4.4.
client
Software component that can invoke an **operation** from a **server**

4.5.
identifier
a character string that may be composed of numbers and characters that is exchanged between the client and the server with respect to a specific identity of a resource

4.6.
identity provider [NR14]
A kind of Service Provider that creates, maintains, and manages identity information for principals and provides principal authentication to other Service Providers within a federation, such as with Web browser profiles.

4.7.
interface
named set of operations that characterise the behaviour of an entity [ISO 19119]

4.8.
operation
specification of a transformation or query that an object may be called to execute [ISO 19119]

4.9.
parameter
variable whose name and value are included in an operation **request** or **response**

4.10.
PEP
Policy Enforcement Point.

4.11.
principal [NR14]
A system entity whose identity can be authenticated.

4.12.
Relying Party [NR26]
A Web application or service that consumes **Security Tokens** issued by a **Security Token Service**.

4.13.
request
invocation of an **operation** by a **client**

4.14.
response
result of an **operation,** returned from a **server** to a **client**

4.15.
Security Token
A collection of **claims**. In the present Best Practice, the so-called "SAML token" is a specific kind of security token where the claims are SAML assertions.

4.16.
Security Token Service
A security token service (STS) is a Web service that issues security tokens.

4.17.
server service instance
a particular instance of a **service** [ISO 19119]

4.18.
service
distinct part of the functionality that is provided by an entity through interfaces [ISO 19119]

capability which a Service Provider entity makes available to a service user entity at the interface between those entities [ISO 19104 terms repository]

4.19.
service interface
shared boundary between an automated system or human being and another automated system or human being [ISO 19101]

4.20.
Service Provider [NR14]
A role donned by a system entity where the system entity provides services to principals or other system entities.

4.21.
transfer protocol
common set of rules for defining interactions between distributed systems [ISO 19118]

5 Symbols and abbreviations

5.1 Symbols (and abbreviated terms)

Some frequently used abbreviated terms:

ATS Abstract Test Suite

BPEL Business Process Execution Language

DAIL Data Access Integration Layer

EO Earth Observation

ETS Executable Test Suite

HMA Heterogeneous Missions Accessibility

HTTP HyperText Transport Protocol

IdP Identity Provider

ISO International Organisation for Standardisation

OASIS Advancing Open Standards for the Information Society

OGC Open Geospatial Consortium

PDP Policy Decision Point

PEP Policy Enforcement Point

RST Request Security Token

RSTR Request Security Token Response

SAML Security Assertion Markup Language

SOAP Simple Object Access Protocol

SP Service Provider

SSO Single Sign-On

STS Security Token Service

URI Uniform Resource Identifier

URL Uniform Resource Locator

URN Uniform Resource Name

WSDL Web Service Definition Language

W3C World Wide Web Consortium

XACML eXtensible Access Control Markup Language

XML eXtensible Markup Language

5.2 Document terms and definitions

This document uses the specification terms defined in Subclause 5.3 of [NR16].

6 System context

This section documents special requirements and describes the context of use.

6.1 Application domain

Web service requests are received by Service Providers. These Service Providers should be able to identify who issued the request and react accordingly. The following approach is proposed:

1) A Security Token Service (STS) provides a Request Security Token operation (RST), which returns a SAML token, an artefact representing an authenticated user. Depending whether STS is in charge of authentication or not, two main cases are defined:

    a. The STS *is* in charge of authentication: the RST contains user identifier, password and optionally his identity provider. This authentication Web service may federate the identity to another identity provider for authentication. At the interface context this is transparent, the federated identity request being identical to the initial request.

    b. The STS *is not* in charge of authentication, i.e. this is taken in charge by an external IdP, which is trusted by STS: the RST just contains user identifier (no password); it shall be signed in order to check that the requester is trusted.

2) Each subsequent service request by the client (Web service consumer) should include the SAML token in the SOAP header as described later in this document.

3) Each Service Provider accepts service requests only via an Authorisation Service or "Policy Enforcement Point" (PEP). The PEP first checks the existence of SAML token and decrypts it.

4) The PEP verifies the SAML token (signature and expiry time)

5) The PEP decides based on the content of the message body, the contents of the message header (including authentication token) and the context (i.e. applicable policies) whether to accept or to refuse the service request or reroute it. Although this is not imposed, the use of XACML [NR21] or geoXACML [NR25] for definition of policy rules is recommended.

6) If the request is authorised, then the request is processed by the target SP.

If any of the steps from 3) to 5) fails, then a fault response is returned to the client.

The distinction between steps 1.a. and 1.b, which discriminate on the IdP responsibility, is depicted in the following diagram (client A authenticates on STS, client B authenticates on external IdP).

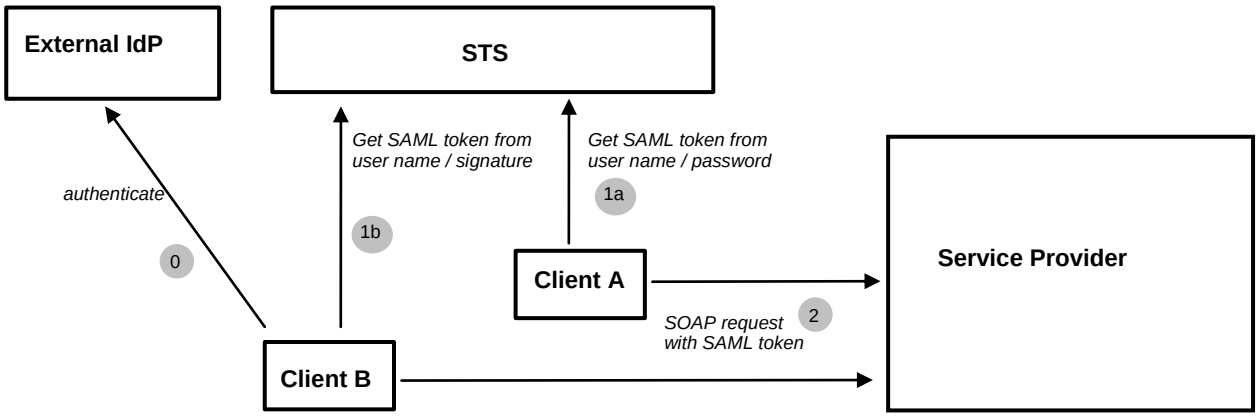

Figure 1 Two cases of authentication

These two cases are refined and detailed in section 6.4.3.

The full authentication & authorisation process is detailed in the following figure. This figure highlights the typical sequence of steps from authentication to request authorisation and processing (the two authentication cases are here abstracted).

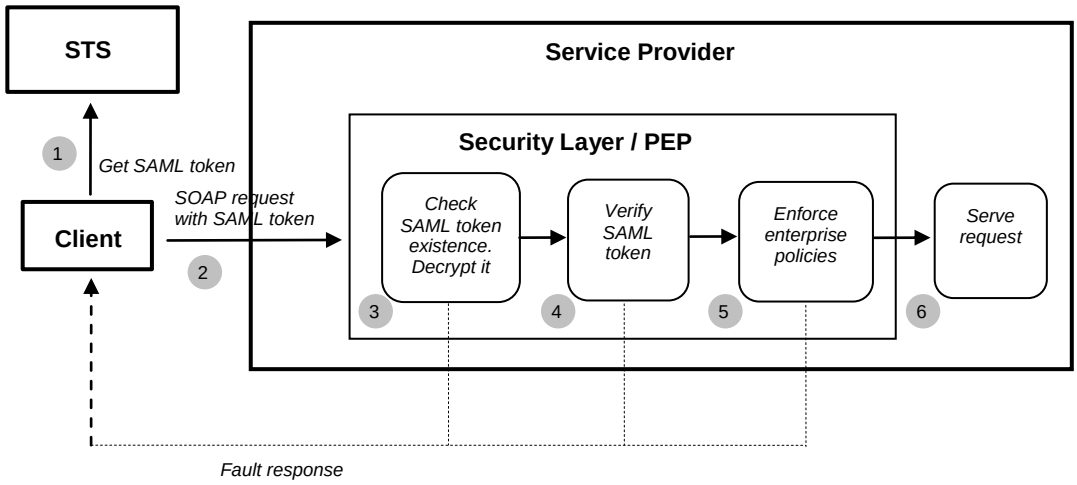

Figure 2 Sequence of authentication/authorisation activities

6.2 Protocol binding

To provide an overall coherent architecture within this context, operations shall support the embedding of requests and responses in SOAP messages. Only SOAP messaging (via HTTP/POST or HTTPS/POST) with document/literal style shall be used. Messages should conform to SOAP 1.2 [NR22].

For RSTs, the body of SOAP envelope is used. The SOAP header may be used to encapsulate a detached signature (see below).

For service requests, the message payload shall be in the body of the SOAP envelope and the authentication token shall be in the WS-Security element in the header of the SOAP envelope.

6.3 Basic use cases

The use cases covered by this Best Practice are shown in the following sequence diagram:

- Authentication: A Request Security Token (RST) is first issued to the Security Token Service (STS).
- Authorisation: A service request sent to the Service Provider (SP). This service request is, for instance, a call to any of the operations defined in the catalogue (OGC 06-131), ordering (OGC 06-141) or programming (OGC 07-018) specifications. The service requests can be synchronous or asynchronous via WS-Addressing. This is transparent for the purposes of this Best Practice.

An entity may be either an identity provider (IdP), a Service Provider (SP) or both IdP and SP.

In all the use cases presented in the document, the "Client" is the entity that issues requests to the IdP or SP. It is not the entity running the front-end application used by the human user (e.g. Web browser); this front-end accesses the client of IdP / SP, but is not by itself such client. This remark is especially important for use case relying on "trusted clients" (see 6.4.3.3).

The policy enforcement on the SP is non-invasive meaning that it is independent of the SP implementation.

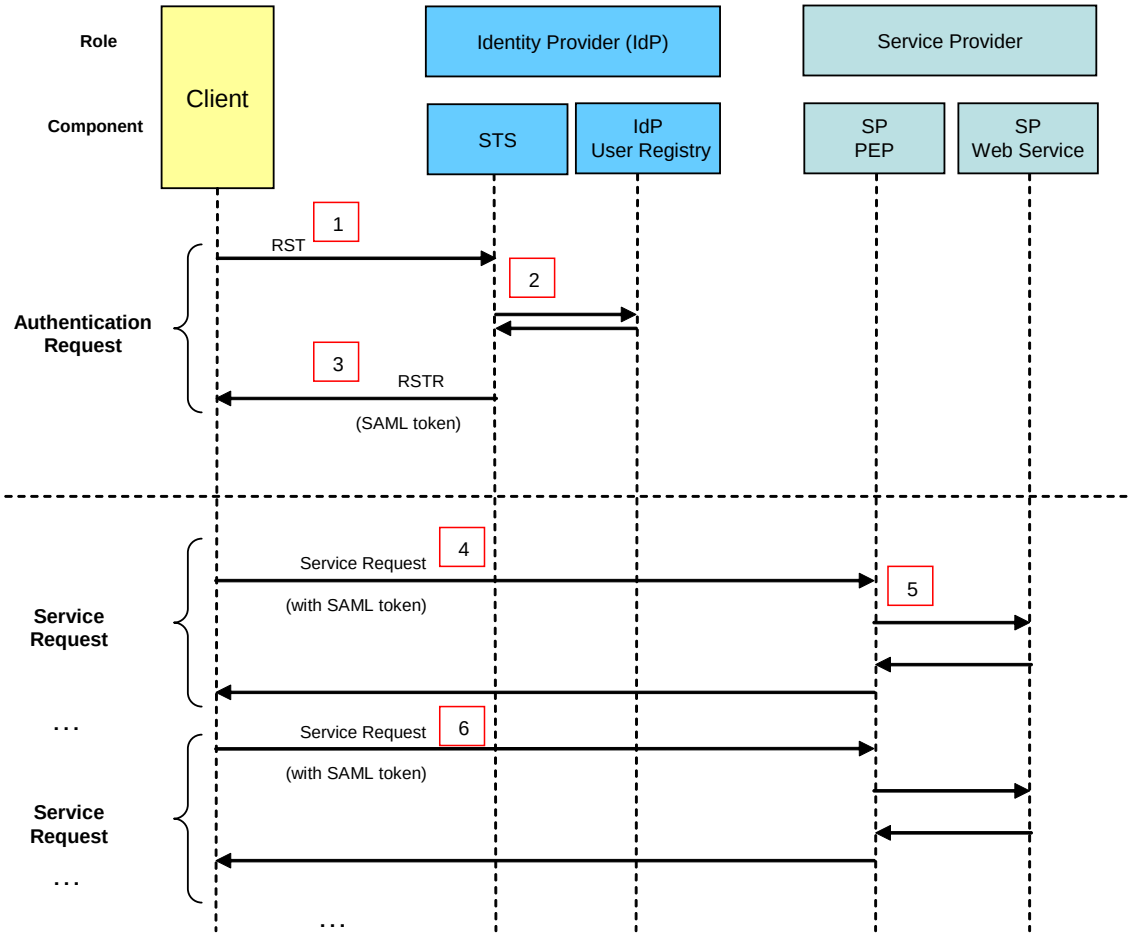

Figure 3 Authentication / Authorisation Use Case

A high level use case for authentication and authorisation is shown in the above figure. Note that the diagram has a higher level of abstraction than the other diagrams present in the remaining of the document; more precisely, the IdP depicted in the figure may either authenticate users on its own or delegate the authentication to another IdP. The same applies for the depicted SP. Following sections of this document further elaborate the detail of the authentication and authorisation.

1. The RST is sent by the client to the STS, which is directly exposed as a Web service. If required, a request could equally be intercepted by a PEP on IdP and routed to the actual STS (this option however is not followed in the remaining of the document).

2. The IdP performs checks on RST and/or user identity and, if successful, retrieves user attributes; then it sends back an RST Response (RSTR) with SAML token containing assertions on these attributes.

3. The client receives the RSTR containing the SAML token.

4. The client then sends a service request containing the SAML token.

5. The request is received by a PEP that take the decision to authorise or refuse the request, based on the attached SAML token.

6. This client may send other service requests with the same SAML token (i.e. without re-issuing authentication request), provided that the validity of this token has not expired.

It is worth mentioning that authentication request is not directly coupled with subsequent service requests. The client is just in charge of attaching a valid SAML token on each request addressed to PEP-protected SP. The same SAML token can be reused to successfully access several services, provided[1]

- that the PEP of the targeted service is a Relying Party matching the SAML token,
- that the SAML token is valid (e.g. expiry time),
- that the access policy enforced by the PEP authorises the request.

Based on thes constraints, the actual sequence of authentication requests and service requests is determined by the client, depending on the token renewal algorithm, on the targeted services and on the expiry period of the SAML token defined by the STS.

6.4 Security Model

The model is based on OASIS SAML 1.1 [NR10][2], WS-Security SAML token profile [NR11] and, for the issuance of SAML token, on OASIS WS-Trust 1.3 [NR23] and OASIS Web Services Security UsernameToken Profile 1.1 [NR24].

For the present need of SAML token delivery, only one operation of WS-Trust 1.3 is required: the *RequestSecurityToken* (RST), limited to the "Issue" action, as it is defined in the Issuance Binding section (§4) of [NR23]. This operation returns a RequestSecurityTokenResponse (RSTR).

The purpose of RST (with "Issue" action) in the present Best Practice is to provide a SAML token to a requester, provided that it gets proof that it can trust this requester. The actual proof of trust depends on which entity is responsible to authenticate users, i.e. which entity is the IdP. The present interface supports two kinds of IdP organisation, which entails two different RST formats:

1. *the IdP is the STS (or it can be accessed by the STS)*: in this case, the RST contains the name and password identifying the user plus an optional definition of the designated IdP; the STS checks that the user can be authenticated with these credentials or relay the authentication to the designated IdP;
2. *the IdP is an other system, not accessible by STS*: in this case, the RST shall contain a user id and shall be digitally signed: the STS checks that the signature corresponds to a requester that it trusts. For this purpose, the STS shall maintain a list of public keys of all the requester entities it trusts.

Case 1, based on user id/password pair, is the usual pattern that has been covered in all previous versions of the present document. Case 2 has been introduced in version 0.0.6; it allows subcontracting user identification to an external system, which should not be compliant with the present interface; we mean here SSO systems like OpenSSO, Shibboleth and ESA UM-SSO (see section 8). The context of case 2 is typically a Portal system that assures that a given user has been authenticated and then issues to the STS that trusts this Portal a signed RST with the authenticated user id.

For the ease of description of the differences between the two cases, we shall use in the following the wording *RST with password* for case 1 and *RST with signature* for case 2.

---

[1] These elements are detailed and explained in the remaining of the document.

[2] See possible integration of SAML 2.0 in "extension points" section (§6.4.5).

In all cases, the returned message is a *Request Security Token Response* (RSTR), carrying a SAML token (see [NR23]), which contains *assertions*[3] about the authentication and attributes of the identified user.

The STS receives user credentials in SOAP over an encrypted channel i.e. HTTPS. The signed and encrypted SAML token is returned as SOAP over HTTPS and subsequently used in service requests. It is an explicit design decision that the client is unable to decrypt the content of the encrypted SAML token.

6.4.1 Encryption

Encryption of the SAML token is performed by the STS during the processing of RST. Decryption is performed by the PEP during the processing of service request. The encryption protocol is a "hybrid cryptosystem", i.e. it uses together symmetric key encryption and public key encryption. More precisely, it is defined by

- a key encapsulation scheme, which is a public-key cryptosystem, and
- a data encapsulation scheme, which is a symmetric-key cryptosystem.

From an external point of view, the hybrid cryptosystem is itself a public-key system, which public and private keys are the same as in the key encapsulation scheme. This statement is important for the remaining of the present document where public-key cryptosystem is assumed while symmetric-key encryption aspect is left aside.

The data encryption algorithm used is the AES-128 (symmetric key) while the key encryption is uses RSA (public key), as defined in [NR15]. The full encryption process is as follows:

1. The STS first creates the symmetric key using the AES-128 encryption algorithm.
2. This symmetric key is then itself encrypted with the public key of the entity that shall consume the SAML token using the RSA encryption algorithm to create a secret key.
3. The SAML token (i.e. the SAML Assertion element) is then encrypted with the generated secret key using the AES-128 encryption algorithm. Note that the encryption type is *Element*, which means that the SAML Assertion element itself is encrypted, not only its child elements; this is specified by the Type attribute of EncryptedData element:
   ```
   <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
   Type="http://www.w3.org/2001/04/xmlenc#Element">
   ```
4. The message is then built.

The rationale of step 2 is that the SAML token is encrypted for a specific target Service Provider, which can be a Federating SP or not. Only the PEP of the targeted SP is able to decrypt the SAML token, through its private key. The criterion used by IdP to choose the "right" public key will be described in the next subsection (6.4.1.1).

Example Request Security Token with password:

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

[3] The concept of "assertion" here is a specific instance, in the SAML context, of the concept of "claim" in WS-Trust ([NR23]).


```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
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/">
    <soapenv: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>
          <wsse:UsernameToken>
             <wsse:Username>JohnDoe</wsse:Username>
             <wsse:Password>MyPassword</wsse:Password>
          </wsse:UsernameToken>
       </wst:RequestSecurityToken>
    </soapenv:Body>
</soapenv:Envelope>
```

Example Request Security Token with signature:

```
<S11:Envelope xmlns:S11="…" xmlns:wsse="…"
        xmlns:xenc="…" xmlns:wst="…">
  <S11:Header>
    <wsse:Security>
       <ds:Signature xmlns:ds="…">
          …
          <ds:Reference URI="#soapbody"/>
          …
       </ds:Signature>
    </wsse:Security>
  </S11:Header>
  <S11:Body Id="soapbody">
    <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>
       <wsse:UsernameToken>
          <wsse:Username>JohnDoe</wsse:Username>
       </wsse:UsernameToken>
    </wst:RequestSecurityToken>
  </S11:Body>
</S11:Envelope>
```

Example Request Security Token Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
 <soapenv:Body>
       <wst:RequestSecurityTokenResponse xmlns:wst="http://docs.oasis-
open.org/ws-sx/ws-trust/200512/"
xmlns:wsa="http://www.w3.org/2005/08/addressing"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
```



```
wssecurity-secext-1.0.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
          <wst:TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-
token-profile-1.1#SAMLV1.1</TokenType>
          <wst:RequestedSecurityToken>
             <xenc:EncryptedData
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
Type="http://www.w3.org/2001/04/xmlenc#Element">
                <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
           <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
             <xenc:EncryptedKey>
               <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
               <xenc:CipherData>

<xenc:CipherValue>cbE8viFOmyDuxR8N4EdwS9UUKpSoUbMrWSVprW7IypMwFZLeHR9Rxd4iw5
dUl4K+TffyNdRJ9Tr9PD8YIdpFLzCvYas63g5x4/XnyA1E2AU8ZBBpM2dtbr3g4IYMywfraWrI76
mHM+MERVZdHMVBWFrhqXhcS92m23m+amt14mk=</xenc:CipherValue>
               </xenc:CipherData>
             </xenc:EncryptedKey>
           </ds:KeyInfo>
           <xenc:CipherData>

<xenc:CipherValue>VEHlprDMQ+DqIpoPqx6TYi/mMX2dGV5JCJCrhDquZHRKqOiaIFfwqOMZvn
2HW2JDFvUxJ6LRTKdNujQI7sxc6h3IGBL7NXF7bx4jGwQ09wAA7nm6OoB4jiGdaqb8wTx0olnzn2
WqOWoVeTng1lwBi0rv2+iD1HWnXAUUHfJH8ALq4IU3hR0vjoqJH6Y21EuXPeXp/dYPUw3oIFn2FE
ID2u+8T+xOxbbq2ezQbU3z8n1LbgvDtN3ex5lUCo260pOOPn92nn7nYErT682eYd+bCKoiENpQSY
gHszvvyqFf9o6O0u87zk4AORWsRhQH74L2gG8wVOeHKyhEx0RsBkf4xZcQKBvQ9JHWQWpDEB51NZ
aJe1hSyaUk6T5gf9ArDnz6UwL0ZTDp6Dxgjha91u5qIMG3ECxVYKcnBv+O6Om1Q0HbL0ecbUDR56
evS+mf0U9JxduBKwFJLqta6D0wmwqYWcaF3ZrKd7SatV8Z2l0DmWTMe5R+x6O1RpbKltlduKl4bL
aSYFpaqaU758ZsmTDmjQQj8fn1qCZbDtp4SEVPWumoTg2k7RAOay2QtV5b+VA9wloSXoxVf2csLS
OOH/NDE1noBIpzgUb9Xm/YIPwikQKsxNPFM72yLrS0vjAho1Cxrg+8l7XIVcmowhPnLqSs6ZpvA0
1YP8EhsOFlN+0y+9EfAuoY4jYcScfwqDehth76lER+EyAdFLi10VhVxKW14VLbmksAydndIQaw6V
zGm1Qwoc3CeCaeq4q4GgFgiem1BmW9IeBaUBTX2wZmIKG8Z9Xhjv6MwT7hOeWH5fefipJs8JS8l6
wQBo8WAczzmw6s1j8JW9YDyAWosfoTPrtOwFTaaYSiaEXvPOnb5RgR/W4ivZ64ioA8FXyLFoWcNE
JJ6AgWHDLAbCDg/zvnVwEs7OdaSxRTxVNsc7cpclGspSmk/HzGYxPHInGhn/QPsac5iN6t6HlwnQ
UJgt81rI/tbFfSYqqtYqXKeNoEtw91/1DZVUi7mSc7Xj2e2Wb65h8PIoYeX3Nli+i4SrOoeAKaZr
HtpqP6f+pI4lpkANS4RFxFDiL9Ddxv1WKD//nMck0Su0HfIbPYUYF0GGvlHsv6IiwT8dj/f0MnCx
kAgegliGageZthQiNavOcURRC/94d+1jDZGayowurzdxmJhxyiEY5REQQt3hK4aAD89wMjndzxHd
tcQEuvXA5uSm3T9qgIm4Qdvuh54PW/SKptG9fDj4paTxVVlfZ+0f/1Vxjj4pPIKOVjE3e4ChBPkJ
XD/nXqZ8DdR+zPxOLWYyiqnMaxv3OInd/Iz2Lq36a09b0JEfMVz4e39sGtFzNDbxXgQnTx4L3jDY
Fdl5+gelUNduK9HtgklXDWfNIMWtY5xhdTX0m3Q8hBtNgKOHeg7BcBXF+uT30mqwgJu5cbJQl/1j
/QlMvromUaUQATN2ULu7mMiTWkoYoMTiijJGAizbKIi6O5xmHvF/jicd7lBcmSz+B+BnrnqxY5DM
/qQSFsnRoGmPKlJeiao2g+QuMD5x7H+pBUIq3B8lkMlUBg5VoKx2+kCHuP2lamGFskQl80PRGygQ
5adA2iWwKoIKoCdcYIc9C2sPVkJz+slExJXizl4L51GEWDlQ8VGsqNV7CzOyIt0uXIIBQW3j0aX/
/7QoYVfM681TiqvtaDEY7Ip4nSV839e5xnj3s+qgzXOpoK5rw5EThDLhthPy97CJiuSbsfcGf1Dv
WNE74x2E4b8HazacItRBIbx0GFDHIoqaHEih6zlhQaqwloLnUHRpL8vAQLVKiW3q94569e3GenoQ
bpjxKQ9F58VuQh3ZiZtJ+17XOxDx6ZDXcTiQDa+3nXiTgT7k9gGtpIv8vYLMuUHDEZx1zGd/rMzU
3JAbbfKO8+Cs3pMnb2KpGL4rLLgivee1P35rbHK04V8D0NbwDk0TOVnmFQWIRsgVtPwmEHXbFl3j
qIUTx4xdishXmkKcDCwaNfqYo4yzmgLUlbtchkDGF9YBA0mXv4gT7z0TiBb1jUFBhTnciL4DBkI9
K8wEjklHUi7w4LjvhCB5B15y1ZG/baIscoRrVu41HUp7crwbsdKjcWGE//dBTXN1vrXDm0maAkCo
nuYNPpMY0Cf+ikVITJO73UaXplFjOOtm+mkql6e5nTd8qGQXwHZl/nGJOEO/rMsXSoVybxIn5bg+
97CFCtAdsRRjAJZRQcrJIztenGJX81U0rvAX+OuoSNrgVpdxcwuH/1x80i+CY5kdUkg3EMkU0m4F
iNQ6CyXiiimVSBRB0sHfWW5/Em+qlYeRjrXCyJBYPo2mCuMtqQN42VeShEkQ1XPx6o09NTaaxXRM
pV2IHjzALLrR0Px6zqbp7CuEhPdLxTYcXetDKQJt/XHJuWdvETMgsJnyQ0cCJSPXp21xsrK6zYLY
cQ1M8rS9RHCmWvFdTZg49mX3QPANOUDPoPR0y3mOT19FWKFYOfQhHN2xPJZAPV6ZJeReeAeTBRkT
vgIJE/3BsQpmqsSusjgEDYCrK8MfaybAC6CpE5ZKnQwV99YlTcbPx7vVKPuU13j3Aj4FjtGjkFun
fCOpLX1AA0FSbBfOOCVeYd94bCGaW8f+j3NBB+29ELYMskew2tyCBiw2HodBrMoDiYVWHbD+bWw8
qMOOBurEQihVdNq5Tbi3R2fnnX9DpfbVljJeKFjyVwCLZA5OdGIYPuJxrXGKsaBI+abTgciL4n4W
bsG7LalURKCMe/HH1jVuy8VjevwJMB+u7CHoOc9jVCwR4YSPjH9fbxcIn9UIuECm1CryEUYB6kkh
BEeyxdQclP0amPYlyxVU80KN+Mxvle4//B6kwUtjS+/Rv943oXrXxaLXTCpeds49x0FWSRo/HCxy
nunzpkyqD4wBfUyB7hYggeRUaCb7aNVuIB1QZSY9EqF3F26Aootz1cYpr1CBtizZK9Q6Ez6N3iYW
1dMUB7dsNp4a4emAU3CfhHYh3JNv4pD21PbPASO/t89v7uMDrsI8SOp1nHqV0hYG2+JhxNyhYKV1
oXv54mKzbW+4vwsU/ySrrexUvmkTzLCsYBI7nSZT5uVprRA+MQJBLx6dKVVuzO1x8hzTv9T2LvJr
7rpd6Ban94JJ8vG7OUO0OaNP9HDrz+34xmCqQRi/f0TkmfSo4uFcsIfAmdQVbd6uu22ZBoWqolaz
lBXjt5Oe2AQV51Zma53dlArSBLpvbg/RoMM7cMhnGn33DkSBDYU9rN2iApw0zswa/KJ/p1r33Jrk
5YTL6wTTEuaG+UxVrtCxX4VHk7syaOjI5dshRELos3ZeIJeqKAgS45H6cK+gjCq0l3qWDDnFHcGm
```



```
zYoP1665lC7c9TOs8i5OBLM6hGqgEgcKEiTIp+tRUvDEhyN1v/YngT3izvWbsijV0QTTJcjsyFWq
DSJiw8G1WH4oFqZAzF2UzE6fzEeQbMVlPPxlnpUjipTqdtWcuayLH7tifX7diBl1fjlUOTqPK2+5
vz1HckVtzJMS4g0W7rWHAbTv5nfrby/1IJBHMDutjI2dh6J7nXbSgFOiT98TFL7upJCNc7T3AH4j
Ro1TzzXqODFShamQeYlooCyvStQxqj08rz74+7ery+GapNEPL4cPZ1qV0bfKCBwOQrTV8lIZXsFt
Jj9TV+7lT6ZcePnCFY6pWI78u5WWePZunmI9FFhz+odZd4vfh0C3VEISmeEN28T8XdvtHt8A78sr
4/SmrPteZpZhByZe2n50ZHQU+ukncDgZirtz5A4LIBedcDLCgeNfonHYCQTNYOKoDA+eq5sBczKP
mqFKjPnBq1533/lptWhsgou8CZfsEaY4kZvEzK8YTVrfVt4T407A851vKxBfHIyXKxFFi17Yddr2
SiqebAUjT3waPAoUwgdJelDYTnnKUQy0Zm25gGRDiE9LUwoOp7ys0H9m/xXJROx76gbljguU3ad9
fcwQIm8RTKZvXvKrVRBUsHutEL6/qZAb5VBQ1JHsa4tknAFTdwh7lsB1/l0lHtZ+HzBdgZ8kOvRm
HiCKYb+2p26WMVNy8SRhW8EeYxx3t79LMU3pIp9w4rCnuClwAYAXN6PP1Gf5GgsGS228ur3vwNKO
8YZIdMatmKJDy8Ufkm1Ljvy4Z0/3+XcGLDWyxRx6M2mLvMPvJIz9iGSr684PRfSydR3nq6W7gwYc
Ohb62cmSLVWyECoaa+cqVFFGOKHcUT3ZS7Xlx0QkniCQI9d46XDEx64PFGeBXL/z4dj7ZYx6woX9
R+F5yOAdKoILV5N9m4xzauPO4EkmKakDBtsf9tzExrArDBoT664Xc7cVJ/2jTzX57Oms09Q7r+T8
hH0JNxhcXAqhxdbMitkcFSy7t0pBgrPXRhdXohbGlhuZPAOMVkWWDMf8x7Yc4k7F3l9ua67w5Z2Q
cDf8NBq5iYM3TkB+2qpmn16L7Pbp5qlAoIcB409+6VwxHiHQgBHOPGsP1xHNYGYyKcfR4VxaUUXf
5G18b5NOnx3S2VCBA9fJGXlHqW3RmtlMEP4dEQdCbhH7jw7jd5El0NabRA0fCBTAYR61vYa9Ov7S
DOIefy6NpDffg9sFltOa36ag==</xenc:CipherValue>
                </xenc:CipherData>
             </xenc:EncryptedData>
          </wst:RequestedSecurityToken>
       </wst:RequestSecurityTokenResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

6.4.1.1 Retrieval of Encryption Public Key

The STS shall encrypt the token using the Relying Party's public key. This constraint is caused by the public-key encryption of SAML token, which entails that only one defined entity is able to decrypt the delivered SAML token (the one that owns the private key associated to the encrypting public key). In order to afford multiple Relying Parties, the STS shall be able to encrypt the SAML token with one selected public key, chosen among a set of multiple registered public keys.

The target Relying Party is known by the Client of the STS: it is the SP entity to which a service request shall be addressed. This information should be conveyed, from STS Client to STS, on the optional AppliesTo element of the RST, which contains a WS-Address (see Annex B).

The STS shall use a keystore containing at least one default public key and an unlimited set of public keys associated the WS-Address of each Relying Party. The rule used by the STS to choose the public key is then based on the AppliesTo element of the received RST:

- *if the AppliesTo element is absent*, then the public key used for encryption shall be the default public key registered on the STS;
- *if the AppliesTo element is present*, then the public key used for encryption shall be the public key of the specific relying party associate to the WS-Address specified in the AppliesTo element; if the WS-Adress is unknown from STS then the RST fails and a fault shall be reported to the requester (see 7.1.4).

Note that the first case is specially tailored for architectures having one (or one main) Federating SP. In this case, the STS should simply be configured with the public key of this Federating SP entity; then the clients should get the right SAML token without having to specify an AppliesTo element. In such context, the STS implementation could leave out the treatment of AppliesTo element but, then, it is recommended that STS reports a fault to the requester if the appliesTo element is present (instead of silently ignore this element).

6.4.2 Signature / Message Digest

The SAML token is signed before it is encrypted. The signature process is characterized by the following statements:

- The secure hash SHA-1 digital signature message digest algorithm is used, as supported by [NR15].
- The element that is signed is the top-level SAML Assertion, i.e. `<urn:oasis:names:tc:SAML:1.0:assertion:Assertion>`.
- The signature is put as an "enveloped signature" method, which means that the signature element is embedded as a child of the afore-mentioned SAML Assertion element.
- No certificate is put in the signature. This means that the PEP verifying the signature has to know (from its keystore, for example) the public key of the IdP that produced the SAML token.
- A canonicalization method shall be used which eliminates namespace declarations that are not visibly used within the SAML token. This shall apply for both
    - SignedInfo element, specified in `Signature/SignedInfo/CanonicalizationMethod/@Algorithm`
    - and actual element to be signed, specified in `Signature/SignedInfo/Reference/Transforms/Transform/@Algorithm`

  A suitable algorithm is ”Exclusive XML Canonicalization” which is implemented through a digital signature declaration:

  `Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"`

  Note that the specified canonicalization algorithm omits the comments.
- The URI attribute of the <ds:Reference URI="..."> element shall refer to the <saml:Assertion> node being signed (using XPointer, see 4.3.3.3 in [NR18]. The XML pattern is as follows:

```
<saml:Assertion … AssertionID="xxxx" … >
   …
   <ds:Signature …">
      <ds:SignedInfo>
         …
         <ds:Reference URI="#xxxx">
            …
         </ds:Reference>
      </ds:SignedInfo>
      …
   </ds:Signature>
</saml:Assertion>
```

The XPointer format, used in AssertionID and reference URI, shall comply with [NR18]; it is not additionally constrained by the present Best Practice document.

Note that the present Best Practice only enforces the signature of SAML token, which is put in the SOAP body of RSTR and in the SOAP header of service request. Other digital signatures on the remaining elements of SOAP messages, which may be required by interfaces of Service Providers, are permitted but these are out of the scope of the present Best Practice.

The example below uses the user attributes listed in Annex D.

Example: signed token before encryption.

```
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
AssertionID="oracle.security.xmlsec.saml.Assertion1955a65"
IssueInstant="2009-06-25T13:34:55Z" Issuer="http://earth.esa.int"
MajorVersion="1" MinorVersion="1">
           <saml:Conditions NotBefore="2009-06-25T13:33:55Z"
NotOnOrAfter="2009-06-25T13:39:55Z"/>
           <saml:AuthenticationStatement AuthenticationInstant="2009-06-
25T13:34:55Z"
AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
                       <saml:Subject>
<saml:NameIdentifier>dail</saml:NameIdentifier>
                                   <saml:SubjectConfirmation>
<saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:bearer</saml:Confirm
ationMethod>
                                   </saml:SubjectConfirmation>
                       </saml:Subject>
           </saml:AuthenticationStatement>
           <saml:AttributeStatement>
                       <saml:Subject>
                           <saml:NameIdentifier>DAIL42</saml:NameIdentifier>
                                   <saml:SubjectConfirmation>
<saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:bearer</saml:Confirm
ationMethod>
                                   </saml:SubjectConfirmation>
                       </saml:Subject>
                       <saml:Attribute AttributeName="Id" >
<saml:AttributeValue>DAIL42</saml:AttributeValue>
                       </saml:Attribute>
                       <saml:Attribute AttributeName="c" >
<saml:AttributeValue>Italy</saml:AttributeValue>
                       </saml:Attribute>
                       <saml:Attribute AttributeName="o" >
<saml:AttributeValue>ESA</saml:AttributeValue>
                       </saml:Attribute>
                       <saml:Attribute AttributeName="ProjectName" >
                                   <saml:AttributeValue>HMA
imp</saml:AttributeValue>
                       </saml:Attribute>
                       <saml:Attribute AttributeName="Account" >
<saml:AttributeValue>dailsp</saml:AttributeValue>
                       </saml:Attribute>
                       <saml:Attribute AttributeName="ServiceName" >
<saml:AttributeValue>catalogue</saml:AttributeValue>
                       </saml:Attribute>
           </saml:AttributeStatement>
           <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                       <ds:SignedInfo>
                                   <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                                   <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                                   <ds:Reference
URI="#oracle.security.xmlsec.saml.Assertion1955a65">
                                               <ds:Transforms>
                                                           <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                                                           <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                                               </ds:Transforms>
                                               <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<ds:DigestValue>nLkuqyqDggsxQnPiGzVDDckxaA0=</ds:DigestValue>
                                   </ds:Reference>
```



```
                        </ds:SignedInfo>
<ds:SignatureValue>oOkdc3KB2HwPB6YzhEa9MHx5yo1u/xqHp81wPj68uf5Ypet/5wHHEvfuN
hxD+S3ejT2f4lKIGkVDcsRNyUqaAn60CnJiN4RBpwcjcWQSUj5/XxesaR4nO4CtDylaLV6acLwww
lLN5PQ66UumASE=
                        </ds:SignatureValue>
            </ds:Signature>
</saml:Assertion>
```

The security model proposed requires that the case of RST is further decomposed into three cases as described in the following section.

6.4.3 Authentication Use Cases

In the present section, we describe three use cases, which refine the two authentication cases that have been introduced in section 6.1.

The first two use cases assume that the STS *is* in charge of authentication (case 1.a in 6.1):

1. **STS as local IdP**: the STS performs authentication from local user registry;
2. **STS as Federating IdP**: the STS relays authentication request to an external IdP.

The third use case assumes that the STS *is not* in charge of authentication (case 1.b in 6.1):

3. **STS with trusted IdP**: the STS does not perform authentication; it delivers security tokens to trusted clients, which rely themselves on an external, trusted, IdP using another authentication protocol; this case intents to make the present Best Practice interoperable with Web-SSO systems like Shibboleth (see section 8).

These three use cases are detailed in the following subsections.

6.4.3.1 STS as local IdP (Default Case)

In this use case, the **RST with password** is used; it contains no IdP identifier, so the authentication is performed locally by the STS itself.[4]

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
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/">
   <soapenv:Body>
      <wst:RequestSecurityToken>
         <wst:TokenType>
```

[4] For people having read versions of the present document anterior to 0.3.0, the use case shown here unifies former use cases 1 and 3. The previous approach made a distinction between "Federating IdP" (use case 1) and "External IdP" (use case 3), although the RST protocol was the same. Further analysis has shown that this distinction is not relevant for the scope of the present Best Practice.


```
    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>
          <wsse:UsernameToken>
             <wsse:Username>JohnDoe</wsse:Username>
             <wsse:Password>MyPassword</wsse:Password>
          </wsse:UsernameToken>
       </wst:RequestSecurityToken>
    </soapenv:Body>
</soapenv:Envelope>
```

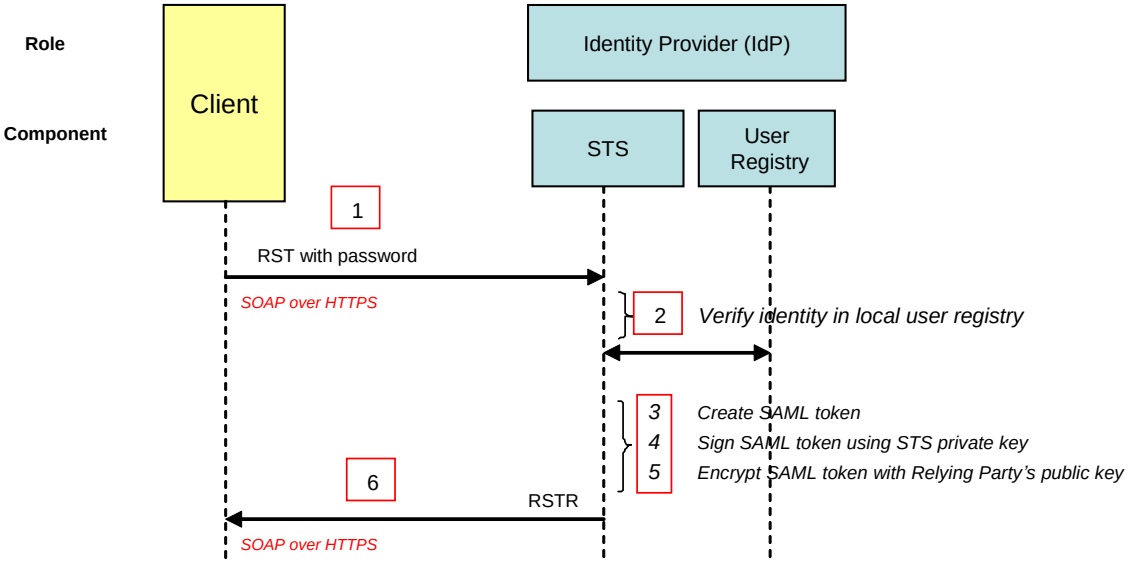

Figure 4 STS - local authentication (Default Case)

1. The RST with password is sent to the STS using SOAP over HTTPS.
2. The STS verifies the identity in the **local** user registry.
3. The STS creates a SAML token using the minimum profile attributes retrieved from the user registry. The SAML token is created containing assertion of the authentication and assertions regarding the attributes of the user.
4. The STS signs the SAML token using the STS private key.
5. The STS encrypts the SAML token with the Relying Party's public key (see subsection 6.4.1.1 for the process of key retrieval).
6. The RSTR containing the encrypted and signed SAML token is returned to the Client using SOAP over HTTPS.

The client is unable to decrypt the content of SAML token present in the received RSTR; only the Relying Party can decrypt the SAML token (using its private key).

6.4.3.2 STS as Federating IdP

In the present use case, the **RST with password** is used; it contains an identifier for the STS of a given external entity *n*. The STS acts here as a Federating IdP that relies on another IdP to perform the actual authentication. The relation table between identifiers and external entities STS URL shall be stored on the server and configured at service deployment time. It must be done in this way for security as the system must deny access to un-trusted IdP.

Example RST with external IdP specified:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
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/">
    <soapenv: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>
            <wst:DelegateTo>
                spot-image
            </wst:DelegateTo>
            <wsse:UsernameToken>
                <wsse:Username>JohnDoe</wsse:Username>
                <wsse:Password>MyPassword</wsse:Password>
            </wsse:UsernameToken>
        </wst:RequestSecurityToken>
    </soapenv:Body>
</soapenv:Envelope>
```


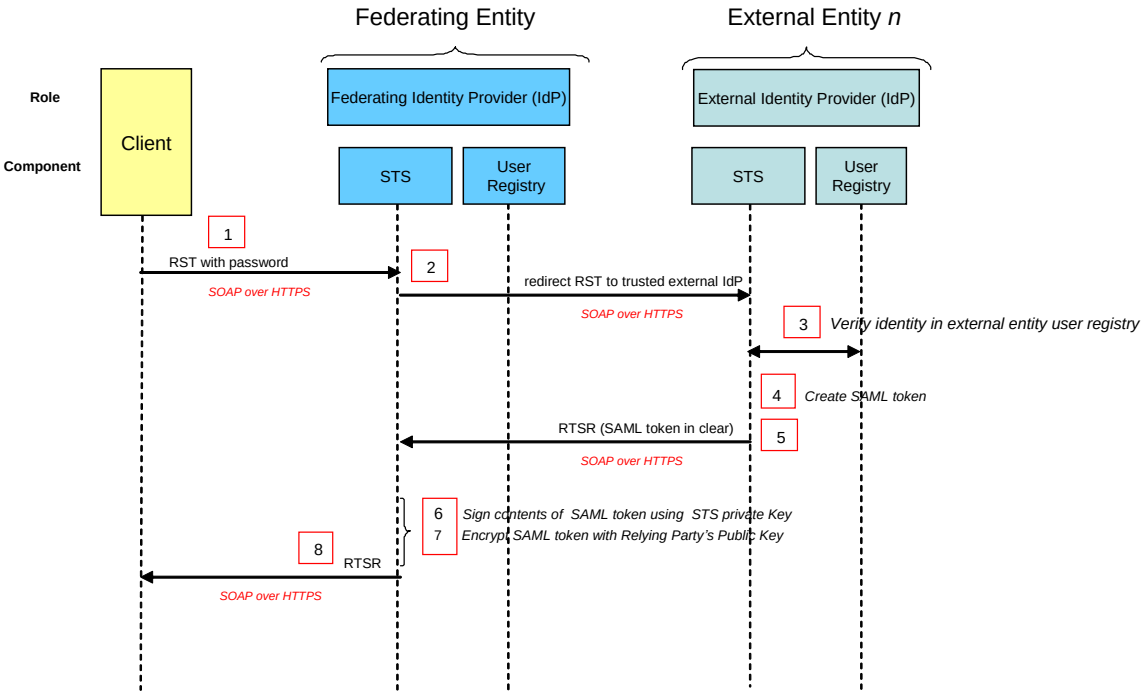

Figure 5 STS - external authentication

1. The RST with password is sent to the STS using SOAP over HTTPS.
2. The STS sees that an identifier of external STS is specified in the RST; it redirects the RST to the designated external IdP (called *federated STS* in the following). The URL of this IdP is extracted from the table previously described.
3. The federated STS verifies the user in the external entity user registry.
4. The federated STS creates the SAML token using the attributes retrieved from the user profile in the user registry.
5. The RSTR containing the SAML token, in clear, in the SOAP body is returned to Federating STS, through SOAP over HTTPS.
6. The STS signs the SAML token using the Federating STS private key.
7. The STS encrypts the SAML token with the Relying Party's public key (see subsection 6.4.1.1 for the process of key retrieval).
8. The RSTR containing the encrypted SAML token is returned to the Client.

Notes:

1. As for the previous use case, the client is unable to decrypt the content of SAML token present in the received RSTR; only the Relying Party can decrypt the SAML token (using its private key).

2. The confidentiality of the SAML token provided in clear by the external IdP is assured 1° by the HTTPS protocol, which encrypts the SAML response and 2° by assuring that the requester of the RST *is* the Federating STS, known in the circle of trust. Actually, about the last point, the rule is:

if the requester is the Federating STS,

then the SAML token is returned in clear *(present use case)*
else the SAML token is encrypted with the Relying Party's public key *(see first use case)*

The mechanism to identify the requester as a known Federating STS is left as an implementation decision. This could use WS-Addressing.
The rationale of this process is to support both Clients that access the Federating STS and Clients that access federated STS directly. Also, the system scales up seamlessly in the case of multiple Federating STS : the external STS should simply know a list of authorised Federating STS (instead of a single one) and check inclusion of the requester in this list.[5]

6.4.3.3 STS with trusted IdP

We cover here the case where there is an external IdP in an external security domain, which does not comply with the present Best Practice but which is trusted by the STS. This use case is meant to make the present Best Practices interoperable with Web-SSO systems like Shibboleth. For instance, ESA UM-SSO, an SSO system based on Shibboleth, defines a specific security domain with its own IdP (see section 8).

In this present case, the **RST with signature** is used. (RST contains no password).

In order to integrate such external IdP, a trust relationship shall be established between the two security domain such that any user that has been authenticated by the external IdP shall be able to get the SAML token.

In order to establish a trust relationship between the two security domains, a given Client[6] *C* of external security domain shall provide its public key to the Federated IdP. The trust relationship between *C* and STS is established as soon as the STS security administrator has registered this public key in the keystore of STS. From that point, the client *C* can obtain SAML token for any users authenticated on external IdP by issuing RST with signature.

[5] Note that several variant mechanisms are feasible, if we allow the inclusion of *multiple SAML tokens* in the RSTR and/or service requests. A client could then own several tokens for the same user at a given time, encrypted with different public keys and potentially carrying different contents. The PEP should then be given several "chances" (one per included SAML token) to succeed in decryption and to authorise a request. These variant mechanisms change the interfaces defined in the present version of the specification and, therefore, are no more than a subject of investigation.

[6] It is important to remind here that the « client » meant here *is not* a front-end application, like a Web browser ; it is the middle-tier entity that issues the RST to the STS, like a Portal server (see last paragraph of 6.3).

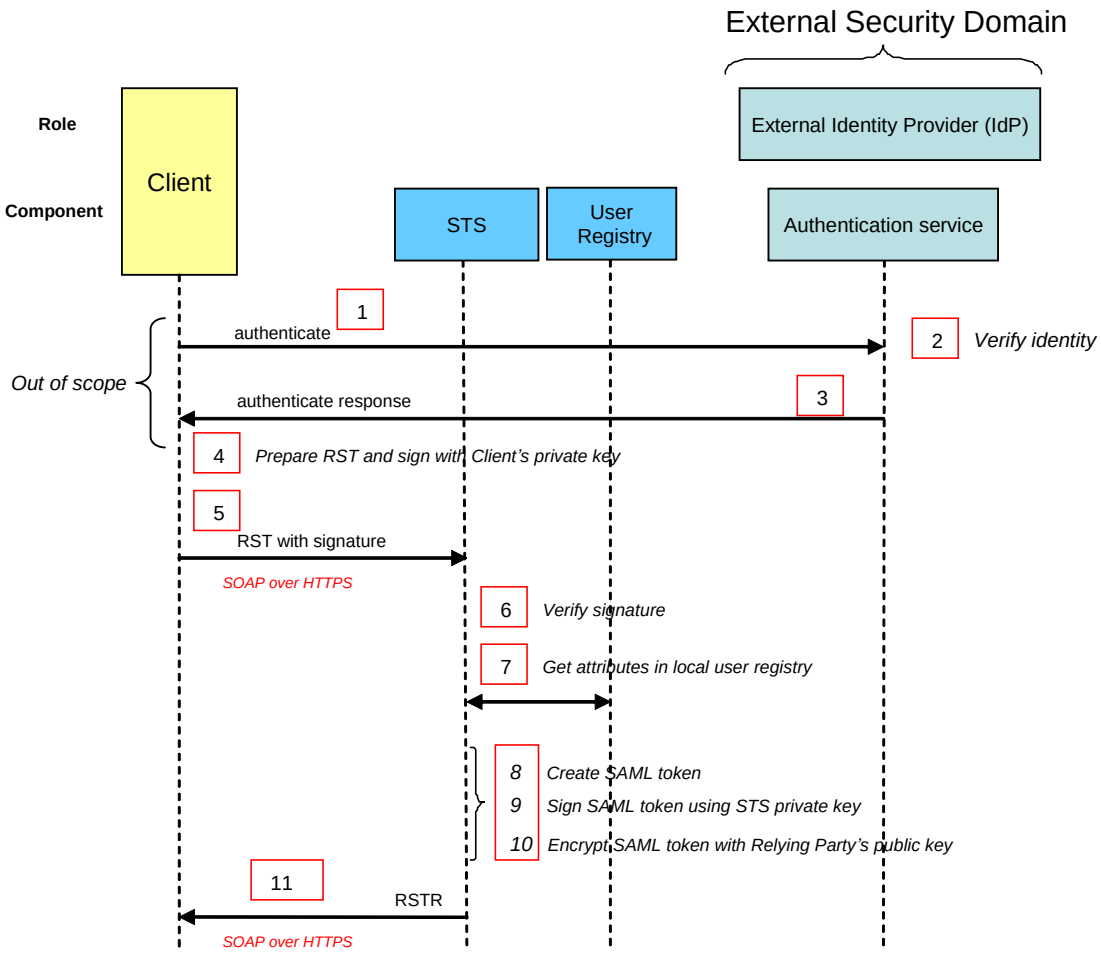

Figure 6 STS – No authentication

1. An authentication request is sent to the external IdP (out of scope of the present Best Practice).
2. The external IdP verifies user identity (out of scope of the present Best Practice).
3. The successful authentication response is returned to the Client (out of scope of the present Best Practice).
4. The Client prepares an RST with user id (no password) and signs it with its private key.
5. The RST with signature is sent to the STS using SOAP over HTTPS.
6. The STS verifies the signature of the RTS, based on the set of registered client's public keys (stored beforehand in STS keystore, as trusted Clients); this succeeds.
7. The STS retrieves user attributes from local user registry.
8. The STS creates a SAML token. The SAML token is created containing assertion of the authentication and assertions regarding the attributes of the user.
9. The STS signs the SAML token using the STS private key.
10. The STS encrypts the SAML token with the Relying Party's public key (see subsection 6.4.1.1 for the process of key retrieval).

11. The RSTR containing the encrypted and signed SAML token is returned to the Client using SOAP over HTTPS.

6.4.4 Service Request

The service request may contain an encrypted SAML token in the WS-Security element of the SOAP header. This SAML token is obtained from an RST as previously described and is used to control access to services.

N.B. It is not mandatory that the service request is preceded by an RST, as the SAML token is not mandatory in the service request. However, access to services is controlled by the policies applied in the PEP.

Since a specific SAML token protocol is required to access the protected Web Services, the use of WS-Policy [NR20] is recommended for the WSDL files describing these Web services. The WS-Policy elements are used to formally specify the presence of SAML token in SOAP header, the encryption algorithm, etc. With such dispositions, the Web services are self-describing, allowing for "discovery" by clients, hence improving the interoperability of the system. An example of WSDL using WS-Policy is provided in annex F.

The access policies applied in each PEP, based on the SAML token, are out of scope of the present Best Practice. However, to help understanding, several examples of authorisation rules along with their XACML counterparts are provided in section 9.

6.4.5 Extension Points

6.4.5.1 SAML 2.0

This Best Practice uses SAML 1.1 [NR10] as baseline. However, SAML 2.0 will be supported in the future. To be conservative and backward-compatible, the STS should be able to deliver SAML token formatted in a given SAML version (1.1 or 2.0), specified in the RST. For this purpose, the standard WS-Trust TokenType element of RST shall be used (see schema in Annex B). More precisely, the format of returned token shall be SAML 1.1 or SAML 2.0 depending of the value of TokenType, respectively,

```
http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV1.1
```

or

```
http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV2.0
```

which are standard identifiers specified in SAML Token Profile [NR11].

7 Interface

7.1 Request Security Token

The Request Security Token (RST) operation, as defined in WS-Trust 1.3 [NR23], allows clients to retrieve authentication metadata from a nominated IdP server. The response to an Authenticate request should be an XML document containing authentication metadata about the authentication and requestor.

7.1.1 Request

Protocol: SOAP over HTTPS

7.1.2 XML encoding

As explained in 6.4, we make a distinction between RST with password and RST with signature.

The following XML-Schema fragment defines the XML encoding of the message body of the RST with password.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
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/">
    <soapenv: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>
            <wsse:UsernameToken>
                <wsse:Username>JohnDoe</wsse:Username>
                <wsse:Password>MyPassword</wsse:Password>
            </wsse:UsernameToken>
        </wst:RequestSecurityToken>
    </soapenv:Body>
</soapenv:Envelope>
```

Figure 7: Example of RST with password

The following XML-Schema fragment defines the XML encoding of the message body of the RST with signature.

```
<S11:Envelope xmlns:S11="…" xmlns:wsse="…"
        xmlns:xenc="…" xmlns:wst="…">
  <S11:Header>
    <wsse:Security>
       <ds:Signature xmlns:ds="…">
          …
          <ds:Reference URI="#soapbody"/>
          …
       </ds:Signature>
    </wsse:Security>
  </S11:Header>
  <S11:Body Id="soapbody">
    <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>
        <wsse:UsernameToken>
```



```
          <wsse:Username>JohnDoe</wsse:Username>
        </wsse:UsernameToken>
    </wst:RequestSecurityToken>
  </S11:Body>
</S11:Envelope>
```

Figure 8: Example of RST with signature

7.1.3 Response

The following XML shows an example of response, which is a Request Security Response (RSTR), as defined in WS-Trust 1.3 [NR23], containing an encrypted SAML token.
The SAML Token is always encrypted with the Federating Entity public key i.e. in both the use cases the client receives the same response:

- the federated response message to the Federating Entity STS and coming from an external Idp.
- The federated response message returned by the Federating Entity STS to a client.

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
 <soapenv:Body>
<RequestSecurityTokenResponse xmlns="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/" xmlns:wsa="http://www.w3.org/2005/08/addressing"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
          <TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-
profile-1.1#SAMLV1.1</TokenType>
          <RequestedSecurityToken>
             <xenc:EncryptedData
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
Type="http://www.w3.org/2001/04/xmlenc#Element">
           <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
           <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
             <xenc:EncryptedKey>
               <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
               <xenc:CipherData>

<xenc:CipherValue>cbE8viFOmyDuxR8N4EdwS9UUKpSoUbMrWSVprW7IypMwFZLeHR9Rxd4iw5
dUl4K+TffyNdRJ9Tr9PD8YIdpFLzCvYas63g5x4/XnyA1E2AU8ZBBpM2dtbr3g4IYMywfraWrI76
mHM+MERVZdHMVBWFrhqXhcS92m23m+amt14mk=</xenc:CipherValue>
               </xenc:CipherData>
             </xenc:EncryptedKey>
           </ds:KeyInfo>
           <xenc:CipherData>

<xenc:CipherValue>VEHlprDMQ+DqIpoPqx6TYi/mMX2dGV5JCJCrhDquZHRKqOiaIFfwqOMZvn
2HW2JDFvUxJ6LRTKdNujQI7sxc6h3IGBL7NXF7bx4jGwQ09wAA7nm6OoB4jiGdaqb8wTx0olnzn2
WqOWoVeTng1lwBi0rv2+iD1HWnXAUUHfJH8ALq4IU3hR0vjoqJH6Y21EuXPeXp/dYPUw3oIFn2FE
ID2u+8T+xOxbbq2ezQbU3z8n1LbgvDtN3ex5lUCo260pOOPn92nn7nYErT682eYd+bCKoiENpQSY
gHszvvyqFf9o6O0u87zk4AORWsRhQH74L2gG8wVOeHKyhEx0RsBkf4xZcQKBvQ9JHWQWpDEB51NZ
aJe1hSyaUk6T5gf9ArDnz6UwL0ZTDp6Dxgjha91u5qIMG3ECxVYKcnBv+O6Om1Q0HbL0ecbUDR56
evS+mf0U9JxduBKwFJLqta6D0wmwqYWcaF3ZrKd7SatV8Z2l0DmWTMe5R+x6O1RpbKltlduKl4bL
aSYFpaqaU758ZsmTDmjQQj8fn1qCZbDtp4SEVPWumoTg2k7RAOay2QtV5b+VA9wloSXoxVf2csLS
OOH/NDE1noBIpzgUb9Xm/YIPwikQKsxNPFM72yLrS0vjAho1Cxrg+8l7XIVcmowhPnLqSs6ZpvA0
1YP8EhsOFlN+0y+9EfAuoY4jYcScfwqDehth76lER+EyAdFLi10VhVxKW14VLbmksAydndIQaw6V
zGm1Qwoc3CeCaeq4q4GgFgiem1BmW9IeBaUBTX2wZmIKG8Z9Xhjv6MwT7hOeWH5fefipJs8JS8l6
wQBo8WAczzmw6s1j8JW9YDyAWosfoTPrtOwFTaaYSiaEXvPOnb5RgR/W4ivZ64ioA8FXyLFoWcNE
JJ6AgWHDLAbCDg/zvnVwEs7OdaSxRTxVNsc7cpclGspSmk/HzGYxPHInGhn/QPsac5iN6t6HlwnQ
UJgt81rI/tbFfSYqqtYqXKeNoEtw91/1DZVUi7mSc7Xj2e2Wb65h8PIoYeX3Nli+i4SrOoeAKaZr
```



```
HtpqP6f+pI4lpkANS4RFxFDiL9Ddxv1WKD//nMck0Su0HfIbPYUYF0GGvlHsv6IiwT8dj/f0MnCx
kAgegliGageZthQiNavOcURRC/94d+1jDZGayowurzdxmJhxyiEY5REQQt3hK4aAD89wMjndzxHd
tcQEuvXA5uSm3T9qgIm4Qdvuh54PW/SKptG9fDj4paTxVVlfZ+0f/1Vxjj4pPIKOVjE3e4ChBPkJ
XD/nXqZ8DdR+zPxOLWYyiqnMaxv3OInd/Iz2Lq36a09b0JEfMVz4e39sGtFzNDbxXgQnTx4L3jDY
Fdl5+gelUNduK9HtgklXDWfNIMWtY5xhdTX0m3Q8hBtNgKOHeg7BcBXF+uT30mqwgJu5cbJQl/1j
/QlMvromUaUQATN2ULu7mMiTWkoYoMTiijJGAizbKIi6O5xmHvF/jicd7lBcmSz+B+BnrnqxY5DM
/qQSFsnRoGmPKlJeiao2g+QuMD5x7H+pBUIq3B8lkMlUBg5VoKx2+kCHuP2lamGFskQl80PRGygQ
5adA2iWwKoIKoCdcYIc9C2sPVkJz+slExJXizl4L51GEWDlQ8VGsqNV7CzOyIt0uXIIBQW3j0aX/
/7QoYVfM681TiqvtaDEY7Ip4nSV839e5xnj3s+qgzXOpoK5rw5EThDLhthPy97CJiuSbsfcGf1Dv
WNE74x2E4b8HazacItRBIbx0GFDHIoqaHEih6zlhQaqwloLnUHRpL8vAQLVKiW3q94569e3GenoQ
bpjxKQ9F58VuQh3ZiZtJ+17XOxDx6ZDXcTiQDa+3nXiTgT7k9gGtpIv8vYLMuUHDEZx1zGd/rMzU
3JAbbfKO8+Cs3pMnb2KpGL4rLLgivee1P35rbHK04V8D0NbwDk0TOVnmFQWIRsgVtPwmEHXbFl3j
qIUTx4xdishXmkKcDCwaNfqYo4yzmgLUlbtchkDGF9YBA0mXv4gT7z0TiBb1jUFBhTnciL4DBkI9
K8wEjklHUi7w4LjvhCB5B15y1ZG/baIscoRrVu41HUp7crwbsdKjcWGE//dBTXN1vrXDm0maAkCo
nuYNPpMY0Cf+ikVITJO73UaXplFjOOtm+mkql6e5nTd8qGQXwHZl/nGJOEO/rMsXSoVybxIn5bg+
97CFCtAdsRRjAJZRQcrJIztenGJX81U0rvAX+OuoSNrgVpdxcwuH/1x80i+CY5kdUkg3EMkU0m4F
iNQ6CyXiiimVSBRB0sHfWW5/Em+qlYeRjrXCyJBYPo2mCuMtqQN42VeShEkQ1XPx6o09NTaaxXRM
pV2IHjzALLrR0Px6zqbp7CuEhPdLxTYcXetDKQJt/XHJuWdvETMgsJnyQ0cCJSPXp21xsrK6zYLY
cQ1M8rS9RHCmWvFdTZg49mX3QPANOUDPoPR0y3mOT19FWKFYOfQhHN2xPJZAPV6ZJeReeAeTBRkT
vgIJE/3BsQpmqsSusjgEDYCrK8MfaybAC6CpE5ZKnQwV99YlTcbPx7vVKPuU13j3Aj4FjtGjkFun
fCOpLX1AA0FSbBfOOCVeYd94bCGaW8f+j3NBB+29ELYMskew2tyCBiw2HodBrMoDiYVWHbD+bWw8
qMOOBurEQihVdNq5Tbi3R2fnnX9DpfbVljJeKFjyVwCLZA5OdGIYPuJxrXGKsaBI+abTgciL4n4W
bsG7LalURKCMe/HH1jVuy8VjevwJMB+u7CHoOc9jVCwR4YSPjH9fbxcIn9UIuECm1CryEUYB6kkh
BEeyxdQclP0amPYlyxVU80KN+Mxvle4//B6kwUtjS+/Rv943oXrXxaLXTCpeds49x0FWSRo/HCxy
nunzpkyqD4wBfUyB7hYggeRUaCb7aNVuIB1QZSY9EqF3F26Aootz1cYpr1CBtizZK9Q6Ez6N3iYW
1dMUB7dsNp4a4emAU3CfhHYh3JNv4pD21PbPASO/t89v7uMDrsI8SOp1nHqV0hYG2+JhxNyhYKV1
oXv54mKzbW+4vwsU/ySrrexUvmkTzLCsYBI7nSZT5uVprRA+MQJBLx6dKVVuzO1x8hzTv9T2LvJr
7rpd6Ban94JJ8vG7OUO0OaNP9HDrz+34xmCqQRi/f0TkmfSo4uFcsIfAmdQVbd6uu22ZBoWqolaz
lBXjt5Oe2AQV51Zma53dlArSBLpvbg/RoMM7cMhnGn33DkSBDYU9rN2iApw0zswa/KJ/p1r33Jrk
5YTL6wTTEuaG+UxVrtCxX4VHk7syaOjI5dshRELos3ZeIJeqKAgS45H6cK+gjCq0l3qWDDnFHcGm
zYoP1665lC7c9TOs8i5OBLM6hGqgEgcKEiTIp+tRUvDEhyN1v/YngT3izvWbsijV0QTTJcjsyFWq
DSJiw8G1WH4oFqZAzF2UzE6fzEeQbMVlPPxlnpUjipTqdtWcuayLH7tifX7diBl1fjlUOTqPK2+5
vz1HckVtzJMS4g0W7rWHAbTv5nfrby/1IJBHMDutjI2dh6J7nXbSgFOiT98TFL7upJCNc7T3AH4j
Ro1TzzXqODFShamQeYlooCyvStQxqj08rz74+7ery+GapNEPL4cPZ1qV0bfKCBwOQrTV8lIZXsFt
Jj9TV+7lT6ZcePnCFY6pWI78u5WWePZunmI9FFhz+odZd4vfh0C3VEISmeEN28T8XdvtHt8A78sr
4/SmrPteZpZhByZe2n50ZHQU+uknc DgZirtz5A4LIBedcDLCgeNfonHYCQTNYOKoDA+eq5sBczKP
mqFKjPnBq1533/lptWhsgou8CZfsEaY4kZvEzK8YTVrfVt4T407A851vKxBfHIyXKxFFi17Yddr2
SiqebAUjT3waPAoUwgdJelDYTnnKUQy0Zm25gGRDiE9LUwoOp7ys0H9m/xXJROx76gbljguU3ad9
fcwQIm8RTKZvXvKrVRBUsHutEL6/qZAb5VBQ1JHsa4tknAFTdwh71sB1/l0lHtZ+HzBdgZ8kOvRm
HiCKYb+2p26WMVNy8SRhW8EeYxx3t79LMU3pIp9w4rCnuClwAYAXN6PP1Gf5GgsGS228ur3vwNKO
8YZIdMatmKJDy8Ufkm1Ljvy4Z0/3+XcGLDWyxRx6M2mLvMPvJIz9iGSr684PRfSydR3nq6W7gwYc
Ohb62cmSLVWyECoaa+cqVFFGOKHcUT3ZS7Xlx0QkniCQI9d46XDEx64PFGeBXL/z4dj7ZYx6woX9
R+F5yOAdKoILV5N9m4xzauPO4EkmKakDBtsf9tzExrArDBoT664Xc7cVJ/2jTzX57Oms09Q7r+T8
hH0JNxhcXAqhxdbMitkcFSy7t0pBgrPXRhdXohbGlhuZPAOMVkWWDMf8x7Yc4k7F3l9ua67w5Z2Q
cDf8NBq5iYM3TkB+2qpmn16L7Pbp5qlAoIcB409+6VwxHiHQgBHOPGsP1xHNYGYyKcfR4VxaUUXf
5G18b5NOnx3S2VCBA9fJGXlHqW3RmtlMEP4dEQdCbhH7jw7jd5El0NabRA0fCBTAYR61vYa9Ov7S
DOIefy6NpDffg9sFltOa36ag==</xenc:CipherValue>
                </xenc:CipherData>
              </xenc:EncryptedData>
            </RequestedSecurityToken>
            <wsp:AppliesTo/>
        </RequestSecurityTokenResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

Figure 9: Example of RST Response

7.1.3.1 Example SAML Token Before Encryption

An example is given here for completeness of the fragment before encryption:

```
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
AssertionID="oracle.security.xmlsec.saml.Assertion1955a65"
```



```
IssueInstant="2009-06-25T13:34:55Z" Issuer="http://earth.esa.int"
MajorVersion="1" MinorVersion="1" >
          <saml:Conditions NotBefore="2009-06-25T13:33:55Z"
NotOnOrAfter="2009-06-25T13:39:55Z"/>
          <saml:AuthenticationStatement AuthenticationInstant="2009-06-
25T13:34:55Z"
AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
                    <saml:Subject>
<saml:NameIdentifier>dail</saml:NameIdentifier>
                              <saml:SubjectConfirmation>
<saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:bearer</saml:Confirm
ationMethod>
                              </saml:SubjectConfirmation>
                    </saml:Subject>
          </saml:AuthenticationStatement>
          <saml:AttributeStatement>
                    <saml:Subject>
<saml:NameIdentifier>DAIL42</saml:NameIdentifier>
                              <saml:SubjectConfirmation>
<saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:bearer</saml:Confirm
ationMethod>
                              </saml:SubjectConfirmation>
                    </saml:Subject>
                    <saml:Attribute AttributeName="Id" >
<saml:AttributeValue>DAIL42</saml:AttributeValue>
                    </saml:Attribute>
                    <saml:Attribute AttributeName="c" >
<saml:AttributeValue>Italy</saml:AttributeValue>
                    </saml:Attribute>
                    <saml:Attribute AttributeName="o" >
<saml:AttributeValue>ESA</saml:AttributeValue>
                    </saml:Attribute>
                    <saml:Attribute AttributeName="ProjectName" >
                              <saml:AttributeValue>HMA
imp</saml:AttributeValue>
                    </saml:Attribute>
                    <saml:Attribute AttributeName="Account" >
<saml:AttributeValue>dailsp</saml:AttributeValue>
                    </saml:Attribute>
                    <saml:Attribute AttributeName="ServiceName" >
<saml:AttributeValue>catalogue</saml:AttributeValue>
                    </saml:Attribute>
          </saml:AttributeStatement>
          <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                    <ds:SignedInfo>
                              <ds:CanonicalizationMethod
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                              <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                              <ds:Reference
URI="#oracle.security.xmlsec.saml.Assertion1955a65">
                                        <ds:Transforms>
                                                  <ds:Transform
Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
                                                  <ds:Transform
Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                                        </ds:Transforms>
                                        <ds:DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
<ds:DigestValue>nLkuqyqDggsxQnPiGzVDDckxaA0=</ds:DigestValue>
                              </ds:Reference>
                    </ds:SignedInfo>
<ds:SignatureValue>oOkdc3KB2HwPB6YzhEa9MHx5yo1u/xqHp81wPj68uf5Ypet/5wHHEvfuN
hxD+S3ejT2f4lKIGkVDcsRNyUqaAn60CnJiN4RBpwcjcWQSUj5/XxesaR4nO4CtDylaLV6acLwww
lLN5PQ66UumASE=
                    </ds:SignatureValue>
          </ds:Signature>
</saml:Assertion>
```


7.1.4 Failed Request Security Token

If the RST cannot provide the RSTR due to a failure (failed authentication, invalid signature, invalid parameters, resource unavailable, etc), then the SOAP Fault mechanism shall be used, following the recommendation of WS-Trust 1.3 for error handling (see section 11 of [NR23]).

An example is given below, for the case of a failed authentication:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust" >
  <soapenv:Body>
    <soapenv:Fault>
      <faultcode>wst:FailedAuthentication</faultcode>
      <faultstring>Authentication failed</faultstring>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

7.1.5 WSDL

The WSDL is given below for the Security Token Service, without the Bindings and Services elements. This WSDL have been obtained by updating reference files from WS-Trust 1.3: http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3.wsdl

Note that this WSDL refers to the local schema file ws-trust.xsd, which is a restricted version of the standard WS-Trust schema http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3.xsd.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:tns="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/" xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512/"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.oasis-open.org/ws-sx/ws-trust/200512/">
    <wsdl:types>
        <xs:schema>
            <xs:import namespace="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/" schemaLocation="ws-trust.xsd"/>
        </xs:schema>
    </wsdl:types>
    <wsdl:message name="RequestSecurityTokenMsg">
        <wsdl:part name="request" element="wst:RequestSecurityToken"/>
    </wsdl:message>
    <wsdl:message name="RequestSecurityTokenResponseMsg">
        <wsdl:part name="response"
element="wst:RequestSecurityTokenResponse"/>
    </wsdl:message>
    <wsdl:portType name="SecurityTokenService">
        <wsdl:operation name="RequestSecurityToken">
            <wsdl:input message="tns:RequestSecurityTokenMsg"/>
            <wsdl:output message="tns:RequestSecurityTokenResponseMsg"/>
        </wsdl:operation>
    </wsdl:portType>
</wsdl:definitions>
```


Figure 10: Security Token Service WSDL

7.2 Service Request

The Client can send a service request to a PEP that shall authorise the access to a given service and, if authorised, shall relay this request to the end-point service (e.g. catalogue, programming, ordering services). The request is made using WS-Security containing the SAML token previously returned in the RSTR.

7.2.1 Request

Protocol: SOAP plus WS-Security over HTTP/HTTPS.

7.2.2 XML encoding

The SAML token (i.e. a ds:EncryptionData element extracted from the RSTR) shall be put in the SOAP header of the request, within the WS-Security element.

The following XML fragment defines the XML encoding of an example GetRecords request sent to a catalogue service.

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
   <env:Header>
      <wsa:MessageID
xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"/>
      <wsa:ReplyTo
xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing">
         <wsa:Address/>
      </wsa:ReplyTo>
      <Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd">
            <xenc:EncryptedData
Type="http://www.w3.org/2001/04/xmlenc#Element"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
               <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-cbc"/>
               <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                  <xenc:EncryptedKey>
                        <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
                        <xenc:CipherData>

   <xenc:CipherValue>k4kkm+nBkutOsmP9Lm6v4gpPvtJqxOOJLEOoKCQfE4Q7qp1yOBkKRlu
j9zb7Y07cNdJf8OhzGaHFGz7OIfM9Tpl5QEntkoeOT9wg/PsYqlIaAsCRoDsYjJoQrqpHdIpv3wl
Cck8iysQus4LpqdK
Hc6pWRk0Gk9022z/3U=</xenc:CipherValue>
                        </xenc:CipherData>
                  </xenc:EncryptedKey>
               </ds:KeyInfo>
               <xenc:CipherData>

   <xenc:CipherValue>+X0OFMae+FV8zOr0pPA02icglYf4AKcaml/jNfP8gdmjIh/dB/utVIC
YxKtarBRSAlptozGoI92r+bwUwmAyIY3D7gX0h6EC0P3LqhojKiMRrNbvCaotOPWoMherMp1SUbx
eYgxdZVlpXa77mNHHEkjhcmNXHydgz4DJoLxzHUIdWm9Lv+UTufH+D680JicO0SCnGdIC6KEpM68
h+3x/PRRNdmzy
QRpS9WZ03DAADBokRmW8IbG5YlUG4NjZDhkPDRlFhaHBTn4ZDP4LEd98KXZclwAlAB9XIICTeNFh
9t0itufclexX0zu/icAM8ws/sEAh7NmLyw+k8MRI7lMXeldnqftBYYj5NzYZqUd87XXqTe6ytnS3
```



```
SwbZXgdkgKylqdp0p0FcJVOGb4obfP/6irwzf1ujK2DMJb+9+mTQzfdNIIXimegV5wY2r1Wsg7Xt
xiVU6TFMI6VA5CH1MkmgYyYFqgI2MoiNXW3c3sgAs6+QlRoPMR3uNzvtB7NKy0m9BET+zqxCRgPt
GPstjX5ATvJ7tbcAlSKGyHubEIE9Am1Q2nGv3ChGzPPw+rwtow1D8xeSnxWOKpp/whmXcN9AEQ/z
5HtDCmbw1+ExRTM8Xy1NWp135If/6ooxcJtOf5vavo5Mx1OQto8TieF35+5FXA0rUiCn//yJzJRz
2mXEMJfoO1HfwPpfGWxwId4yuhWeylNAA4sKWt2OVDc/zkZpRTIHOoOWuut2LdsZS1fBZ+RMnpt/
u8tivsRITLyd2htTILLXKIenNpdRWeUD4d23RxcFFt+bGh
yrbHnsrT+DIZJD0PfS0zxigXu+NG4wy+Plhj11h4pn2AosIP5v0ZXN/tObgsQonwKyjFwgqGH8js
Ik/96PLnu1ODRRYVIBOGlcV9K7NrHeqCkM1157HCwu/rflTXK61jzRsZ8/hzC8ADiOXQnpk8K4EO
AFs/A6LY54A8MFQndHkAHN6NEK0nbAkqhOTur99PHrXQtYfsf26Mrd8rTKhkP5zd3pdfzvhnqOnx
OSe2FuWX3WHwUTgAzMBlJC2PRzHM4Q4q/pHFyK+UrLE2QLYsBn6VzouHfcI3dikR/0d2RQPsrQKQ
PB8WXMjJxK7v05jRBjZaNYpmsFk08zweM1OWuVB8L57zSzAfb7CKpPgsgRk4ezLWrPVK7Z2UuQ+z
/UH66S3a9dYsneQMaDMh7wYQtLe3fEeUhbBYrjRBZWHriOnhxoN7Rll8bnKOXoJUJJMZqyZyvN7q
HTLUxG98KUncu3HYwKSIgZEog
ZhHsfbqp9jee0tx4MhW+t8z9Upsh7TPhWcEvaFxpn5pfz4c704nsM7Tmcq4IijlnW8m3kX/mBR/O
TFcjWh2mV8XZoa/Hro7Rj69HVjELBTlF/W+S7pNsN+hoErRDlWxuHqC6v7KDMakaLF/Cekz
WA27ozxm7z1+6a/BeiXSTNojoodOybmv7hxjObWg53RUp5H3rejnN52+7IDHJwik2DONIErqL5GP
Soy3+adE5mSnAQklZ4zeErsl/A82ySgovaQNskuqa+6aBLvhHQ9CPeQmlddOfyU6HebMSN2mE8OZ
2yejwhujnT5ha4KoefrE21bwi5xWkNDobJbfBPXgg7Wdf4M6n3zsRTT6ixugXrkdRhnYyTzpRJ6E
PpL5Cdunh2gyQHiyJbcCh37rzTcsx0CiWHfak0ObqDRUeJ8tPyOS5PhyxcNknQ4p3RCI0QJTxUYG
3jpWntAK7ZU3d0Bmk+DAaPGGGJ44fKZ3HZTWjnFTcqWxwYOXxsiJ/kKE8ZVcDJ6IL4pf4dTnJHaa
8hT09LVutVZJrqcYbI5q/QL1hcMclPBByoaP4EmFPxX3dpbap0uf2qbX80G+jjVtsHd9rhEmyoc6
tJjj27Z2B0ANPAI53AMDXGF5HHHzzficfizN0vK48EO/xk34EEylSmCIInrf72m54f7wh8RojgoO
zIWZkIU+tpCfO2HTcxRUT/rd6Wfb624YE9ov20+T1U0Yc9nyj
zoNDNBjCXh7+tr4FkUoZGyzqm5OlYkfkKvkwkp6rH8RzUhqKewHjWb7zdlYHEH8XkMtcHneYgeQ
pDQ/E8bshTuLILOELUtRODjszaTXyrm4xlhCzJ7mWlZa+viTV4PzHdRQCVByGxOsFfhJGtrOOrx
SUnyUNVDBbKxiTe7tzuTtZ9demhjAE17svhCxh5tIWg5NMJ3FeG4HzOL9UUoah4gwjyvFa/0azdt
1ZWaYc6SPufQIWk9I2HWRb1m4waiA24LkLBXvYMjWTto+DsVWPP3WQXtPaSBNtnD/VEbM+2bbPER
A1+drMTui6Gvl/iHNQ9uq+UrmXPoR9NtfSEAh/M5BdSH75zGifd369R1eFJqBuBWile2R7ryqnPP
BbVf89md0nhYe3RzdD0DKbJen5/r3eicrc3PculW8cGJDqtuEI9kC1xULyXuhWmpEACgTabmNmw3
T+3LnzEuKU97DtLpgqlJoAqXHBBImDsPmzX
JIDOoNc7J8ouh53L8ZM6jZwFXGqQBteV0HsDPwxTBSGE+tuPQHj/cWvovOBeltewUuBskG0EwsDt
kYwmP5yNlbY4vn0ouL+4tlprZr2oNSitv3Lle/usE5ps70ALpQYvzG369DAqf2T+m3Ld5HaKZ/N+
cWtQSt/EJGjjBTTodrzbkAe7Mkz/euMxU0Pj5Gv4kyLysfivPPuvar+ZuRos/jm5N+mHUQUzWd2i
zk4BvBuyWHNe3Jq45H/ELAND0OEZRoXCRbpz0io+a9C6T44BlOECJyXI1gp/m9lsbb5iSv0HpMSv
4xsLpM1AuXLpKmeqdsHO/zEnLUOhR2Dpk6hoqpnPvK+QbVO0c/YdJ+lkeGIz6C2OsrbbsiO+cUou
l+7mJ6WG7VqifJWNX8mzd6RklCntt0W1CgBk93vzOspDJfnvBkHSZ1VmuiWLWpesttUrYCwF77lc
SLDR3Rvqa29hUORrV4BRHl1LAuf9ofAyg9r3vb2TjlrG58FOekzRxomjP/rTL0jteYiBf6YwoMEw
g2chC3lhRoaTpzTpMSbAoByuI3VCsqMN81JwEAUOtXmjRFg/C8Cnc1/Zg8tYfNtIymNbzH9S5Mmo
Qy2oEUaM/RmAU1yhqEE+J1q4wBfJb933T8oPOQBgbNjntcDsJXwQt1xa/QxDs7d1TaGMkzbY6YxZ
Qxt0s6cTwyda2XwqO6Bd9pugluF4c4l218g+42PTzwHWpTcXbqOaQDLVITfM5LWK7JvAEK5fai9X
c2dooFiNiR+QgU3SwRrZ4GZ1d4wWgOJsPexgEkYOKQuwo6S1vl0WNZtuYC7/+ct4qiHzA2tk+5HB
vMoZ1WyxUQNXo/sfhRx5xK7lT2vHqBx0k1gwoZW7l8/IPPbo0frpdKT2iOGLB/YH46GkWztp0Ft8
+7uPbFebu76teSF04ei3utFf9h+UmcxgNmtGR1BuILEs5ERKI2KDLfr90+ltKhDZu6gOBrcOWxik
x+bhojouvj0o+LdZt9zjSaNPZTkym01zPoFv24xyAA3OUAc1WESHKcuPbwO18LIopmURoEROB3dY
N8veuTfekmYPv3tHOaLdZaL66oJklarJBcHWYlr/ob0/gElFmn+20y/kK7oq9vEP/oOSMYgtiyCW
mBEgcnm6rIQvklFxzt9FFMz06+2I5/W0OSRnr371Pb1nukHFXHJC5bDRMbnR7JobKhPAcDibzl2i
Nt/uWNX3K7L3Ddh1hCHFF/Dl+won2HJsfItOvbfxVoL3fs1Rk6+FXvO5QRqcrQVOKN/z2cn2Y8N/
bLIr86AH3+J7r4fGAspyqx985VMKzzl5OHvi+DzGDmmuzgtHpB3/R0NRbWgbW5
ebpeduehhmzGQ4vL3KbrrbH+QkU2DIlcp+DOYysnVNtDxlFJVSDfvHxBAeOYwm9sjzvrslpHMqkl
tSmiqOnuU/shfPtAdYyxoTTDV11R+TJNQo80Mq7cJUd9NeiYi+TjorpN5qtJW9/XQIPQjO
riuWkK3d/mv4dwGWQkS14CJ/5Em4ONdEJnJzwU4FndrLGh76IWczBM+3grhCVWBWf5EohxV8rMEJ
D7m3HeP6koPo4uxTylRkhSFO8GgP0aFR5cEGsjnIhyPVcf7ad1L9t+A3ajzpPW9m+pcdgWqvamCT
47B/Uc6S/nN8VA+7bIdXVCqTsNiyyNoNSEplk8Qi97nz2ZF6/UcxdoD6aVD/HvJA53usmluCKuyl
nFbFX9eIyOF0DGppo3RsP8ka61pSt+jXrn95xkisO1u/Efmt8lb0bhrPET+NEkA==</xenc:Ciph
erValue>
                </xenc:CipherData>
             </xenc:EncryptedData>
       </Security>
   </env:Header>
   <env:Body>
       <csw:GetRecords maxRecords="10" outputFormat="application/xml"
outputSchema="urn:oasis:names:tc:ebxml-regrep:xsd:rim:3.0"
resultType="results" service="CSW" startPosition="1" version="2.0.2"
xmlns:aoi="http://www.esa.int/xml/schemas/mass/aoifeatures"
xmlns:common="http://exslt.org/common"
xmlns:csw="http://www.opengis.net/cat/csw/2.0.2"
xmlns:gml="http://www.opengis.net/gml"
xmlns:ogc="http://www.opengis.net/ogc"
```



```
xmlns:portal="http://www.esa.int/mass" xmlns:rim="urn:oasis:names:tc:ebxml-
regrep:xsd:rim:3.0" xmlns:serviceNs="http://www.opengis.net/cat/wrs/1.0"
xmlns:wrs="http://www.opengis.net/cat/wrs/1.0">
          <csw:Query typeNames="rim:RegistryPackage rim:ExtrinsicObject
rim:ExtrinsicObject rim:ExtrinsicObject_acquisitionPlatform
rim:ExtrinsicObject_dataLayer rim:Association_acquisitionPlatAsso
rim:Association_dataLayerAsso rim:Classification rim:ClassificationNode">
             <csw:ElementSetName
typeNames="rim:RegistryPackage">full</csw:ElementSetName>
             <csw:Constraint version="1.1.0">
                <ogc:Filter>
                   <ogc:And>
                         <ogc:BBOX>

   <ogc:PropertyName>/rim:ExtrinsicObject/rim:Slot[@name=&quot;urn:ogc:def:e
bRIM-Slot:OGC-06-
131:multiExtentOf&quot;]/wrs:ValueList/wrs:AnyValue[1]</ogc:PropertyName>
                                  <gml:Envelope srsName="EPSG:4326"
xmlns="http://www.esa.int/xml/schemas/mass/aoifeatures"
xmlns:sch="http://www.ascc.net/xml/schematron"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
                                        <gml:lowerCorner>23.1368 -
40.7547</gml:lowerCorner>
                                        <gml:upperCorner>58.3726
32.2642</gml:upperCorner>
                                  </gml:Envelope>
                            </ogc:BBOX>
                            <ogc:PropertyIsEqualTo>

   <ogc:PropertyName>/rim:ExtrinsicObject/rim:Slot[@name=&quot;urn:ogc:def:e
bRIM-Slot:OGC-06-
131:parentIdentifier&quot;]/rim:ValueList/rim:Value[1]</ogc:PropertyName>

   <ogc:Literal>urn:ogc:def:EOP:ESA:SIMU.EECF.ENVISAT_MER_FR_xS</ogc:Literal
>
                            </ogc:PropertyIsEqualTo>
                            <ogc:PropertyIsEqualTo>

   <ogc:PropertyName>/rim:ExtrinsicObject/@objectType</ogc:PropertyName>
                                  <ogc:Literal>urn:x-ogc:specification:csw-
ebrim:ObjectType:EO:EOProduct</ogc:Literal>
                            </ogc:PropertyIsEqualTo>
                            <ogc:PropertyIsGreaterThanOrEqualTo>

   <ogc:PropertyName>/rim:ExtrinsicObject/rim:Slot[@name=&quot;urn:ogc:def:e
bRIM-Slot:OGC-06-
131:beginPosition&quot;]/rim:ValueList/rim:Value[1]</ogc:PropertyName>
                                  <ogc:Literal>2009-06-
26T00:00:00.000</ogc:Literal>
                            </ogc:PropertyIsGreaterThanOrEqualTo>
                            <ogc:PropertyIsLessThanOrEqualTo>

   <ogc:PropertyName>/rim:ExtrinsicObject/rim:Slot[@name=&quot;urn:ogc:def:e
bRIM-Slot:OGC-06-
131:endPosition&quot;]/rim:ValueList/rim:Value[1]</ogc:PropertyName>
                                  <ogc:Literal>2009-06-26T23:59:59.000

                                  </ogc:Literal>
                            </ogc:PropertyIsLessThanOrEqualTo>
                            <ogc:PropertyIsEqualTo>

   <ogc:PropertyName>$acquisitionPlatform/@objectType</ogc:PropertyName>
                                  <ogc:Literal>urn:x-ogc:specification:csw-
ebrim:ObjectType:EO:EOAcquisitionPlatform</ogc:Literal>
                            </ogc:PropertyIsEqualTo>
                            <ogc:PropertyIsEqualTo>
```



```
    <ogc:PropertyName>$acquisitionPlatAsso/@sourceObject</ogc:PropertyName>

    <ogc:PropertyName>/rim:ExtrinsicObject/@id</ogc:PropertyName>
                            </ogc:PropertyIsEqualTo>
                            <ogc:PropertyIsEqualTo>

    <ogc:PropertyName>$acquisitionPlatAsso/@associationType</ogc:PropertyName
>
                                    <ogc:Literal>urn:x-ogc:specification:csw-
ebrim:AssociationType:EO:AcquiredBy</ogc:Literal>
                            </ogc:PropertyIsEqualTo>
                            <ogc:PropertyIsEqualTo>

    <ogc:PropertyName>$acquisitionPlatAsso/@targetObject</ogc:PropertyName>

    <ogc:PropertyName>$acquisitionPlatform/@id</ogc:PropertyName>
                            </ogc:PropertyIsEqualTo>
                        </ogc:And>
                    </ogc:Filter>
                </csw:Constraint>
            </csw:Query>
        </csw:GetRecords>
    </env:Body>
</env:Envelope>
```

Figure 11: Example of Service Request

7.2.3 Failed Request

An example is given below:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <soapenv:Fault>
      <faultcode>AuthorisationFailed</faultcode>
      <faultstring>Country of origin not authorised</faultstring>
    </soapenv:Fault>
  </soapenv:Body>
</soapenv:Envelope>
```

7.3 Service Response

7.3.1 Synchronous Service Response

The service response for a synchronous operation is as defined in the service interface, which is detailed for example in the catalogue, ordering or programming specifications.

7.3.2 Asynchronous Service Response

The service response for an asynchronous operation is as defined in the service interface, which is detailed for example in the catalogue, ordering and programming specifications.

The asynchronous protocols based on WS-Addressing and polling clearly boils down to the normal synchronous request/response case. The choice is left whether to set-up or not an authentication / authorisation layer for asynchronous service responses. This shall be decided by agreement of all the parties of the circle of trust that provide or use asynchronous services.

If the choice is made to set-up an authentication / authorisation layer for asynchronous service responses, then the response shall be protected by the same encryption and signature as defined for the service request and authentication.

- For protocols based on polling, the client and SP keep their initial roles and the use cases are exactly the same than those covered previously.
- For protocols based on WS-Addressing, the SP takes the role of the client and conversely. The sequence of steps is as follows:

  1. The SP prepares the response to the endpoint mentioned in the WS-Addressing.
  2. This response is addressed to the PEP of the SP.
  3. The PEP of the SP attaches to the asynchronous response a SAML token authenticating itself as SP. This requires the SP to access an IdP (STS) belonging to the circle of trust, the user registry of the accessed IdP containing an entry that is a surrogate for the SP. For the ease of implementation and integration, it is recommended to have an architecture with one single IdP for the circle of trust. Other architectures with multiple IdP are possible however, including architectures where SP and IdP reside on the same entity.
  4. The asynchronous response is returned to the address provided in the WS-Addressing of the request. This will be the address of a PEP that knows the public key of the IdP providing the SAML token attached to asynchronous response, for the purpose of signature verification. If multiple IdP architecture is chosen, then the PEP shall know the set of public keys associated to all these IdP.

8 Web Portal / Web Service Broker Integration

The present section provides specific information to use the present best practices in the context of an integration of a Web-SSO system with a Web service broker. More specifically, it covers the integration of an authentication environment based on HTTP binding (e.g. Shibboleth [OR1]) with the one based on SOAP and SAML (the present document), as expressed in HL-REQ060.

As an exampe of Web-SSO system, UM-SSO is an operational SSO system based on Shibboleth for ESA Web-based Applications, which could be adopted by other EO Providers as well. It features typical user management functions (login-authentication, registration-account maintenance, access control). It allows ESA Web applications to outsource the sign-on process and offers a given user access to several ESA EO Portals with one single sign-on on his browser.

A given Web-SSO system defines a specific security domain, which is separated from the security domain defined by the service broker. In concrete terms, the two security domains rely on different security tokens; the Web-SSO IdP authenticates Web Portal users without providing the SAML token defined by the present interface.

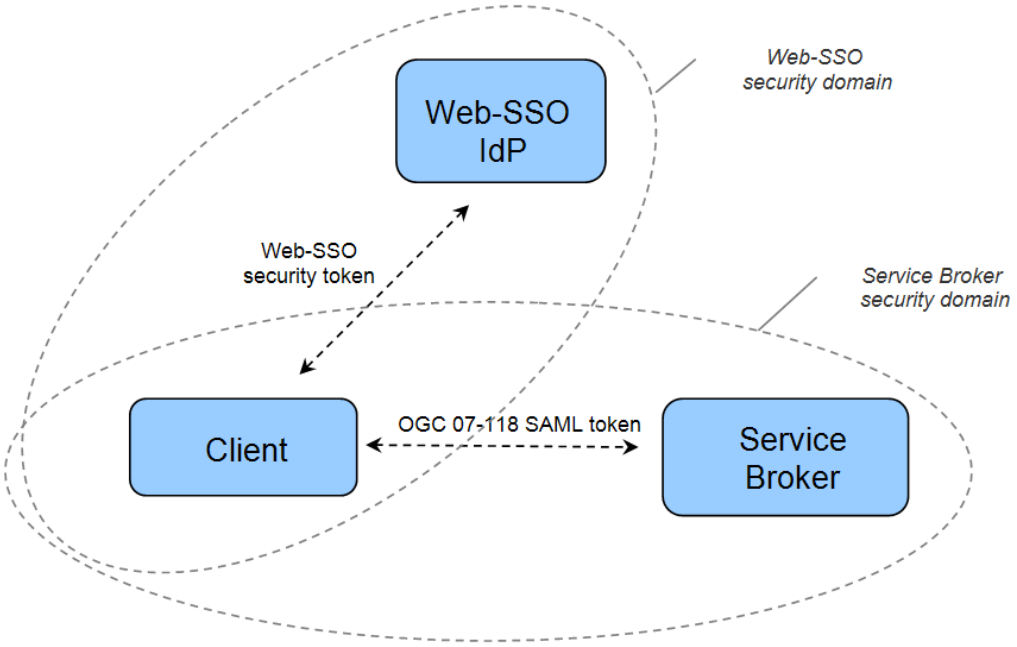

Figure 12 Web-SSO and Service Broker security domains

In this specific context, a secure bridge shall be established between the two security domains, relying on a trust relationship. RST with signature (see 6.4.3.3) shall be used for that purpose.

In order to establish this trust relationship, a given Client *C* of the Web-SSO security domain shall provide a certificate with its public key to the Web service broker's STS. The trust relationship between *C* and service broker IdP is established as soon as the service broker security administrator has put this public key in the keystore of the Web service-broker's STS. From that point, the client *C* can obtain a SAML token for any Web-SSO authenticated users by issuing RST with signature. The sequence of steps is as follows, for a user *U* that has not yet signed on the Web-SSO (this is largely simplified, in order to provide the most significant components and steps):

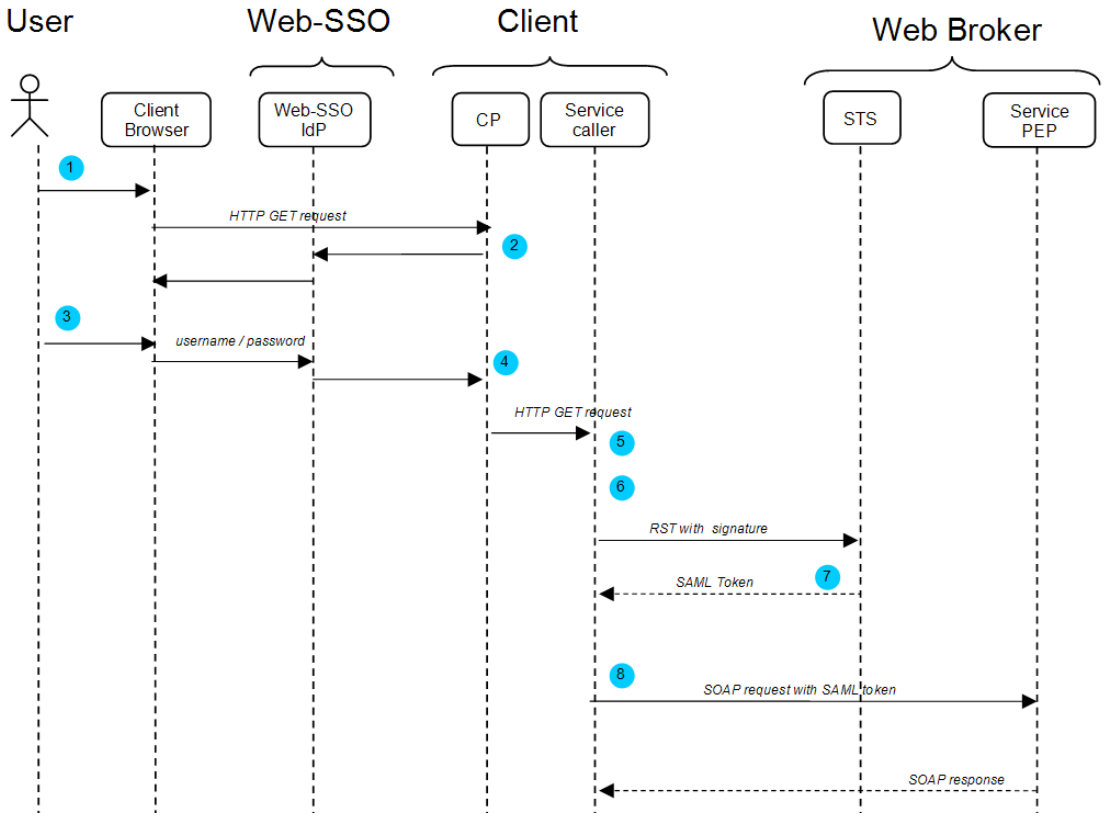

Figure 13 Web-SSO / Service Broker integration sequence diagram

1. The user *U* activates a function on client *C*, that shall use a Web service broker service
2. The Web-SSO checkpoint on *C* relays the authentication to the Web-SSO IdP
3. The user *U* enters his/her credentials and successfully signs on the Web-SSO IdP
4. The Web-SSO checkpoint on *C* sends a GET request containing Web-SSO-ID in HTTP header
5. The client *C* prepares an RST with signature, by putting Web-SSO-ID as username and by signing the request.
6. the client *C* issues the RST to the service broker's STS.
7. The service broker STS verifies the RST signature and returns an encrypted SAML token.
8. The client *C* issues service request(s) to the Web service broker PEP with encrypted SAML token in SOAP header.

If the user has already sign-on on Web-SSO, then the system shall skip the sign-on process (steps 2 and 3) and the sequence resumes at step 4.

9 Security Considerations

The interface that is presented in the current document was designed according to a specific set of security requirements. Other application domains may want to take additional security measures which are complementary to the minimal interface defined in the current document.

The present section identifies different types of attack or threats that are specific to the present interface; it provides for each of these types of attack or threat the answer or countermeasure, as entailed by the interface. When required, the distinction is made between RSTs and service requests.

Type of attack / threat	Answer / countermeasures
Identity Spoofing	If the IdP complies with the present Best Practice (see cases 6.4.3.1 and 6.4.3.2), then the sole artefact that conveys user identity, i.e. an evidence of authentication, is the SAML token, obtained by an RST with password. The IdP guarantees that the SAML token for user *X* is returned if and only if the credentials of *X* have been provided (see next threat topics related to password). If the IdP is an External Entity not complying with the present Best Practice (see case 6.4.3.3), then the threat of identity spoofing has to be analysed at the level of this IdP, as well as the level of security gateway that shall request SAML token to STS. The action of registering the public key of such External entity on the STS means that this STS *trusts* both external IdP and security gateway. If this is done, then the STS shall serve any RST secured by signature from that security gateway, with no further identity control. The signature verification shall guarantee that the RST that it has been issued by a trusted security gateway and that it has not been tampered with. For the service requests, the risk is the theft of SAML token, which could be rebound to a new service request issued by a malicious user. This risk is limited by putting short expiry time on SAML token; as the expiry time is part of the SAML token itself, it is protected from changes by signature. The expiry time and signature are both checked by the PEP. Also, HTTPS could be used to avoid (through encryption) the risk of such forged service request. Another countermeasure consists in putting an IP filter to check whether the client is authorised to make service requests.
Man-in-the-middle	For RST (with password or with signature): the transport protocol is HTTPS, which is based on SSL; SSL includes a certificate mechanism to protect against man-in-the-middle attack. For service requests (if no secured HTTP is used): the signature protocol guarantees that the emitter of SAML token is a trusted IdP and that the token has not been tampered with; this is checked by the PEP. The threat is therefore located on the message payload (SOAP body) or its binding with SAML token. Such threat is analysed in Identity Spoofing, Data integrity, Data confidentiality topics.
Data integrity	The signature protocol enforced on SAML Token allows for the verification of its own data integrity, at the PEP level. The data integrity of the message payload may be checked by another signature mechanism on the SOAP body. Such signature should be bound in some way with the SOAP header, in order to avoid the risk of forged service request (see "identity spoofing" topic).
Data confidentiality	Encryption of SAML token (both for RSTs and service requests) guarantees that no entity excepting the target PEP can read

/ privacy violation	conveyed user attributes. For service request, the data confidentiality of the message payload may be enforced by using HTTPS protocol or by encryption of the SOAP body. The user registry (e.g. LDAP) is protected by password, which is known only by security officer and IdP. The IdP is a "trusted" entity.
Replay attack	For RSTs: the transport protocol is HTTPS, which is based on SSL; SSL includes a "nonce" mechanism to protect against replay attack. For service requests (if simple HTTP is used): the risk of unauthorised access through replay of a past service request is limited by putting short expiry time on SAML token, which is checked by the PEP. Also, a replay protection may be implemented using a hashing function or digital signature which provides a unique identifier that can be used to determine if the same message is received multiple times.
Denial of Service	Web service is susceptible to message flooding denial of service attacks from message replay. “replay detection” mechanisms can be used.
Password Disclosure	If the IdP complies with the present Best Practice (see cases 6.4.3.1 and 6.4.3.2), then RST with password is used, which relies on HTTPS. The password is therefore encrypted during transmission from client to IdP. It is an implementation decision whether deployments use an LDAP registry. If LDAP is used, the LDAP registry is protected by password, which is known only by security officer and IdP. The user passwords are stored encrypted on LDAP registry. Secure LDAP (SLDAP) protocol may be used also. The risk of password disclosure is therefore limited to known and usual factors, which can be mitigated by enforcing an adequate password policy (out of scope of the present interface). If the IdP is an External Entity not complying with the present Best Practice (see case 6.4.3.3), then the RST with signature is used, which contains no password. The threat of password disclosure shall be analysed at the level of the external IdP, which is out of scope of the present Best Practice.
Password Cracking / Guessing	If the IdP complies with the present Best Practice (see cases 6.4.3.1 and 6.4.3.2), then RST with password is used. The risk of password cracking/guessing is limited to known and usual factors, which can be mitigated by enforcing an adequate password policy (out of scope of the present interface). If the IdP is an External Entity not complying with the present Best Practice (see case 6.4.3.3), then the RST with signature is used, which contains no password. The threat of password cracking/guessing shall be analysed at the level of the external IdP, which is out of scope of the present Best Practice.
Unauthorised access	The authorisation to Web services relies on PEP and associated access policy rules. The rules are based on asserted user attributes in the SAML token. The fact that these attributes match the actual requesting user relies on authentication.

The following table covers implementation-dependant threats.

Type of attack / threat	Answer / countermeasures
SQL injection	If a RDBMS is used for user registry, there is a risk of SQL injection for the authentication operation, i.e. a hacker enters as user id or password, some malicious character string that are interpreted by SQL engine. Such risk can be prevented by performing string validation and character escaping on input user id / password strings, before SQL lookup (out of scope of the present interface).
LDAP injection	If a LDAP registry is used for user registry, there is a risk of LDAP injection, i.e. a hacker enters as user id or password, some malicious character string that are interpreted by LDAP or JNDI API. See http://www.blackhat.com/presentations/bh-europe-08/Alonso-Parada/Whitepaper/bh-eu-08-alonso-parada-WP.pdf Such risk can be prevented by performing string validation and character escaping on input user id / password strings, before LDAP lookup (out of scope of the present interface).

10 Authorisation Use Cases (non-normative)

As explained before, authorisation rules that grant access to Web services shall be evaluated by a dedicated PEP that wraps such services. However, the PEP treatments and the way access rules are stored and evaluated are not in the scope of the present document. The present section provides non-normative information about this topic.

In order to separate responsibilities, a PEP typically relies on a PDP (Policy Decision Point) that performs the actual evaluation of access rules based on the request payload (i.e. the SOAP body), on the attributes of the SAML token, if any, present in the SOAP header and on "external" elements (e.g. current time). Each PDP should have a dedicated policy store, where needed access rules or policies can easily be stored, retrieved and maintained.

The rules used by each PDP should be expressed in a standard syntax: the eXtended Access Control Markup Language (XACML) is recommended here. XACML (see [NR21]) is, in essence, a declarative access control policy language implemented in XML. It is worth mentioning here also GeoXACML (see [NR25]), which is an extension of XACML for the declaration and enforcement of geo-specific access restrictions for geographic data.

The following provides use cases and examples of policy rules, with XACML fragments implementing them. More comprehensive examples shall be found in annex E.

10.1 Uses Case: restrict access for time period

Generic policy rule:

Restrict data access for a given time period

Analysis:

XACML allows to define Rules based on “environment attributes”, such as date and time. A rich set of functions for handling date, time and dateTime values (as defined in the W3C XML Schema specification) are predefined in XACML.

Example:

Although able to access the service the user cannot access images from period t1=09:00:00 to t2=12:00:00.

The time restriction can be expressed as a Condition in an XACML rule as follows:

```
<Condition>
  <Apply FunctionId=”urn:oasis:names:tc:xacml:2.0:function:time-in-range”>
    <Apply FunctionId=”urn:oasis:names:tc:xacml:1.0:function:time-one-and-only”>
      <EnvironmentAttributeDesignator
               AttributeId=”urn:oasis:names:tc:xacml:1.0:environment:current-time”
               DataType=”http:www.w3.org/2001/XMLSchema#time”/>
    </Apply>
    <AttributeValue
        DataType=”http:www.w3.org/2001/XMLSchema#time”>09:00:00</AttributeValue>
    <AttributeValue
        DataType=”http:www.w3.org/2001/XMLSchema#time”>12:00:00</AttributeValue>
  </Apply>
</Condition>
```

See annex E for a more comprehensive example.

10.2 Uses Case: enforce rules for specific group of users

Generic policy rule:

enforce rules, like temporal restriction seen before, for specific group of users

Analysis:

XACML allows defining rules which target specific subjects. The rule for the current requirement can be expressed by targeting the group of users whose access shall be regulated together with a time restriction condition.

Needless to say, the group of users shall be targetable through an attribute contained in the SAML authentication token. In this way, a Rule with the following target could be defined:

Example:

Enforce rule for the users having the role "guest".

```
<Target>
  <Subjects>
    <Subject>
      <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
            guest
        </AttributeValue>
        <SubjectAttributeDesignator
             AttributeId="urn:ogc:um:eop:0.0.4:saml:role"
             DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </SubjectMatch>
    </Subject>
  </Subjects>
</Target>
```


where `AttributeId="urn:ogc:um:eop:0.0.4:saml:role"` is a user-defined attribute contained in the XACML decision request which holds the suitable SAML Token attribute value identifying the group of users subjects to the Rule.

Notice that a Rule Target can match more than one Subject.

See annex E for a more comprehensive example.

10.3 Uses Case: restrict access to the type of data

Generic policy rule:

restrict access to the type of data e.g. high or low resolution data

Analysis:

XACML allows to define Rules which target specific attributes of the resource to access. However, we assume that this information is either contained in the client request to the Service, or in a configuration file.

Notice that, building a Rule restricting access for certain data values but these data values are not provided in input, can result in an Indeterminate Policy (Indeterminate means that an error occurred or some required value was missing, so a decision cannot be made).

Example:

See annex E.

10.4 Uses Case: restrict access to data based on the age of the data

Generic policy rule:

restrict access to data based on the age of the data

The age of data is an essential parameter to be considered for some products within EUMETSAT data policy (for instance at the moment Meteosat data are only accessible for retrieval from the archive 24 hours after sensing time).

Analysis:

If the age of data is a piece of information contained in the service request, it is possible to define a rule which set restrictions on the access to the data based on their age.

Example:

For example, the following Condition evaluates to true if the current dateTime is greater than the acquisition end time of the data + 24 hours.

```
<Condition>
<Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function: dateTime-greater-than-or-
equal ">
  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:dateTime-one-and-only">
    <EnvironmentAttributeDesignator
       AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-dateTime"
       DataType="http://www.w3.org/2001/XMLSchema#dateTime"/>
  </Apply>

  <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function: dateTime-add-
dayTimeDuration">
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:dateTime-one-and-only">
      <ResourceAttributeDesignator
AttributeId="urn:ogc:def:ebRIM-Slot:OGC-06-131:endPosition"
```



```
DataType="http://www.w3.org/2001/XMLSchema#dateTime"/>
    </Apply>
    <AttributeValue
DataType="http://www.w3.org/TR/2002/WD-xquery-operators-20020816#dayTimeDuration">
      <xf:dt-dayTimeDuration>
              PT24H
      </xf:dt-dayTimeDuration>
    </AttributeValue>
  </Apply>
</Apply>
</Condition>
```

where `AttributeId="urn:ogc:def:ebRIM-Slot:OGC-06-131:endPosition"` is a user-defined attribute contained in the XACML decision request which holds the corresponding value of the service request.

10.5 Uses Case: imposing geographical constraints

Generic policy rule:

imposing geographical constraints, i.e. area of interest (AOI), allowing some users to access more areas than others.

Analysis:

XACML is a general-purpose access control policy language and does not include specific functions and attributes to handle geographical rules. Given that it is also an extensible language, the user can add his/her own attributes and functions, or, better, in this case, he/she can integrate the XACML rules with GeoXACML [NR25], which specifically addresses geographical constraints.

10.6 Uses Case: access and check source, content, user credentials and time

Generic policy rule:

access and check source, content, user credentials and time

Analysis:

XACML rules targets the following groups of attributes:

- Subject
- Resource
- Action
- Environment

A rich set of attributes are predefined for each group together with functions to handle them, according to their types. Additionally, XACML can be extended with user defined attributes and functions.

10.7 Uses Case: restricting access to users from certain geographic locations.

Generic policy rule:

restricting access to users from certain geographic locations.

Analysis:

An XACML Rule can be defined to restrict access to users from geographical locations provided that this information is contained in the request to the Service Provider.

For example, if the authentication is performed according to the present "User Management Interfaces for Earth Observation" Best Practice, then the request may contain a SAML Token with attributes defined according to the "GMES Minimum Profile"; one of these attribute is the "country of origin" of the subject requesting access. Consequently, this attribute will be embedded in the XAML decision request and a Rule can be defined accordingly.

Example:

See annex E.

10.8 Uses Case: route service access based on user type

Generic policy rule:

Route a service access based on user type.

Note: e.g. This would allow a "scientific" user request to be routed to DLR and a "commercial" user request to be routed to Infoterra.

Analysis:

This requirement could be met using the XACML Obligations; the Obligation is defined as follows:

"Obligation - An operation specified in a policy or policy set that should be performed by the PEP in conjunction with the enforcement of an authorisation decision"

In our case, the operation to be carried out is sending the request to the suitable provider; for each user type value, a policy can be defined with the following features:

- a rule matching a target subject type and having effect "permit";
- an obligation to send the request to the suitable Service Provider if the policy evaluates to "permit";

Annex A: Abstract Test Suite (Normative)

1 Conformance Test Class: The core

1.1 Test Module M.1 Basic requirements

This Test Module is made up of Abstract Test Cases which establishes preliminaries conditions to the actual test cases, such as the protocol bindings, messaging framework, adoption of specification and algorithms to encrypt and sign the messages.

1.1.1 ATC-1.1 SOAP Binding of the request/response messages

Test case identifier	"urn:ogc:cite:ats:um:0.0.4:07-118r1:soap-binding"
Test assertion [purpose]	Operations shall support the embedding of requests and responses in SOAP messages. Only SOAP messaging (via HTTP/POST or HTTPS/POST) with document/literal style shall be used. Messages should conform to SOAP 1.2. The following assertions shall hold: • The SOAP Header holds the authentication token [if applicable], embedded in a WS-Security element. • The SOAP Body holds the message payload.
Test method	Send a request embedded in a SOAP Envelope over the HTTP[S] protocol; verify that a response is returned (even in case of failure) embedded in a SOAP Envelope over the HTTP[S] protocol. The SOAP Envelope shall be compliant with version 1.2 of SOAP (namespace http://schemas.xmlsoap.org/soap/envelope/)
Reference	Clause 6.2
Test type	Capability

1.1.2 ATC-1.2 SAML token encoding for authentication information

Test case identifier	"urn:ogc:cite:ats:um:0.0.4:07-118r1:saml-token"

Test assertion [purpose]	SAML 1.1 is proposed to encode the user authentication token. WS-Security is proposed to encode the SAML assertions in the SOAP header. A SAML token is made of the following statements: • Authentication statements: a typical authentication statement asserts Subject S authenticated at time t using authentication method m. • Attribute statements: a typical attribute statement asserts Subject S is associated with attributes X,Y,Z having values v1,v2,v3. The set of attribute statements returned in a SAML token shall be defined arbitrarily.
Test method	• Send a valid RST to the STS; the response shall contain a SOAP message whose SOAP Body holds an encrypted SAML token. • Decrypt the SAML token using the Relying Party's private key, and verify that the SAML token has the expected statements covering the (arbitrarily) defined set of attributes. **Pre-condition:** For carrying out this test, the client needs a copy of the STS private key. For testing purposes, a couple of private/public keys can be generated using available tools (for example, 'keytool' on JRE), where the certificate with the public key is self-signed by the STS itself.
Reference	Clauses 6.2 and 6.3
Test type	Capability

1.1.3 ATC-1.3 Encryption algorithm for SAML token

Encryption of the SAML token is performed by the STS when creating a RSTR.

Decryption of SAML token is performed by [the PEP of] the Service Provider.

Test case identifier	urn:ogc:cite:ats:um:0.1.0:07-118r5:encryption
Test assertion [purpose]	The encryption algorithm used for the SAML token is the AES-128. The symmetric AES-128 key used for encryption is made available to the recipient as follows: • The key is encrypted using the asymmetric RSA encryption algorithm with the public key of the recipient. • The resulting value is added to the encrypted message, using the XML Encryption [NR17] and XML Signature [NR18] specifications

Test method	1. Send a valid RST to STS; the response shall contain a SOAP message whose SOAP Body holds encrypted data. 2. Decrypt the AES-128 symmetric key contained in the response using the Relying Party's private key. 3. Decrypt the SAML token using the AES-128 symmetric key and check that the result contains a valid SAML Assertion **Pre-condition:** For carrying out this test, the client needs a copy of the STS private key. For testing purposes, a couple of private/public keys can be generated using available tools (for example, 'keytool' on JRE), where the certificate with the public key is self-signed by the STS itself.
Reference	Clauses 6.4.1
Test type	Basic

1.1.4 ATC-1.4 Digest algorithm for signing SAML tokens

Test case identifier	urn:ogc:cite:ats:um:0.1.0:07-118r5:digest
Test assertion [purpose]	The secure hash SHA-1 digital signature message digest algorithm is proposed. The SAML Token is signed before encryption. The XML signature <ds:Signature> element of can be used for signature, according to WS-Security specification.
Test method	1. Send an RST to the STS 2. Check that the response contains an encrypted SAML token and decrypt it following the process specified in ATC 1.3 3. Digest the SAML token using the SHA-1 algorithm 4. Decrypt the signature using the private key of the Orchestrating Service Provider. 5. Compare the digest obtained at step 3 with the value resulting from step 4. The two values shall match. **Pre-condition:** For carrying out this test, the client needs a copy of the STS private key. For testing purposes, a couple of private/public keys can be generated using available tools (for example, 'keytool' on JRE), where the certificate with the public key is self-signed by the STS itself.
Reference	Clause 6.4.2
Test type	Basic

1.2 Test Module M.2 RST

1.2.1 Test Module M.2.1 RST with password

This Test Module is made up of Abstract Test Cases related to the management of RSTs with password and responses.

- The first test case is related to the following scenario: the client issues an RST with password to the STS without indicating the Identity Provider in charge of fulfilling the request; this is the default case, and implies that the recipient Federating Entity shall fulfil the request.

- The second test case is related to the following scenario: the client issues an RST with password to the STS explicitly indicating the STS as the Identity Provider in charge of fulfilling the request.

- The third test case is related to the following scenario: the client issues an RST with password to the STS explicitly indicating an external entity as the Identity Provider in charge of fulfilling the request.

1.2.1.1 ATC-2.1.1 No request designated IdP - STS resolved as IdP

Test case identifier	"urn:ogc:cite:ats:um:0.1.0:07-118r5:authentication-1"
Test assertion [purpose]	The STS is assumed to be the request designated IdP. In this use case the RST contains only the user credentials (username, password).
Test method	The client issues an RST with: • mandatory username/password information; Verify that the client receives a SAML token which is signed and encrypted according to ATC-1.4. The protocol to be used for the message exchange is SOAP/HTTPS. The SAML token shall be returned in the SOAP Body of the response.
Reference	Clause 6.4.3.1
Test type	Capability

1.2.1.2 ATC-2.1.2 STS is request designated Id

Test case identifier	"urn:ogc:cite:ats:um:0.1.0:07-118r5:authentication-2"
Test assertion [purpose]	The STS is the request designated IdP. In this use case the RST contains an identifier for the STS.

Test method	The client issues an RST with: • mandatory username/password information; • an identifier for the STS. Verify that the client receives a SAML token which is signed and encrypted according to ATC-1.4. The protocol to be used for the message exchange is SOAP/HTTPS. The SAML token shall be returned in the SOAP Body of the response.
Reference	Clause 6.4.3.1
Test type	Capability

1.2.1.3 ATC-2.1.3 External Entity is request designated IdP

Test case identifier	"urn:ogc:cite:ats:um:0.1.0:07-118r5:authentication-3"
Test assertion [purpose]	The External Entity is request designated IdP. In this use case the RST contains an identifier for the external entity.
Test method	The client issues an RST with: • mandatory username/password information; • an identifier for the External Entity. Verify that the client receives a SAML token which is signed and encrypted according to ATC-1.4. The protocol to be used for the message exchange is SOAP/HTTPS. The SAML token shall be returned in the SOAP Body of the response.
Reference	Clause 6.4.3.2
Test type	Capability

1.2.1.4 ATC-2.1.4 RST failure

Test case identifier	"urn:ogc:cite:ats:um:0.1.0:07-118r5:authentication-failure"
Test assertion [purpose]	The STS shall return a SOAP fault message if an RST cannot be fulfilled. The SOAP fault shall clearly indicate raison of failure
Test method	The client issues an RST to the STS, with wrong credentials. Verify that a SOAP fault response is returned indicating reason of failure
Reference	Clause 6.4.3.1 and 7.1.4
Test type	Capability

1.2.2 Test Module M.2.2 RST with signature

This Test Module is made up of Abstract Test Cases related to the management of RST with signature and responses.

1.2.2.1 ATC-2.2.1 succesful RST with signature

Test case identifier	"urn:ogc:cite:ats:um:0.1.0:07-118r5:rst-sign-1"
Test assertion [purpose]	In this use case the RST contains only username and a valid signature
Test method	The client issues an RST with: • mandatory username information and signature Verify that the client receives a SAML token which is signed and encrypted according to ATC-1.4. The protocol to be used for the message exchange is SOAP/HTTPS. The SAML token shall be returned in the SOAP Body of the response.
Reference	Clause 6.4.3.3
Test type	Capability

1.2.2.2 ATC-2.2.2 unsuccesful RST with signature

Test case identifier	"urn:ogc:cite:ats:um:0.1.0:07-118r5:rst-sign-2"
Test assertion [purpose]	In this use case the RST contains only username and a invalid signature
Test method	The client issues an RST with: • mandatory username information and signature Verify that the client receives an error message reporting that the signature is invalid The protocol to be used for the message exchange is SOAP/HTTPS. The SAML token shall be returned in the SOAP Body of the response.
Reference	Clause 6.4.3.3 and 7.1.4
Test type	Capability

1.3 Test Module M.3 Authorisation

This Test Module is made up of Abstract Test Cases related to the management of service requests and responses.

Two abstract test cases are defined for service requests, either for synchronous or asynchronous responses. In both test cases, the service request contains a SAML token in the

WS-Security element of the SOAP header. This SAML token is obtained from a previous RST and is used to control access to services.

1.3.1 ATC-3.1 Authorisation with synchronous response

Test case identifier	"urn:ogc:cite:ats:um:0.1.0:07-118r5:synchronous-authorisation"
Test assertion [purpose]	Only an authorised client can access a requested protected service. The service request header contains a SAML Token returned by a previous successful RST.
Test method	Verify that the service to be invoked is protected, i.e. its WSDL specifies WS-Security policies. The client issues a request containing a SAML token previously obtained through authentication. Verify that the client is authorised to access the protected service, that is a successful response shall be returned.
Reference	Clauses 7.3.1.
Test type	Capability

1.3.2 ATC-3.2 Authorisation with asynchronous response

NOTE: This abstract test case is still under finalization

Test case identifier	"urn:ogc:cite:ats:um:0.1.0:07-118r5:asynchronous-authorisation"
Test assertion [purpose]	Only an authorised client can access a requested protected service. The service request header contains a SAML Token returned by a previous successful RST and WS-Addressing information to allow dispatching of the response.
Test method	Verify that the service to be invoked is protected, i.e. its WSDL specifies WS-Security policies. The client issues a request containing a SAML token, previously obtained through authentication. The Service Provider shall return a service response according to the following format: • The SOAP Header contains a SAML Token which authenticates the Service Provider, signed with the private key of the Service Provider and encrypted with the public key of the Federating Entity; • The SOAP Body contains the actual response of the service. **Pre-condition:** The IUT shall support the asynchronous communication for the requested service.

Reference	Clauses 7.3.2
Test type	Capability

1.3.3 ATC-3.3 Service request failure

Test case identifier	"urn:ogc:cite:ats:um:0.1.0:07-118r5:authorisation-failure"
Test assertion [purpose]	The Service provider shall return a SOAP fault message if an service request cannot be fulfilled. The SOAP fault shall clearly indicate raison of failure
Test method	The client issues a request containing a SAML token, previously signed and encrypted, but it is not authorised to access the protected service. Verify that a SOAP fault response is returned, such that: • the <faultstring> element holds an "Authorisation failure" [or equivalent] statement; • the <detail> element holds application specific information about the reason of failure.
Reference	Clause 7.2.3
Test type	Capability

Annex B: Schemas (Normative)

Since the schemas of WS-Trust have many optional elements, we provided here a narrower schema that limits the degree of freedom of the standard schemas, focusing on RST and RSTR. When the underlying child schemas can not be changed, English annotations are used to specify specific constraints. The constrained schema has been obtained by updating reference files from WS-Trust 1.3:
http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3.xsd

The constrained schema is compatible with the standard WS-Trust 1.3 (i.e. any service implementation conforming to constrained files shall also conform to the standard ones).

In the following, we provide, as support to the WS-Trust 1.3 schema, information on structure of RST, RSTR, then the constrained `ws-trust.xsd schema and oasis-sstc-saml-schema-assertion-1.1.xsd`.

For the following two subsections, namespace prefixes are defined in the following table:

Prefix	Namespace
ds	http://www.w3.org/2000/09/xmldsig#
saml	urn:oasis:names:tc:SAML:1.0:assertion
xenc	http://www.w3.org/2001/04/xmlenc#
wsp	http://schemas.xmlsoap.org/ws/2004/09/policy
wsse	http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd
wst	http://docs.oasis-open.org/ws-sx/ws-trust/200512/

RequestSecurityToken (RST)

The schema for RequestSecurityToken is illustrated in the following diagram.

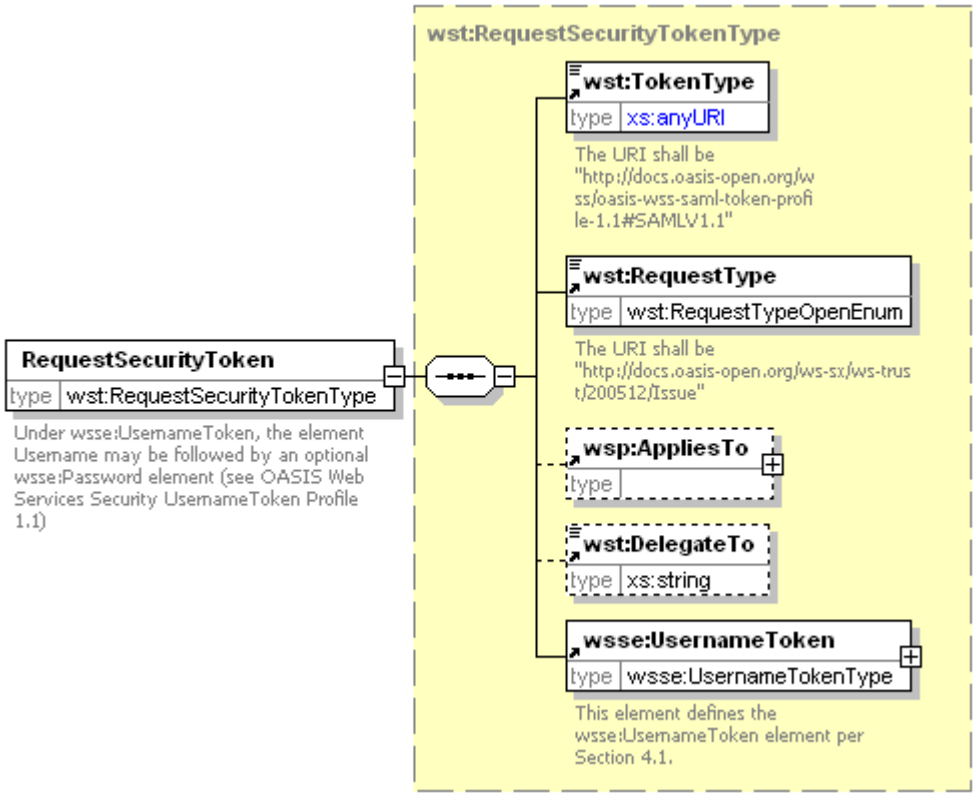

Figure 14 RequestSecurityToken schema

Refer to WS-Trust 1.3 (§4.1 in [NR23]), with the following constraints:

wst:RequestSecurityToken/wst:TokenType

is REQUIRED and shall have the following URI, defined in [NR11] (only SAML 1.1 [NR10] is supported for the moment):
`http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1`
The ability to use this element to support SAML 2.0 is described in the "Extension Points" section (§6.4.5.1).

wst:RequestSecurityToken/wst:RequestType

is REQUIRED and shall have the following URI, (only Issue action is supported by the RST, for the moment):
`http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue`

wst:RequestSecurityToken/wsp:AppliesTo

is OPTIONAL. It shall contain a wsa:EndPointReference, itself containing a wsa:Address. This element is used to inform the STS about which relying party, if not the default one, is supposed to consume the SAML token; the STS can then use the associated public key to encrypt this token.

wst:RequestSecurityToken/wsp:DelegateTo

is OPTIONAL. It is used to require the STS to delegate user identification to an external trusted IdP. It shall contain an identifier known by STS; from this identifier the STS is supposed to retrieve the URL of the external IdP. If the DelegateTo element is absent, then the user identification is performed locally on the STS.

wst:RequestSecurityToken/wsse:UsernameToken
: is REQUIRED. It contains the mandatory element Username, with the user id for which a SAML token is requested. In case of *RST with password*, a wsse:Password element is REQUIRED after Username. In case of *RST with signature*, it is REQUIRED to NOT put wsse:Password element.

Other elements defined in [NR23] are allowed in the RST but they shall be ignored by the STS complying with the present Best Practice.

In case of *RST with signature*, it is REQUIRED to put in the SOAP header a wsse:Security element containing a ds:Signature element. The ds:Signature shall contain the digital signature of the SOAP body (that contains the wst:RequestSecurityToken element), as a detached signature. The following

- The secure hash SHA-1 digital signature message digest algorithm is used, as supported by [NR15].
- The element that is signed is SOAP Body. The URI attribute of the <ds:Reference URI="..."> element shall refer to the <soap:Body> node being signed (using XPointer, see 4.3.3.3 in [NR18].
- The signature is “detached”.
- No certificate is put in the signature. This means that the STS verifying the signature has to know (from its keystore, for example) the public key of the requester, as an evidence of the trust it commits on this requester.
- A canonicalization method shall be used which eliminates namespace declarations that are not visibly used within the SAML token. A suitable algorithm is ”Exclusive XML Canonicalization” which is implemented through a digital signature declaration:

  ```
  <ds:CanonicalizationMethod
  Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
  ```

  Note that the specified canonicalization algorithm omits the comments.

RequestSecurityTokenResponse (RSTR)

The schema for RequestSecurityTokenResponse is illustrated in the following diagram.

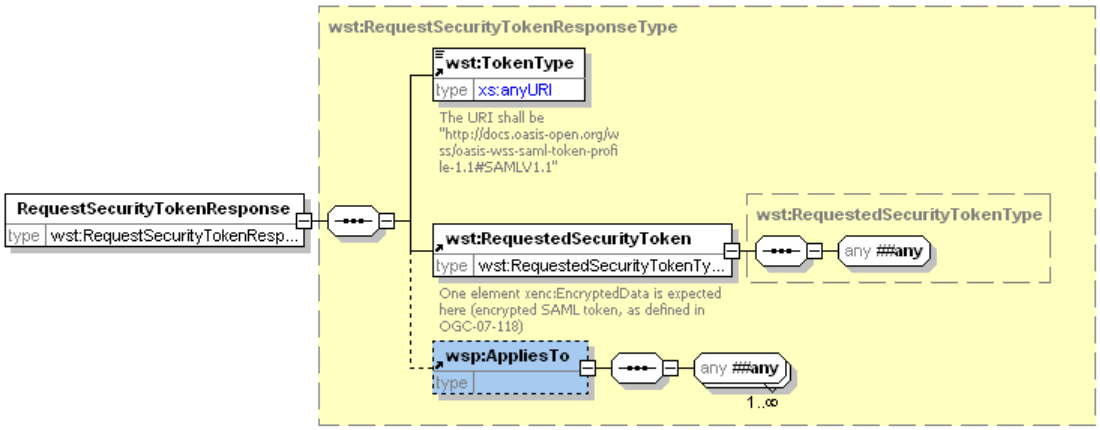

Figure 15 RequestSecurityTokenResponse schema

Refer to WS-Trust 1.3 (§4.1 in [NR23]), with the following constraints:

wst:RequestSecurityToken/wst:TokenType
: is REQUIRED and shall have the following URI, defined in [R11] (only SAML 1.1 is supported for the moment):
`http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1`

wst:RequestSecurityToken/wst:RequestedSecurityToken
: is REQUIRED an shall contains one `<xenc:EncryptedData>` element; once decrypted, it shall be a SAML 1.1 assertion, as defined in oasis-sstc-saml-schema-assertion-1.1.xsd (see below). Specific requirements concerning the encryption and signature of SAML assertion are provided in 6.4.1 and 6.4.2, respectively.

ws-trust.xsd

The following schema file defines the types for RST and RSTR.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema targetNamespace="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/" xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-
trust/200512/" xmlns:wsa="http://www.w3.org/2005/08/addressing"
xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
      <xs:import namespace="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
schemaLocation="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-secext-1.0.xsd"/>
      <xs:import namespace="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
schemaLocation="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd"/>
      <xs:import
namespace="http://schemas.xmlsoap.org/ws/2004/09/policy"
schemaLocation="http://schemas.xmlsoap.org/ws/2004/09/policy/ws-
policy.xsd"/>
      <xs:import namespace="http://www.w3.org/2005/08/addressing"
schemaLocation="http://www.w3.org/2006/03/addressing/ws-addr.xsd"/>
      <!-- WS-Trust Section 3.1 -->
      <xs:element name="RequestSecurityToken"
type="wst:RequestSecurityTokenType">
            <xs:annotation>
                  <xs:documentation>Under wsse:UsernameToken, the
element Username may be followed by an optional wsse:Password element
(see OASIS Web Services Security UsernameToken Profile
1.1)</xs:documentation>
            </xs:annotation>
      </xs:element>
```



```
      <xs:complexType name="RequestSecurityTokenType">
            <xs:sequence>
                  <xs:element ref="wst:TokenType"/>
                  <xs:element ref="wst:RequestType"/>
                  <xs:element ref="wsp:AppliesTo" minOccurs="0"/>
                  <xs:element ref="wsse:UsernameToken"/>
            </xs:sequence>
            <xs:attribute name="Context" type="xs:anyURI"
use="optional"/>
            <xs:anyAttribute namespace="##other"
processContents="lax"/>
      </xs:complexType>
      <xs:element name="TokenType">
            <xs:annotation>
                  <xs:documentation>The URI shall be
"http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-
1.1#SAMLV1.1"</xs:documentation>
            </xs:annotation>
            <xs:simpleType>
                  <xs:restriction base="xs:anyURI">
                        <xs:enumeration value="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1"/>
                        <xs:enumeration value="http://docs.oasis-
open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0"/>
                  </xs:restriction>
            </xs:simpleType>
      </xs:element>
      <xs:element name="RequestType" type="wst:RequestTypeOpenEnum">
            <xs:annotation>
                  <xs:documentation>The URI shall be
"http://docs.oasis-open.org/ws-sx/ws-
trust/200512/Issue"</xs:documentation>
            </xs:annotation>
      </xs:element>
      <xs:simpleType name="RequestTypeOpenEnum">
            <xs:union memberTypes="wst:RequestTypeEnum xs:anyURI"/>
      </xs:simpleType>
      <xs:simpleType name="RequestTypeEnum">
            <xs:restriction base="xs:anyURI">
                  <xs:enumeration value="http://docs.oasis-
open.org/ws-sx/ws-trust/200512/Issue"/>
            </xs:restriction>
      </xs:simpleType>
      <!-- WS-Trust Section 3.2 -->
      <xs:element name="RequestSecurityTokenResponse"
type="wst:RequestSecurityTokenResponseType"/>
      <xs:complexType name="RequestSecurityTokenResponseType">
            <xs:sequence>
                  <xs:element ref="wst:TokenType"/>
                  <xs:element ref="wst:RequestedSecurityToken"/>
                  <xs:element ref="wsp:AppliesTo" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="Context" type="xs:anyURI"
use="optional"/>
            <xs:anyAttribute namespace="##other"
processContents="lax"/>
      </xs:complexType>
      <xs:element name="RequestedSecurityToken"
type="wst:RequestedSecurityTokenType">
            <xs:annotation>
```



```
                  <xs:documentation>One element xenc:EncryptedData is
expected here (encrypted SAML token, as defined in OGC-07-
118)</xs:documentation>
            </xs:annotation>
      </xs:element>
      <xs:complexType name="RequestedSecurityTokenType">
            <xs:sequence>
                  <xs:any namespace="##any" processContents="lax"/>
            </xs:sequence>
      </xs:complexType>
</xs:schema>
```

oasis-sstc-saml-schema-assertion-1.1.xsd

The schema for SAML assertions 1.1 is defined at the following URL:

http://www.oasis-open.org/committees/download.php/3408/oasis-sstc-saml-schema-assertion-1.1.xsd

oasis-200401-wss-wssecurity-secext-1.0.xsd

Each service request may include, if required, the encrypted SAML token returned in the RSTR. In such situation, the SOAP header shall contain a `<wsse:Security>` element (WS-Security 1.1) having a `<xenc:EncryptedData>` (the SAML token) as child.

The schema defining the `<wsse:Security>` element is defined at the following URL:

http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd

Annex C: SOAP 1.1 Implementation (normative)

The normative protocol binding is SOAP 1.2 (see section 6.2). The support to SOAP 1.1 is optional. The present annex is normative specifically in this later case.

If SOAP 1.1 is used, only SOAP messaging (via HTTP/POST) with document/literal style shall be used. The expected SOAP action is:

```
http://docs.oasis-open.org/ws-sx/ws-trust/200512#RequestSecurityToken
```


Annex D: Example of SAML Token Attributes Specification (Non-Normative)

The following subset of attributes necessary to implement the basic EO DAIL policy steps are proposed to be included in the SAML token:

SAML Token attribute name	Description
Id	Unambiguous federated identity
C	Country of origin
O	Organisation
ProjectName	Names of projects with which user is affiliated.
Account	The account number
ServiceName	Associated services
UserProfile	Type of user (Commercial/GMES/Scientific)

Table 1: Attributes in SAML Token

Annex E: XACML Examples (Non-Normative)

Uses Case: restrict access for time period

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

<Request  xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
urn:oasis:names:tc:xacml:2.0:context:schema:os http://docs.oasis-open.org/xacml/access_control-
xacml-2.0-context-schema-os.xsd">

  <Subject>

    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id" DataType="xs:string">

      <AttributeValue>anonymous</AttributeValue>

    </Attribute>

  </Subject>

  <Resource>

    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" DataType="xs:string">

      <AttributeValue>WEB_Map_Server</AttributeValue>

    </Attribute>

  </Resource>

  <Action>

    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" DataType="xs:string">

      <AttributeValue>GetMap</AttributeValue>

    </Attribute>

  </Action>

  <Environment/>

</Request>
```

Policy:

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

<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" xmlns:xacml-
context="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
urn:oasis:names:tc:xacml:2.0:policy:schema:os http://docs.oasis-open.org/xacml/access_control-xacml-
2.0-policy-schema-os.xsd" PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:HL-IDM-480"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">

  <PolicyDefaults>

    <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-19991116</XPathVersion>

  </PolicyDefaults>

  <Target>

    <Resources>
```



```
    <Resource>
      <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
        <AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#string">WEB_Map_Server</AttributeVal
ue>
        <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:resource:resource-id"
          DataType="http://www.w3.org/2001/XMLSchema#string"/>
      </ResourceMatch>
    </Resource>
  </Resources>
 </Target>

 <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:HL-IDM-480" Effect="Deny">
  <Description>
   User cannot access the service for getting maps  in the time range 9:00 AM - 12:00 AM
   </Description>
  <Target>
   <Actions>
    <Action>
     <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">GetMap</AttributeValue>
       <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
        DataType="http://www.w3.org/2001/XMLSchema#string"/>
     </ActionMatch>
    </Action>
   </Actions>
  </Target>
  <Condition>
   <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:time-in-range">
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
     <EnvironmentAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"
      DataType="http:www.w3.org/2001/XMLSchema#time"/>
    </Apply>
    <AttributeValue
DataType="http:www.w3.org/2001/XMLSchema#time">09:00:00</AttributeValue>
    <AttributeValue
DataType="http:www.w3.org/2001/XMLSchema#time">12:00:00</AttributeValue>
```



```
    </Apply>
   </Condition>
  </Rule>

  <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:HL-IDM-480-OTHER"
Effect="Permit"/>


</Policy>
```

Uses Case: enforce rules for specific group of users

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

<Request xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
urn:oasis:names:tc:xacml:2.0:context:schema:os http://docs.oasis-open.org/xacml/access_control-
xacml-2.0-context-schema-os.xsd">
  <Subject>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id" DataType="xs:string">
      <AttributeValue>dail_user_1</AttributeValue>
    </Attribute>
    <Attribute AttributeId="urn:ogc:um:eop:0.0.4:saml:role" DataType="xs:string">
      <AttributeValue>guest</AttributeValue>
    </Attribute>
  </Subject>
  <Resource>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" DataType="xs:string">
      <AttributeValue>csw-ebrim_catalogue</AttributeValue>
    </Attribute>
  </Resource>
  <Action>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" DataType="xs:string">
      <AttributeValue>GetRecords</AttributeValue>
    </Attribute>
  </Action>
  <Environment/>
</Request>
```

Policy:


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

<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" xmlns:xacml-
context="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
urn:oasis:names:tc:xacml:2.0:policy:schema:os http://docs.oasis-open.org/xacml/access_control-xacml-
2.0-policy-schema-os.xsd" PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:HL-IDM-490"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">

 <PolicyDefaults>

  <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-19991116</XPathVersion>

 </PolicyDefaults>

 <Target>

  <Resources>

   <Resource>

    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">

     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">

        csw-ebrim_catalogue

     </AttributeValue>

     <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:resource:resource-id"

        DataType="http://www.w3.org/2001/XMLSchema#string"/>

    </ResourceMatch>

   </Resource>

  </Resources>

 </Target>

 <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:HL-IDM-490" Effect="Deny">

  <Description>

   User with "guest" role cannot access the service in the time range 9:00 AM - 12:00 AM

   </Description>

  <Target>

   <Subjects>

    <Subject>

     <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">

      <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">guest</AttributeValue>

       <SubjectAttributeDesignator AttributeId="urn:ogc:um:eop:0.0.4:saml:role"

        DataType="http://www.w3.org/2001/XMLSchema#string"/>

     </SubjectMatch>

    </Subject>

   </Subjects>

  </Target>
```



```
  <Condition>
   <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:time-in-range">
    <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:time-one-and-only">
     <EnvironmentAttributeDesignator
AttributeId="urn:oasis:names:tc:xacml:1.0:environment:current-time"
      DataType="http:www.w3.org/2001/XMLSchema#time"/>
    </Apply>
    <AttributeValue
DataType="http:www.w3.org/2001/XMLSchema#time">09:00:00</AttributeValue>
    <AttributeValue
DataType="http:www.w3.org/2001/XMLSchema#time">12:00:00</AttributeValue>
   </Apply>
  </Condition>
 </Rule>
 <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:HL-IDM-490-OTHER"
Effect="Permit"/>
</Policy>
```

Uses Case: restrict access to the type of data

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
urn:oasis:names:tc:xacml:2.0:context:schema:os http://docs.oasis-open.org/xacml/access_control-
xacml-2.0-context-schema-os.xsd">
 <Subject>
  <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id" DataType="xs:string">
   <AttributeValue>dail_user_1</AttributeValue>
  </Attribute>
  <Attribute AttributeId="urn:ogc:um:eop:0.0.4:saml:role" DataType="xs:string">
   <AttributeValue>guest</AttributeValue>
  </Attribute>
 </Subject>
 <Resource>
  <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" DataType="xs:string">
   <AttributeValue>csw-ebrim_catalogue</AttributeValue>
  </Attribute>
 </Resource>
 <Action>
```



```
  <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" DataType="xs:string">
   <AttributeValue>GetRecords</AttributeValue>
  </Attribute>
 </Action>
 <Environment/>
</Request>
```

Policy:

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" xmlns:xacml-
context="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
urn:oasis:names:tc:xacml:2.0:policy:schema:os http://docs.oasis-open.org/xacml/access_control-xacml-
2.0-policy-schema-os.xsd" PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:HL-IDM-500"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
 <PolicyDefaults>
  <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-19991116</XPathVersion>
 </PolicyDefaults>
 <Target>
  <Resources>
   <Resource>
    <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
        csw-ebrim_catalogue
      </AttributeValue>
     <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:resource:resource-id"
       DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </ResourceMatch>
   </Resource>
  </Resources>
 </Target>
 <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:HL-IDM-500" Effect="Deny">
  <Description>
   User with the "guest" role cannot access high-resolution data
  </Description>
  <Target>
   <Subjects>
    <Subject>
```



```
        <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">guest</AttributeValue>
          <SubjectAttributeDesignator AttributeId="urn:ogc:um:eop:0.0.4:saml:role"
            DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </SubjectMatch>
      </Subject>
    </Subjects>
  </Target>
  <Condition>
    <Apply FunctionId="urn:oasis:names:tc:xacml:2.0:function:double-greater-than">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:double-one-and-only">
        <ResourceAttributeDesignator DataType="http://www.w3.org/2001/XMLSchema#double"
          AttributeId="urn:ogc:def:ebRIM-Slot:OGC-06-131:sensorResolution"/>
      </Apply>
      <AttributeValue DataType="http:www.w3.org/2001/XMLSchema#double">
        resolution_threshold
      </AttributeValue>
    </Apply>
  </Condition>
 </Rule>
 <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:HL-IDM-500-OTHER"
Effect="Permit"/>
</Policy>
```

Uses Case: restricting access to users from certain geographic locations

```
<?xml version="1.0" encoding="UTF-8"?>
<Request xmlns="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
urn:oasis:names:tc:xacml:2.0:context:schema:os http://docs.oasis-open.org/xacml/access_control-
xacml-2.0-context-schema-os.xsd">
 <Subject>
  <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id" DataType="xs:string">
   <AttributeValue>dail_user_1</AttributeValue>
  </Attribute>
  <Attribute AttributeId="urn:ogc:um:eop:0.0.4:saml:country" DataType="xs:string">
```



```
    <AttributeValue>France</AttributeValue>
   </Attribute>
  </Subject>
  <Resource>
   <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" DataType="xs:string">
    <AttributeValue>csw-ebrim_catalogue</AttributeValue>
   </Attribute>
  </Resource>
  <Action>
   <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id" DataType="xs:string">
    <AttributeValue>GetRecords</AttributeValue>
   </Attribute>
  </Action>
  <Environment/>
</Request>
```

Policy:

```
<?xml version="1.0" encoding="UTF-8"?>
<Policy xmlns="urn:oasis:names:tc:xacml:2.0:policy:schema:os" xmlns:xacml-
context="urn:oasis:names:tc:xacml:2.0:context:schema:os"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="
urn:oasis:names:tc:xacml:2.0:policy:schema:os http://docs.oasis-open.org/xacml/access_control-xacml-
2.0-policy-schema-os.xsd" PolicyId="urn:oasis:names:tc:xacml:2.0:example:policyid:HL-IDM-550"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:deny-overrides">
  <PolicyDefaults>
   <XPathVersion>http://www.w3.org/TR/1999/Rec-xpath-19991116</XPathVersion>
  </PolicyDefaults>
  <Target>
   <Resources>
    <Resource>
     <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
         csw-ebrim_catalogue
      </AttributeValue>
      <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:2.0:resource:resource-id"
         DataType="http://www.w3.org/2001/XMLSchema#string"/>
     </ResourceMatch>
    </Resource>
```



```
    </Resources>
  </Target>
  <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:HL-IDM-550" Effect="Deny">
    <Description>
     User from the "France" country cannot access the service
     </Description>
    <Target>
      <Subjects>
        <Subject>
          <SubjectMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">France</AttributeValue>
            <SubjectAttributeDesignator AttributeId="urn:ogc:um:eop:0.0.4:saml:country"
              DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </SubjectMatch>
        </Subject>
      </Subjects>
    </Target>
  </Rule>
  <Rule RuleId="urn:oasis:names:tc:xacml:2.0:example:ruleid:HL-IDM-550-OTHER"
Effect="Permit"/>
</Policy>
```


Annex F: Example of WSDL using WS-Policy (Non-Normative)

-To be completed-

Annex G: Revision history

Date	Version	Editor	Sections modified	Description
15 September 2007	0.0.1 Draft	R.Smillie	All	Initialised Draft Document.
23 April 2008	0.0.2	R.Smillie		Updated in line with EO DAIL implementation project
07 Feb 2009	0.0.3	R.Smillie		Updated in line with EO DAIL implementation project SOAP version changed to 1.1 Authentication request does not use WS-Security Message examples added Encryption and signature descriptions improved
30 June 2009	0.0.4	R.Smillie		Updated in line with EO DAIL RID PRE-AR2#34: • Namespace in encrypted message example corrected to http://earth.esa.int/um/eop/saml <http://earth.esa.int/um/eop/saml> • decryptandCheckSignature removed from STS • authenticating identity correctly asserted in examples • authenticate and authenticateFederated merged into one operation • Attribute assertions updated in examples • WSDL provided for STS • Clarification made for the assertion element and schema attached • All schemas and references given in annex.

30 October 2009	0.0.5	P. Denis	All	Updates following RIDS of EUMETSAT/con terra, analysis by FP-7 GENESIS and HMA-T projects • Removed references to DAIL and GS, for the sake of generality • Added conformance (chapter 2) and Abstract Test Suite (annex A) • Put SOAP 1.2 as baseline protocol binding and put SOAP 1.1 support in annex C • Demote SAML token attributes specification as an example (annex D) • Added authorisation use cases (chapter 10) and XACML examples (annex E) • Added WS-Policy recommendation • Removed non-standard Assertion element • Remove certificate from SAML token • Changed protocol for authentication through external IdP • Added "future work" section • Clarify roles of each entity in the system • Added threats / countermeasures analysis (chapter 9) • Misc corrections and clarifications

29 January 2010	0.0.6	P. Denis	All	• Alignment on OASIS WS-Trust 1.3, i.e. Security Token Service (STS), providing Request Security Token operation (RST) • Precisions and changes on signature of security tokens • Created annex G for ESA UM-SSO / EO-DAIL Integration • Misc corrections and clarifications
5 March 2010	0.1.0	P.Denis	All	Updates following RIDS of ESA, EUMETSAT/con terra, misc corrections and precisions • Removal of references to LDAP, for the sake of generality • Improved description of encryption protocol ("hybrid cryptosystem"). • Ability to have multiple relying parties (hence multiple Federating Entities) through "AppliesTo" element described in "Extension Points" section. • Removal of sections about too general topics on SAML, encryption and signature. • Added test module for RST with signature • Correction on "Federated IdP - external identification" use case and associated RST schema: it shall use "DelegateTo" element of WS-Trust, instead of "AppliesTo" element. • Reference to GeoXACML • Misc corrections and clarifications • Corrections of typos and wrong section numbering

5 July 2010	0.3.0	P.Denis		"Friendly amendments" following OGC TC 9th March 2010: • Described the re-use mechanism for secured tokens • Detail Single-Sign-On • Removed PEP in front of IdP • Made explicit that an external IdP can be based on Shibboleth • Defined the scope of the document "as specific as possible" Updates following RIDS of ESA, EUMETSAT/con terra, misc corrections and precisions • Generalisation to architectures without a Federating Entity; clarification of roles of STS and Relying Parties; unification of use cases 1 and 3. • Description of the general mechanism used by STS to get the encryption public key. • Clarification on the meaning of the "Client" actor in use cases • Replaced "authorisation request" by "service request" for uniformity • Annex G promoted as section 8 and updated to be more general (Web Portal / Web Service Broker Integration) • Updated document type as "Candidate Best Practice" (Carl Reed) • Precisions made on the scope of the document (P.G. Marchetti)
8 July 2010	0.3.1	P.Denis		Update following HMA AWG Meeting: • Put "Shibboleth" as an example of Web-SSO (not more)

3 September 2010	0.3.2	P.Denis		Updates following issues found in activities of integration based on the present best practice: • Precisions made on XML canonicalization method for signature of SAML tokens and corrections of XML examples • Precisions made on signature reference in SAML tokens and corrections of XML examples • In figure 6, step 10 corrected to be in line with the text that follows • New options allowed for asynchronous service responses

