

Open Geospatial Consortium Inc.

Date: 2010-03-05

Reference number of this document: OGC 07-118r5

Version: 0.1.0

Category: OpenGIS® Engineering Report

Editors: P. Denis, SPACEBEL s.a.

User Management Interfaces for Earth Observation Services

To obtain additional rights of use, visit <http://www.opengeospatial.org/legal/>.

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® Engineering Report
Document subtype: Candidate Implementation Specification
Document stage: Draft
Document language: English

Contents

1	SCOPE	10
2	CONFORMANCE	10
2.1	CONFORMANCE TO BASE SPECIFICATIONS.....	10
2.2	CONFORMANCE CLASSES	10
3	REFERENCES	11
3.1	NORMATIVE REFERENCES	11
3.2	OTHER REFERENCES	12
4	TERMS AND DEFINITIONS	13
5	SYMBOLS AND ABBREVIATIONS	15
5.1	SYMBOLS (AND ABBREVIATED TERMS)	15
5.2	DOCUMENT TERMS AND DEFINITIONS	16
6	SYSTEM CONTEXT	16
6.1	APPLICATION DOMAIN	16
6.2	PROTOCOL BINDING	17
6.3	BASIC USE CASES	18
6.4	SECURITY MODEL.....	19
6.4.1	<i>Encryption</i>	20
6.4.2	<i>Signature / Message Digest</i>	23
6.4.3	<i>Authentication Use Cases</i>	25
6.4.3.1	Federated IdP - local identification (Default Case).....	26
6.4.3.2	Federated IdP - external identification	27
6.4.3.3	External IdP - local identification	30
6.4.3.4	External IdP - external security domain	31
6.4.4	<i>Authorisation Request</i>	33
6.4.5	<i>Extension Points</i>	33
6.4.5.1	Multiple Relying Parties.....	33
	SAML 2.0	33
6.4.5.2	33
7	INTERFACE	34
7.1	REQUEST SECURITY TOKEN.....	34
7.1.1	<i>Request</i>	34
7.1.2	<i>XML encoding</i>	34
7.1.3	<i>Response</i>	35
7.1.3.1	Example SAML Token Before Encryption	37
7.1.4	<i>Failed Request Security Token</i>	38
7.1.5	<i>WSDL</i>	38
7.2	SERVICEREQUEST	39
7.2.1	<i>Request</i>	39
7.2.2	<i>XML encoding</i>	39
7.2.3	<i>Failed Request</i>	42
7.3	SERVICERESPONSE	43
7.3.1	<i>Synchronous</i>	43
7.3.2	<i>Use Case: User logs in at client and makes Synchronous Service Request to Federating Entity Service</i>	43
7.3.3	<i>Asynchronous</i>	44
8	SECURITY CONSIDERATIONS	45
9	AUTHORISATION USE CASES (NON-NORMATIVE)	47
9.1	USES CASE: RESTRICT ACCESS FOR TIME PERIOD	48
9.2	USES CASE: ENFORCE RULES FOR SPECIFIC GROUP OF USERS	48
9.3	USES CASE: RESTRICT ACCESS TO THE TYPE OF DATA	49
9.4	USES CASE: RESTRICT ACCESS TO DATA BASED ON THE AGE OF THE DATA	49
9.5	USES CASE: IMPOSING GEOGRAPHICAL CONSTRAINTS	50
9.6	USES CASE: ACCESS AND CHECK SOURCE, CONTENT, USER CREDENTIALS AND TIME	50
9.7	USES CASE: RESTRICTING ACCESS TO USERS FROM CERTAIN GEOGRAPHIC LOCATIONS.	51

9.8	USES CASE: ROUTE SERVICE ACCESS BASED ON USER TYPE	51
ANNEX A: ABSTRACT TEST SUITE (NORMATIVE)		52
1	CONFORMANCE TEST CLASS: THE CORE	52
1.1	TEST MODULE M.1 BASIC REQUIREMENTS	52
1.1.1	ATC-1.1 SOAP Binding of the request/response messages	52
1.1.2	ATC-1.2 SAML token encoding for authentication information	52
1.1.3	ATC-1.3 Encryption algorithm for SAML token	53
1.1.4	ATC-1.4 Digest algorithm for signing SAML tokens	54
1.2	TEST MODULE M.2 RST	55
1.2.1	Test Module M.2.1 RST with password	55
1.2.1.1	ATC-2.1.1 No request designated IdP - Federating entity resolved as IdP	55
1.2.1.2	ATC-2.1.2 Federating Entity is request designated Id	56
1.2.1.3	ATC-2.1.3 External Entity is request designated IdP	56
1.2.1.4	ATC-2.1.4 RST failure	57
1.2.2	Test Module M.2.2 RST with signature	57
1.2.2.1	ATC-2.2.1 succesful RST with signature	57
1.2.2.2	ATC-2.2.2 unsuccessul RST with signature	58
1.3	TEST MODULE M.3 AUTHORISATION	58
1.3.1	ATC-3.1 Authorisation with synchronous response	58
1.3.2	ATC-3.2 Authorisation with asynchronous response	59
1.3.3	ATC-3.3 Authorisation request failure	59
ANNEX B: SCHEMAS (NORMATIVE)		61
ANNEX C: SOAP 1.1 IMPLEMENTATION (NORMATIVE)		67
ANNEX D: EXAMPLE OF SAML TOKEN ATTRIBUTES SPECIFICATION (NON-NORMATIVE)		68
ANNEX E: XACML EXAMPLES (NON-NORMATIVE)		69
ANNEX F: EXAMPLE OF WSDL USING WS-POLICY (NON-NORMATIVE)		78
ANNEX G: ESA UM-SSO / EO-DAIL INTEGRATION		79

Figures

Figure 1	Two cases of authentication	17
Figure 2	Sequence of authentication/authorisation activities	17
Figure 3	User Management Use Cases	18
Figure 4	Federating (Local) Entity is request designated IdP (Default Case)	27
Figure 5	External Entity is request designated IdP	28
Figure 6	External Entity is IdP	30
Figure 7	External Entity is IdP on external Security Domain	32
Figure 8:	Example RST with password	34
Figure 9:	Example RST with signature	35
Figure 10:	Example Authenticate Response	37
Figure 11:	Security Token Service WSDL	39
Figure 12:	Service Request Example	42
Figure 13	Federating Entity Sequence Diagram Showing Client Request with synchronous Response	44
Figure 14	Sequence Diagram showing asynchronous request	44

i. Preface

This document explains how user and identity management information is included in the protocol specifications for EO (Earth Observation) services, for example catalogue access (OGC 06-131), ordering (OGC 06-141) and programming (OGC 07-018).

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 a new specification; however, it describes how existing specifications from W3C and OASIS can be used in combination to pass identity information to Web services some of which are based on OGC Best Practice specifications.

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 from 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. 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		<p>Updated in line with EO DAIL implementation project</p> <p>SOAP version changed to 1.1</p> <p>Authentication request does not use WS-Security</p> <p>Message examples added</p> <p>Encryption and signature descriptions improved</p>
30 June 2009	0.0.4	R.Smillie		<p>Updated in line with EO DAIL RID PRE-AR2#34:</p> <ul style="list-style-type: none"> • 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	<p>Updates following RIDS of EUMETSAT/con terra, analysis by FP-7 GENESIS and HMA-T projects</p> <ul style="list-style-type: none"> • 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	<ul style="list-style-type: none"> • 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	<p>Updates following RIDS of ESA, EUMETSAT/con terra, misc corrections and precisions</p> <ul style="list-style-type: none"> • 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

v. Future work

vi. Foreword

This document, through its 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 GIS Consortium, Inc. shall not be held responsible for identifying any or all such patent rights.

Introduction

This specification is complementary to a set of specifications that describe services for managing Earth Observation (EO) data products. These services include collection level, and product level catalogues, online-ordering for existing and future products, on-line access to these products, etc. The application of the current specification is not limited to the Earth Observation domain however.

The intent of this specification is to describe an identity management interface that can be 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 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 Earth Observation services.

2 Conformance

2.1 *Conformance to base specifications*

This present section covers the compliance testing of an implementation candidate with the present document.

It is worth highlighting that the present OGC document puts together specifications (SAML, WS Security, XACML) that come from other organizational bodies (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 of the document 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-SAMLSecurityTokenProfile.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/xmlsig-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

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]

To confirm a system entity's asserted principal identity with a specified, or understood, level of confidence.

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.

client

software component that can invoke an **operation** from a **server**

4.4.

external entity

This is the entity owning a protected Web service federated by the Federating Entity. The external entity can be both an identity provider and Service Provider. There can be many external entities.

4.5.

federated identity [NR14]

A principal's identity is said to be federated between a set of Providers when there is an agreement between the providers on a set of identifiers and/or attributes to use to refer to the Principal.

4.6.

Federating Entity

This is the entity performing the federation of the identities. The RST always passes through the Federating Entity. The Federating Entity can be both identity provider and Service Provider. The present specification assumes that there is only one Federating Entity; the possibility of multiple Federating Entities is covered as an "extension point".

4.7.

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

4.8.

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.

**4.9.
interface**

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

**4.10.
operation**

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

**4.11.
parameter**

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

**4.12.
PEP**

Policy Enforcement Point.

**4.13.
principal [NR14]**

A system entity whose identity can be authenticated.

**4.14.
request**

invocation of an **operation** by a **client**

**4.15.
response**

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

**4.16.
server
service instance**

a particular instance of a **service** [ISO 19119]

**4.17.
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]

**4.18.
service interface**

shared boundary between an automated system or human being and another automated system or human being [ISO 19101]

**4.19.
Service Provider [NR14]**

A role donned by a system entity where the system entity provides services to principals or other system entities.

4.20.

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 (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 authorized, 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).

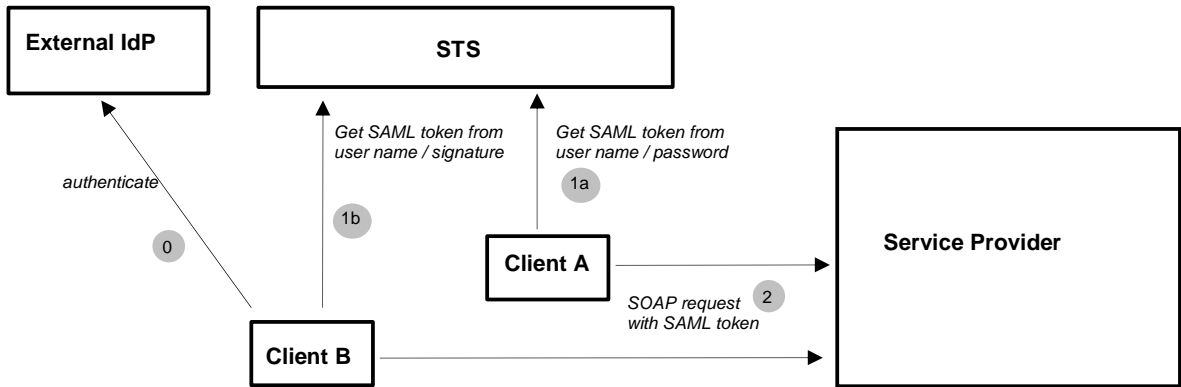


Figure 1 Two cases of authentication

These two cases shall be refined and detailed in section 6.4.3.

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

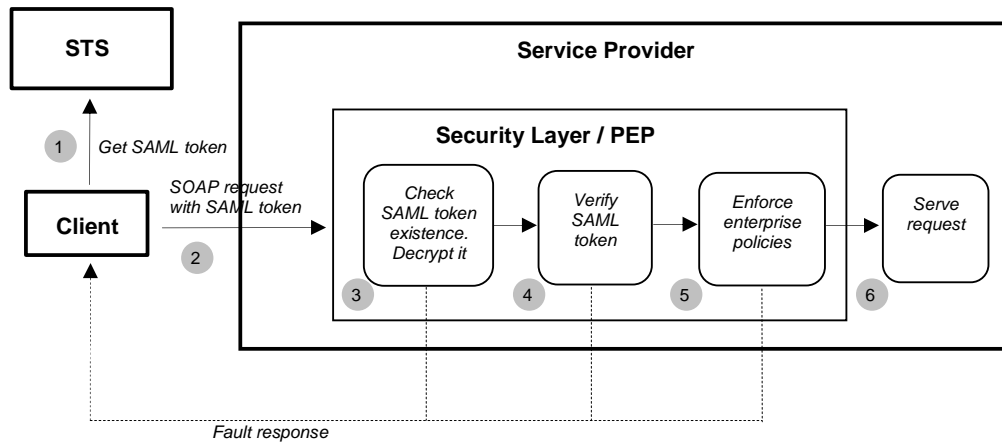


Figure 2 Sequence of authentication/authorisation activities

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 put a detached signature (see below). For authorisation 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 specification 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 current specification.

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

This specification covers identity federation whereby the receiving IdP (Federating Entity), if not the IdP for the request, resolves the IdP and passes the RST to the correct IdP.

Authorisation requests (service requests) may address more than one entity, to perform so-called multi-mission requests, these requests are orchestrated by a BPEL workflow.

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

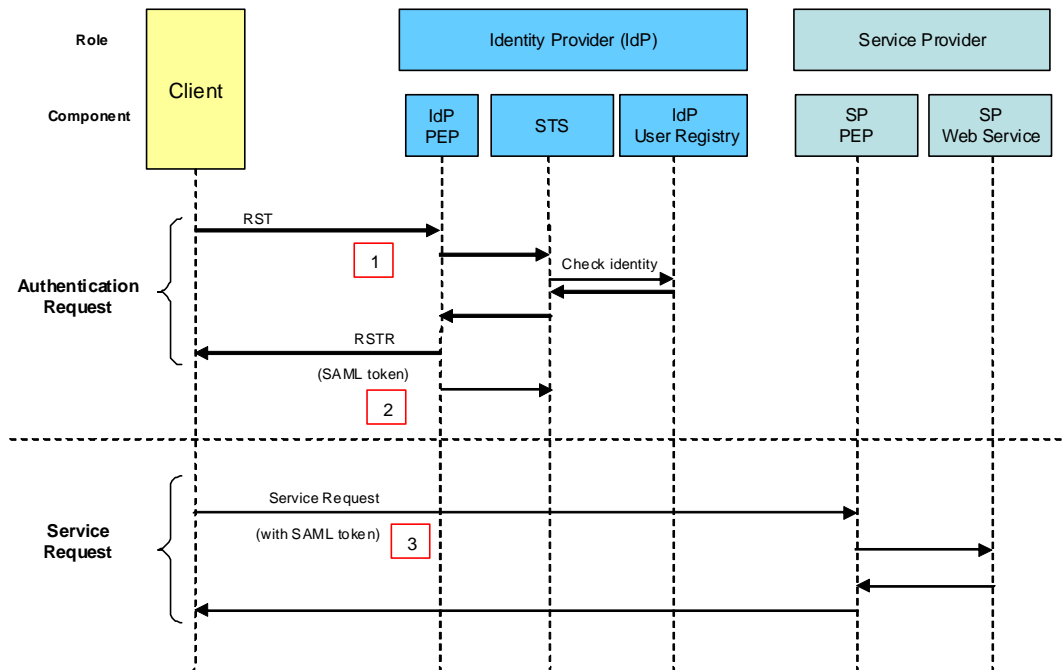


Figure 3 User Management Use Cases

The 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 be either on the Federating Entity or on any external entity. 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 in the Federating Entity is directly exposed as a Web service and does not pass through the PEP. However, if required a request could equally be intercepted by the IdP PEP and routed.
2. The client receives the RSTR containing the SAML token
3. The client then sends a service request i.e. an authorisation request. This request contains the SAML token.

6.4 Security Model

The model is based on OASIS SAML 1.1 [NR10]¹, 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 specification 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 specification. Case 2 is a new pattern, allowing 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 annex G). 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 cases 1 and *RST with signature* for case 2.

¹ 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*² 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 authorisation 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, whose 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 token³ 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/xmenc#"
Type="http://www.w3.org/2001/04/xmenc#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 federated 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 Authentication Use Cases (section 6.5.3).

² The concept of "assertion" here is equivalent to the concept of "claim" in WS-Trust ([NR23]).

³ In the present specification, this is the public key of Federating Entity in all use cases where Federating Entity hosts both STS and federating Service Provider. Generalization to multiple relying parties is covered in "Extension Points" section (see § 6.4.5).

Example Request Security Token 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>
```

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
dU14k+TffynGRJ9Tr9PD8YIdpFLzCvYas63g5x4/XnyAlE2AU8ZBBpM2dtbr3g4IYMywfraWrI76
mHM+MERVZdHMBWFrhQXhcs92m23m+amt14mk=</xenc:CipherValue>
          </xenc:CipherData>
        </xenc:EncryptedKey>
      </ds:KeyInfo>
    </xenc:EncryptedData>
  </wst:RequestedSecurityToken>
  <xenc:CipherValue>VEHlprDMQ+DqIpoPqx6TYi/mMX2dGV5JCJCrhDquZHRKqOiaIFfwqOMZvn
2HW2JDFvUxJ6LRTKdNujQI7sxc6h3IGBL7NXF7bx4jGwQ09wAA7nm6OoB4jgiGdaqb8wTx0oInzn2
WqOWoVeTngllwBi0rv2+iDlHwnXAUUHfJH8ALq4IU3hr0vjoqJH6Y21EuXPeXp/dYPuW3oIFn2FE
ID2u+8T+xOxbbbq2ezQbU3z8n1LbgvDtN3ex5lUc0260pOOPn92nm7nYERt682eYd+bCKoiENpQSY
gHszvvyqFf9o600u87zk4AORWsRhQH74L2gG8wVOeHKyhEx0RsBkf4xZcQKBvQ9JHWQWpDEB51NZ
aJelhSyaUk6T5gf9ArDnz6UwL0ZTDp6DxgJha9l9u5qIMG3ECxVYKcnBv+O6Om1QOHbL0echUDR56
evS+mf0U9JxduBKwFJLqta6D0mwmgYWcaF3ZrKd7SatV8Z210DmWTMe5R+x601RpbKl1tlduK14bL
aSYFpaqaU758ZsmTdmjQQj8fn1qCZbDtp4SEVPWumoTg2k7RAOay2QtV5b+VA9wloSXoxVf2csLS
OOH/NDElnoBtpzGUb9Xm/YIPwikQKsXPFM72yLrS0vjAho1CxrG+8l7XIVcmowhPnLqSs6ZpvA0
1YP8EhsOFln+0y+9EfAuoY4jYcScfwqDehth761ER+EyAdFLi10VhVxKW14VLbmksAydndIQaw6V
zGmlQwoc3CeCaeq4q4GgFgiem1BmW9IeBaUBTX2wZmIKG8Z9XhJv6MwT7hOeWH5fefipJs8J5S816
wQBo8WAczzmw6s1j8JW9YDyAWosfoTPrtoWFTaaYSiaEXvPOnb5RgR/W4ivZ64ioA8FXyLFoWcNE
J76AgWHDLABCDg/zvnVwEs7OdaSxRTxVnsc7cplGspSmk/HzGYxPHInGhn/QPsac5iN6t6HlwnQ
UJgt8lrI/tbFESYqqtYqXKeNoEtw9l/1DZVUi7mSc7Xj2e2Wb65h8PIoYeX3Nli+i4SrOoeAKAZr
HtpqP6f+pi4lPkANS4RFxFDiL9Ddxv1WKD//nMck0Su0HfIbPYUYF0GGv1Hsv6IiwT8dj/f0MnCx
kAgegliGageZthQiNavOcURRC/94d+1jDZGayowurzdxmJhxyiEY5REQQt3hK4aAD89wMjndzxHd
tcQEuvXA5uSm3T9qgIm4Qdvuh54PW/SKptG9fdj4paTxvV1fz+0f/1Vxjj4pPIKOVjE3e4ChBpkJ
XD/nXqZ8Ddr+zPzOLWYyiqnMaxv30Ind/Iz2Lq36a09b0JEFMVZ4e39sGtFzNDbXxgQnTx4L3jDY
Fdl5+ge1UNduK9Htgk1XDwfNIMWtY5xhdTX0m3Q8hBtNgKOHeg7BcBxf+uT30mqwJJu5cbJQ1/1j
/QlMvromUaUQATN2ULu7mMiTWkoYoMTiijJGAizbKIi6O5xmHvF/jicd7lBcmSz+B+BnrnqxY5DM
/qQSFsnRoGmPKlJjeiao2g+QuMD5x7H+pBUIq3B8lkMlUBg5VoKx2+kChuP21amGFskQ180PRGygQ
5adA2iWwKoIKoCdcYic9C2sPVkJz+slExJXizl4L51GEWdlQ8VGsqNV7CzOyIt0uXIIBQW3j0aX/
/7QoYVfM681TiqvtaDEY7Ip4nSV839e5xnj3s+qgzXOpoK5rw5EThDLthPy97CJiUsbsfcGf1Dv
WNE74x2E4b8HazacItrBIbx0GFDHIoqaHEih6zlhQaqlwLnUHRpL8vAQLVKiW3q94569e3GenoQ
bpjxKQ9F58VuQh3ZiZtJ+17XOxDx6ZDXcTiQDa+3nXiTgT7k9gGtpIv8vYLMuUHDEZxlzGd/rMzU
3JAbbfK08+Cs3pMnb2KpGL4rLLgivee1P35rbHK04V8D0NbwDk0TOVnmFQWIRsgVtPwmEHXbF13j
qIUTx4xdishXmkKcDCwNfqYo4yzmgLU1btchkDGF9YBA0mXv4gT7z0TiBbljUFbHTnciL4DBkI9
K8wEjklHui7w4LjvhCB5B15ylZG/baIscorRvu41HUj7crwbsdKjcwGE//dBTXN1vrXDmOmaAkCo
nuYNPpMY0Cf+ikVITJO73UaXplFj00tm+mkql6e5nTd8qQXwHZl/nGJOEO/rMsXSoVybxIn5bg+
97CFctAdsRRjAJZRQcrJIztenGJX81U0rvAX+OuoSNrgVpdxcwuH/1x80i+CY5kdUkg3EMkU0m4F
iNQ6CyXiimVSRB0shfW5/Em+qlYeRjrxCYjYBPo2mCuMtqQN42VeShEkQ1XPx6o09NTaaxXRM
pV2IHjzALLrR0P6xqbp7CuEhPdLxTYcXetDKQJt/XHJuWdvetMgsJnyQ0cCJSPXp21xsrK6zYLY
cQ1M8rS9RHcmWvFdtZg49mX3QPANOUdPoPR0y3m0T19FWKfYOfQhHN2xPJZAPV6ZJeReeAeTBRkT
vgIJE/3BsQpmqsSusjgEDYCrK8MfaybAC6CpE5ZKnQwV99YlTcbPx7vVKPuU13j3Ajj4FjtGjKFun
fCOpLX1AA0FSBbFOOCVeYd94bCGaW8f+j3NBB+29ELYmskew2tyCBiw2HodBrMoDiVWVHbd+bWw8
qMOOBurEQihvDnq5Tbi3R2fnnX9Dpfbv1jJeKfjYVwCLZA50dGIYPUjxrXGKsabi+abTgciL4n4W
bsG7LalURKcMe/HH1jVuy8VjevWJMB+u7CHOoc9jVCwR4YSPjH9fbcxIn9UIuECmlCryEUyB6kkh
BEeyxdQclP0ampPlyxVU80KN+Mxvle4//B6kwUtjS+/Rv943oXrXxaLXTCpeds49x0FWSRo/HCxy

```

```

nunzpkYqD4wBFUyB7hYggeRUaCb7aNVuIB1QZSY9EqF3F26AootzlcYprlCBtizZK9Q6Ez6N3iYW
ldMUB7dsNp4a4emAU3CfhHYh3JNv4pD21PbPASO/t89v7uMDrsI8SOpLnHqV0hYG2+JhxNyhYKVL
oXv54mKzBw+4vwsU/ySrrexUvmkTzLcSYBI7nSZT5uVprRA+MQJBLx6dKVVuzO1x8hzTv9T2LvJr
7rpd6Ban94JJ8vG7OU000aNP9HDrz+34xmCqQRi/f0TkmfSo4uFcsIfAmdQVbd6uu22ZBoWqolaz
lBXjt5Oe2AQV51Zma53dlArSBLpvbg/RoMM7cMhnGn33DkSBDYU9rN2iApw0zswa/KJ/plr33Jrk
5YTL6wTTEuaG+UxVrtCxX4Vhk7syaOjI5dshRELos3ZeIJeQkAgS45H6cK+gjCq013qWDDnFHCgM
zYoP16651C7c9T0s8i5OBLM6hGqgEgcKEiTIp+tRUvDEhyNlv/YngT3izvWbsijV0QTTJcjsyFWq
DSJiw8G1WH4oFqZAZf2UzE6fzEeQbMv1PPxlnpUjipTqdtWcuayLH7tifX7diBl1fj1UOTqPK2+5
vz1HckVtZJMS4g0W7rWHAbTv5nfrby/1IJBHMDutji2dh6J7nXbSgFOit98TFL7upJcNc7T3AH4j
Ro1TzXqODFShamQeYl0oCyyvStQxqj08rz74+7ery+GapNEPL4cPZ1qV0bfKCBwoQrTV81IZXsFt
Jj9TV+71T6ZcePnCFY6pWI78u5WwepZunmI9FFhz+odZd4vfh0C3VEISmeEN28T8XdvTht8A78sr
4/SmrPteZpZhByZe2n50ZHQU+ukncDgZirtz5A4LIbedcDLcgeNfonHYCQTYNOKoDA+eq5sBczKP
mqFKjPnBq1533/lptWhsgou8CzfsEaY4kZvEzK8YTVrfvt4T407A851vKxBfHIYXKxFFi17Yddr2
SigeBAUjT3waPAoUwgDjeldYTnnKUQy0Zm25gGRDiE9LUwoOp7ys0H9m/xXJROx76gbljguU3ad9
fcwQIm8RTKZvXvKrVRBUshutEL6/qZAb5VBQ1JHsa4tknAFTdwh7lsB1/101htZ+HzBdgZ8kOvRm
HiCKYb+2p26WVMy8SRhW8EeYxx3t79LMU3pIp9w4rCnuClwAYAXN6PP1Gf5GgsGS228ur3vwNKO
8YZIdMatmKJdy8Ufkm1Ljvy4Z0/3+XcGLDWyxRx6M2mLvMPvJIz9iGSr684PRfSydR3nq6W7gwYc
Ohb62cmSLVWYECoa+acqVFFGOKHCUT3ZS7Xlx0QkniCQI9d46XDEx64PFGeBXL/z4dj7ZYx6woX9
R+F5yOAdKoILV5N9m4xzaUP04EkMKakDBtsf9tzExrArDBoT664Xc7cVJ/2jTzX57Oms09Q7r+T8
hH0JNxxhcXAqhxdbMitkcFSy7t0pBgrPXRhdXohbG1huZPAOMVkwWDMf8x7Yc4k7F319ua67w5Z2Q
cdF8NBq5iYM3TkB+2qpmm16L7Pbp5qlAoIcB409+6VwxHiHqgBHOPGsPlxHNYGyYKcfr4VxaUUXf
5G18b5N0nx3S2VCBA9fJGXLHqW3RmtlMEP4dEqdCbh7jw7jd5E10NabRA0fCBTAYR61vYa90v7S
DOIEfy6NpDffg9sFltOa36ag==</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </wst:RequestedSecurityToken>
</wst:RequestSecurityTokenResponse>
</soapenv:Body>
</soapenv:Envelope>

```

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. 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.

- 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 ...Id="xxxx" ...>
  ...
  <ds:Signature ...">
    <ds:SignedInfo>
      ...
      <ds:Reference URI="#xxxx">
        ...
      </ds:Reference>
    </ds:SignedInfo>
  ...
</ds:Signature>
</saml:Assertion>
```

The XPointer format, used in Id and reference URI, shall comply to [NR18]; it is not additionally constrained by the present specification document.

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

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" Id="xxxx"
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/TR/2001/REC-xml-c14n-20010315"/>
            <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
            <ds:Reference URI="#xxxx">
                <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+S3ejT2f41KIGkVDcsRNYUqaAn60CnJiN4RBpwcjcWQSUj5/Xxesar4n04CtDylaLV6acLwww
lLN5PQ66UumASE=
        </ds:SignatureValue>
    </ds:Signature>
</saml:Assertion>

```

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

6.4.3 Authentication Use Cases

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

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

1. **Federated IdP - local identification:** the STS of Federated IdP performs authentication from local user registry
2. **Federated IdP - external identification:** the STS of Federated IdP relays authentication request to an external IdP
3. **External IdP - local identification:** the STS of an external IdP performs authentication

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

4. **External IdP - external security domain**

These four use cases are detailed in the following subsections.

6.4.3.1 Federated IdP - local identification (Default Case)

In this use case, the RST with password is used; it contains no IdP identifier, so the identification is performed by the STS.

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>
        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>
```

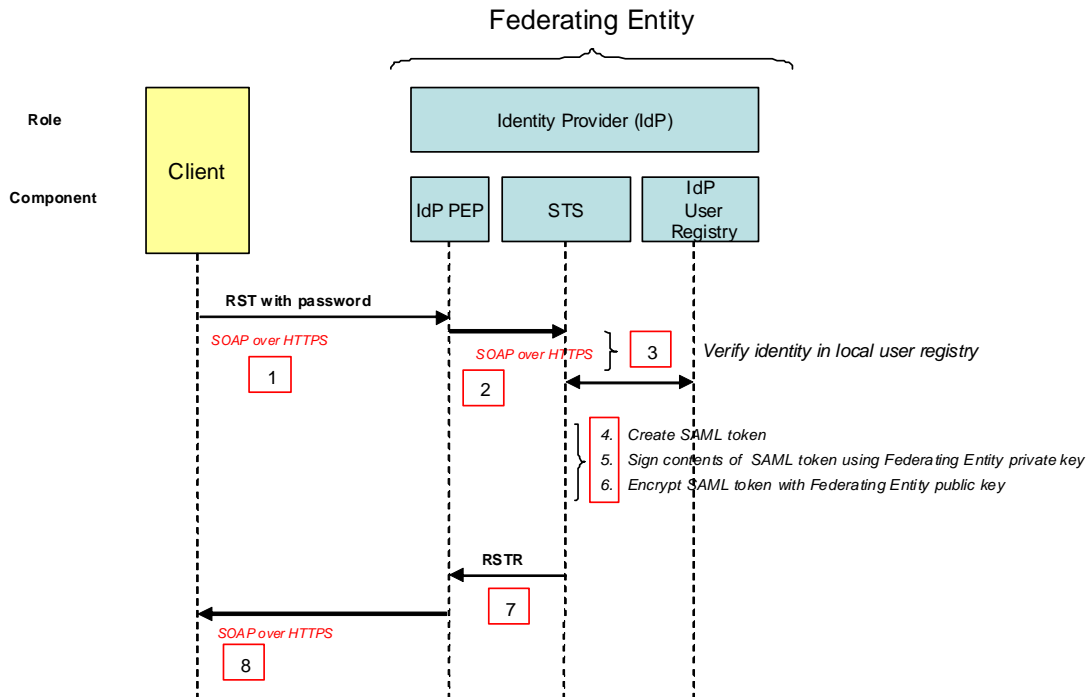


Figure 4 Federating (Local) Entity is request designated IdP (Default Case)

1. The RST with password is sent to the STS using SOAP over HTTPS. (May pass through the policy enforcement point (PEP) of the Federating Entity.)
2. (The PEP of the Federating Entity receives the request and forwards it to the STS of the Federating Entity.)
3. The STS verifies the identity in the **local** user registry.
4. 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.
5. The STS signs the SAML token using the Federating (local) Entity private key.
6. The STS encrypts the SAML token with the Federating (local) Entity public key.
7. The RSTR containing the encrypted and signed SAML token is returned to the PEP.
8. This RSTR 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 PEP of the Federating Entity can decrypt the SAML token.

6.4.3.2 Federated IdP - external identification

In this use case, the RST with password is used; it contains an identifier for the STS of a given external entity n . 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 authentication server.

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>
```

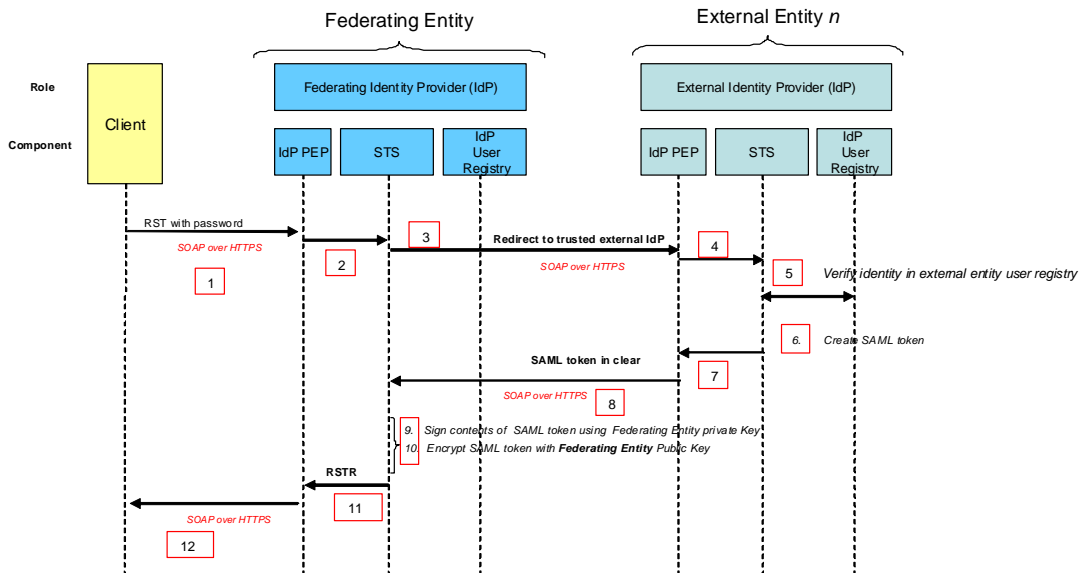


Figure 5 External Entity is request designated IdP

1. The RST with password is sent to the policy enforcement point (PEP) of the Federating Entity using SOAP over HTTPS.
2. The PEP of the Federating Entity receives the request and forwards it to the STS of the Federating Entity.
3. The STS redirects the RST to the PEP of the designated External IdP. The URL of this IdP is extracted from the table previously described.
4. The PEP of the external entity forwards the message to the STS of the external entity.
5. The STS verifies the user in the external entity user registry.
6. The STS creates the SAML token using the attributes retrieved from the user profile in the user registry.
7. The RSTR containing the SAML token, in clear, in the SOAP body is returned to the External Entity PEP.
8. The RSTR is returned to the Authentication Service of the Federating IdP, through SOAP over HTTPS.
9. The STS signs the SAML token using the Federating (local) Entity private key.
10. The STS encrypts the SAML token with the Federating (local) Entity public key.
11. The RSTR containing the encrypted SAML token is returned to the PEP of Federating IdP.
12. The RSTR is returned to the client.

Notes:

1. The client is unable to decrypt the content of SAML token present in the received RSTR; only the PEP of the Federating entity can decrypt the SAML token, by using the local private key, which matches the public key used in step 10.
2. The confidentiality of the SAML token provided in clear by the external IdP is assured 1° by the HTTPS protocol, which encrypts the SOAP response and 2° by assuring that the requester of the RST is the Federating Entity, known in the circle of trust. Actually, about the last point, the rule is:
 - if** _____ the requester is the federated entity,
 - then** _____ the SAML token is returned in clear *(present use case)*
 - else** _____ the SAML token is encrypted with the
external entity's public key *(see next use case)*

The mechanism to identify the requester as a known federated entity 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 Entity and Clients that access External Entity directly (provided that this last has its own IdP). Also, the system scales up seamlessly in the case of multiple federating entities, should this need occurs⁴, sharing one or several federated SPs and/or IdPs :

⁴ The design to which this specification applies has a single Federating Entity, which can have multiple clients. These clients are typically, but not only, Portals. For example the EO-DAIL (Federating Entity) will have at least two clients: EOLI-SA and DAIL Portal. Instead of having several federating entities, nothing prevents an existing Portal to become a client of the single Federating Entity. For example the EO-Portal may become a client of the EO-DAIL.

the external IdP should simply know a list of authorized federating entities (instead of a single one) and check inclusion of the requester in this list.⁵

6.4.3.3 External IdP - local identification

In this use case, a Client interacts directly with an external entity, that has its own IdP and user registry, bypassing the federating IdP. The Client sends the RST with password to this IdP and receives an encrypted SAML token, that can only be used on this external entity. The process is essentially the same as the one described in the first use case seen above (Federated IdP - local identification").

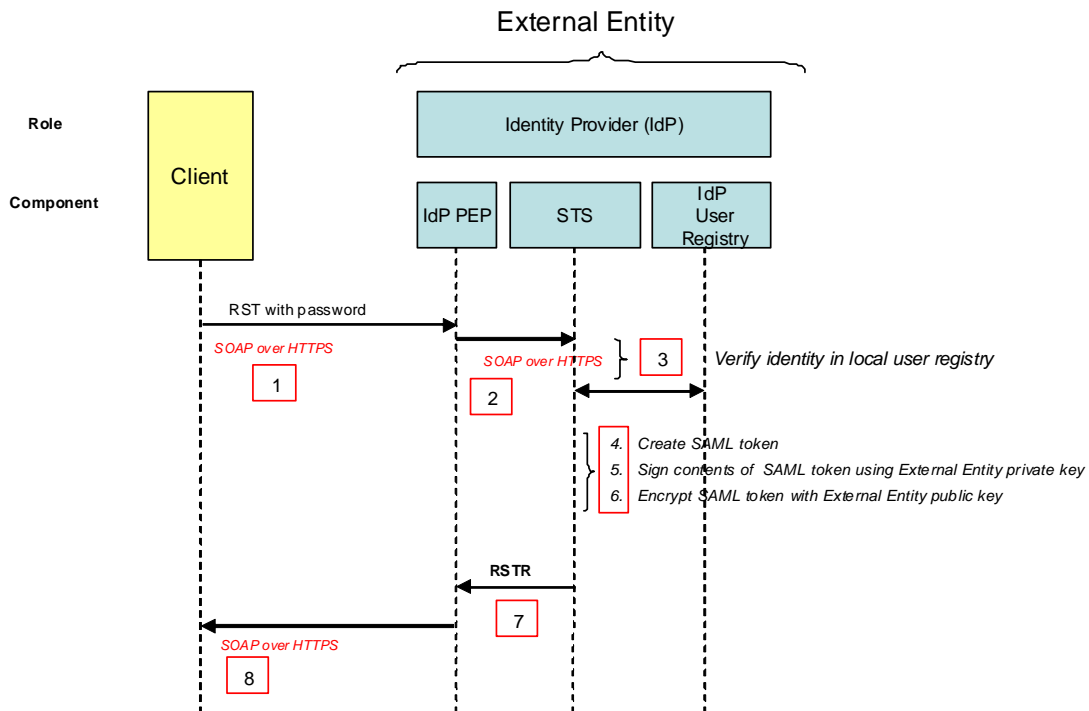


Figure 6 External Entity is IdP

1. The RST with password is sent to the STS using SOAP over HTTPS. (May pass through the policy enforcement point (PEP) of the External Entity.)
2. (The PEP of the External Entity receives the request and forwards it to the STS of the External Entity.)

⁵ Note that several variant mechanisms are feasible, if we allow the inclusion of *multiple SAML tokens* in the RSTR and/or authorisation 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 authorize 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.

3. The STS verifies the identity in the **local** user registry.
4. 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.
5. The STS signs the SAML token using the External Entity private key.
6. The STS encrypts the SAML token with the External Entity public key.
7. The RSTR containing the encrypted and signed SAML token is returned to the PEP of External Entity IdP.
8. This RSTR 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 PEP of the External Entity can decrypt the SAML token.

Note that the choice made at step 6 to encrypt the SAML token with the local public key is made on the basis of the decision rule presented in the previous use case (here, the Client has not been identified as the Federating Entity).

6.4.3.4 External IdP - external security domain

We cover here the case where there is an external IdP in an external security domain; it does not comply to the present specification but it is trusted by the STS. For instance, ESA UM-SSO defines a specific security domain with its own IdP (see annex G).

In this present case, the RST with signature is used.

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 *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 put 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.

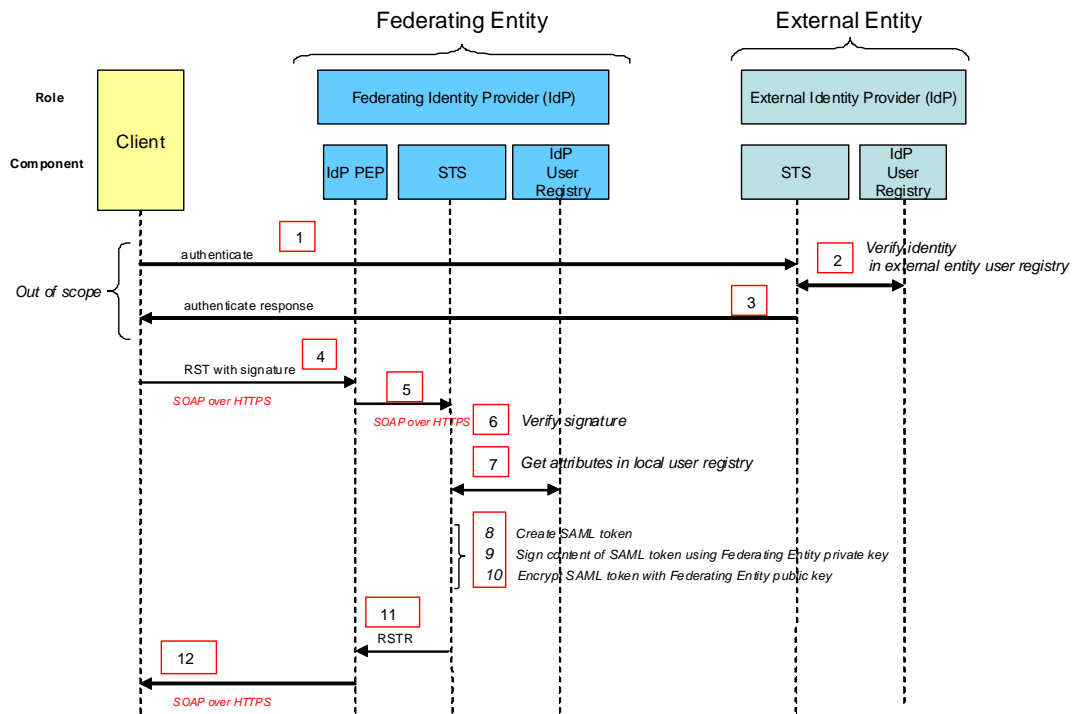


Figure 7 External Entity is IdP on external Security Domain

1. An authentication request is sent to the external IdP.
2. The external IdP verifies user identity in the external entity user registry.
3. The successful RSTR is returned to the Client
4. The RST with signature is sent to the STS using SOAP over HTTPS. (May pass through the policy enforcement point (PEP) of the External Entity.)
5. (The PEP of the Federated Entity receives the request and forwards it to the STS of the External Entity.)
6. The STS verifies the signature of the RTS.
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 Federating (local) Entity private key.
10. The STS encrypts the SAML token with the Federating (local) Entity public key.
11. The RSTR containing the encrypted and signed SAML token is returned to the PEP of federating Entity IdP.
12. This RSTR is returned to the client using SOAP over HTTPS.

6.4.4 Authorisation Request

The authorisation 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 authorisation 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 specification. 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 Multiple Relying Parties

The present specification is limited to only one unique relying party. This constraint is caused by the public-key encryption of SAML token, which entails that only one defined entity is able to decrypt the SAML token (the one that owns the private key associated to the public key used by the STS for encryption). The specified architecture obeys this constraint de facto since there is one unique Federating Entity, hosting both a federating identity provider (STS) and federating service provider (PEP authorising federated services). One possible way to extend the specification is by adding on STS the ability to choose among several public keys for encrypting the SAML token.

For this purpose, the RST should use the optional AppliesTo element, which is presently ignored (see Annex B).

The future mechanism could be:

- if the AppliesTo element is absent, then the SAML token shall be encrypted with the public key of the federating entity known by the STS;
- if the AppliesTo element is present, then the WS-Address shall be used to retrieve the public key of a specific relying party and encrypt the SAML token accordingly, so that this relying party (other federating entity or GS) can use the SAML token directly.

6.4.5.2 SAML 2.0

The present specification takes SAML 1.1 [NR10] as baseline. However, it is foreseen to support also SAML 2.0 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:wss="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 8: Example 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 9: Example 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
dU14K+TffynRJR9Tr9PD8YIdpFLzCvYas63g5x4/XnyALe2AU8ZBBpM2dtbr3g4TYMywfraWrI76
mHM+MERVZdHmVBWFrhqXhcs92m23m+amt14mk=</xenc:CipherValue>
</xenc:CipherData>
</xenc:EncryptedKey>
</ds:KeyInfo>
<xenc:CipherData>

<xenc:CipherValue>VEHlprDMQ+DqIpoPqx6TYi/mMX2dGV5JCCrhdQuZHRKqOiaIFfwqOMZvn
2HW2JDFvUxJ6LRTKdNuJQI7sxc6h3IGBL7NXF7bx4jGwQ09wAA7nm6OoB4jjiGdaqb8wTx0o1nzn2
WqOWoVeTngl1wBi0rv2+id1HwnXAUUHfJH8ALq4IU3hr0vjoqJH6Y21EuXPeXp/dyPUw3oIFn2FE
ID2u+8T+xOxbBq2ezQbU3z8n1LbgvDtN3ex51UCo260pOOPn92nn7nYerT682eY4+bCKoiENpQSY
gHszvvyqFf9o600u87zk4AORwsRhQH74L2gG8wVOeHKYhEx0RsBkf4xZcQKBvQ9JHWQWpDEB51NZ
aJelhSyaUk6T5Gf9ArDnz6UwL0ZTDp6DxgJha91u5qIMG3ECxVYKcnBv+O6OmlQOHbL0echUDR56
evS+mf0U9JxduBKwFJLqta6D0mwmgYwcaF3ZrKd7SatV8Z210DmWTMe5R+x601RpbK1tlduK14bL
aSYFpaqU758ZsmTDMjQQj8fn1qCZbDtp4SEVPwumoTg2k7RAOay2QtV5b+VA9wloSXoxVf2csLS
OOH/NDElnoB1pzgUb9Xm/YIPwikQKsXNPFM72yLrS0vjAho1Ccxrg+817XIVcmowhPnLqSs6ZpvA0
1YP8EhsOFLN+0y+9EfAuoY4jYcScfwqDehth761ER+EyAdFLi10VhVxKW14VLbmksAyndIQaw6V
zGmlQwoc3CeCaeq4q4GgFgiem1BmW9IeBaUBTX2wZmIKG8Z9Xhjv6MwT7hOeWH5fefipJs8J5S816
wQBo8WAczzmw6s1j8JW9YDyAWosfoTPrtOwFTaaYSiaEXvPOnb5RgR/W4ivZ64ioA8FXyLFoWcNE
J76AgWHDLabCDg/zvnVwEs70daSxRTxVNS77cplGspSmk/HzGYxPHInGhn/QPSas5iN6t6HlwnQ
UJgt81rI/tbFfSYqqtYqXKeNoEtw91/1DZVUI7mSc7Xj2e2Wb65h8PIoYeX3Nli+i4SroOeAKaZr
HtpqP6f+pi41pkANS4RFxFDiL9Ddxv1WKD//nMck0Su0HfIbPYUYF0GGv1Hsv6IiwT8dj/f0MnCx
kAgegl1GageZthQiNavOcURRC/94d+1jDZGayowurzdxmJhxyiEY5REQQt3hK4aAD89wMjndzxHd
tcQEuvXA5uSm3T9qgIm4Qdvuh54PW/SKptG9fdj4paTxVv1fZ+0f/1Vxjj4pPIKOVjE3e4ChBpkJ
XD/nXqZ8DDR+zPxOLWYyiqnMaxv30Ind/Iz2Lq36a09b0JefMVz4e39sGtFzNDbxXgQnTx4L3jDY
Fdl5+gelUNduK9Htgk1XDWfNIMWtY5xhdTX0m3Q8hBtNgKOHeg7BcBXf+uT30mqwgJu5cbJQ1/1j
/QlMvromUaUQATN2ULu7mMiTWkoYoMTiijJGAizbKIi6O5xmHvF/jicd71BcmSz+B+BnrnqxY5DM
/qQSFsnRoGmPK1JeiAo2g+QuMD5x7H+pBUIq3B8lkm1UBg5VoKx2+kChuP2lamGFskQ180PRGygQ
5ada2iWwKoIKoCdcYic9C2sPVkJz+slExJXizl4L51GEWdlQ8VGsqNV7CzOyIt0uXIIBQW3j0aX/
/7QoYVfM681TiqvtaDEY7Ip4nSV839e5xnj3s+qgzXOpok5r5EThDLhtPy97CJiUsbsfcgf1Dv
WNE74x2E4b8HazacItRBIbx0GFDHIoqaHEih6z1hQaqwloLnUHRpL8vAQLVKiW3q94569e3GenoQ
bpjxKQ9F58VuQh3ZiZtJ+17XOXDx6ZDXcTiQDa+3nXiTg7k9gGtpIv8vYLMuUHDEZx1zGd/rMzU
3JAbbFKO8+Cs3pMnb2KpGL4rLLGivee1P35rbHK04V8D0NBwDk0TOVnmFQWIRsgVtPwmEHXbF13j
qIUTx4xdishXmkKcDcWanfQYo4yzmgLUlbtchkDGF9YBA0mXv4gT7z0TiBb1jUFbHTnciL4DBkI9
K8wejkLHUi7w4LjvHCB5B15y1ZG/baIscRrvU41HU7p7crwbsdkjcwGE//dBTXN1vrXDM0maAkCo
nuYNPpMY0Cf+ikVITJ073UaXp1Fj00tm+mkql6e5nT8d8qQXwH21/nGJOEO/rMsXS0VybxIn5bg+
97CFctAdsRRjAJZRQcrJIztenGJX81U0rvAX+OuoSNrgVpdxcwuH/1x80i+CY5kdUkg3EMkU0m4F
iNQ6CyXiimVSRB0sHFwW5/Em+qlYeRjrxCYjYBPo2mCuMtqQN42VeShEkQ1XPx609NTaaxXRM
pV2IHjzALLrR0P6xqzbp7CuEhPdLxTYcXetDKQJt/XHJuWdvetMgsJnyQ0cCJSPXp21xsrK6zYLY
cQ1M8rS9RHcmWvFdtZg49mX3QPANoUDP0PR0y3m0T19FWKfYOfQhHN2xPJZAPV6ZJeReeAeTBRkt
vgIJE/3BsQpmqsSugEDYCrK8MfaybAC6CpE5ZKnQwV99Y1TcbPx7vVKPu13j3A4j4FjtgjkFun
fCOpLX1AA0FSBbFOOCVeYd94bCGaW8f+j3NBB+29ELYmskew2tyCBiw2HodBrMoDiYVWHbd+bWw8
qMOOBurEQihVdNq5Tbi3R2fnnX9Dpfbv1jJeKfjyVwCLZA50dGIYPUjxrXGKsaB1+abTgc1L4n4W
bsG7LalURKcMe/HH1jVuy8VjevWJMB+u7ChoOc9jVCwR4YSPjH9fbxcIn9UIuECmlCryEUYB6kxh
BEeydQc1P0amPlyxVU80KN+Mxvle4//B6kwUtjS+/Rv9430xRxxaLXTcPeds49x0CfWSR0/Hcxy
nunzpkypqD4wBfUyB7hYggerUaCb7anVvIB1QZSY9Eqf3F26Aootz1cYpr1CBtizZK9Q6Ez6N3iYW
ldMUB7dsNp4a4emAU3CfhHYh3JNv4pD21PbPASO/t89v7uMDrsI8SOp1nHqV0hYG2+JhxNyhYKV1
oXv54mKzbW+4vwsU/ySrrexUvmkTzLCSYBI7nSZT5uVprRA+MQJBLx6dKVvuz01x8hzTv9T2LvJr
7rpd6Ban94J4J8vG7OU00aNP9HDrz+34xmCqQRi/f0TkmfSo4uFcsIfAmdQVbd6uu22ZBoWqolaz
lBXjt50e2AQV51Zma53dlArSBLpvbg/RoMM7cMhnGn33DkSBDYU9rN2iApw0zswa/KJ/plr33Jrk
5YTL6wTTEuaG+UxVrtCcxX4Vhk7sya0ji5dshRELos3ZeIJeKAgS45H6cK+gjCq013qWDDnFhCgm
zYoP16651C7c9Tos8i5OBLM6hGqgEgcKEiTiP+trUvDEhyN1v/YngT3izvWbsijv0QTTJcjsyFWq
DSJiw8G1WH40FqZAZF2UzE6fzEeQbMV1PPxlnpUjipTqdtWcuayLH7tifiX7diB11fj1UoTqPK2+5
vz1HckVtzJMS4g0W7rWHAbTv5nfrby/1IJBHMDutji2dh6J7nXbSgFoIT98TFL7upJCNc7T3AH4j
Ro1TzXqODFShamQeYl0oCvYStQxqj08rz74+7ery+GapNEPL4cP21qV0bfKCBWQ1rTV81IZXsft
Jj9TV+71T6ZcePnCFY6pWI78u5WwEPZunmI9FFhry+odZd4vfh0C3VEISmeEN28T8XdvtHt8A78sr
4/SmrPteZpZhByZe2n50ZHQU+ukncDgZirtz5A4LIbedcDLCgeNfonHYCQTYNOKoDA+eq5sBczKP
mqFKjPnBq1533/lptWhsgou8CZfsEaY4kZvEzK8YTVrfvt4T407A851vKxBfHiYXkFFfi17Yddr2
SicqebAUjT3waPaOUwgdJeldYTnnKUQy0Zm25gGRDiE9LUwoOp7ys0H9m/xXJROx76gbljguU3ad9
fWqIm8RTKZvXvKrVRBUsHutEL6/qZAb5VBQ1JHsa4tknAFTdwh7lsB1/10lHtZ+HzBdgZ8K0vRm
HiCKYb+2p26WMVny8SRhW8EeYxx3t79LMU3pIp9w4rCnuClwAYAXN6PP1Gf5GgsGS228ur3vwnKO
8YZIdMatmKJdy8UfkmLlJvvy4Z0/3+XcGLDWyxRx6M2mLvMPvJIz9iGSr684PrfSydR3nq6W7gwYc
Ohb62cmSLVWYECoa+acqVFFGOKHcUT3ZS7Xlx0QkniCQI9d46XDE64PFGeBXL/z4dj7ZYx6woX9
R+F5yOAdKoILV5N9m4xzauPO4EkmKakDBtsf9tzExrArDBoT664Xc7cVJ/2jTzX57Oms09Q7r+T8
hH0JNxcXaQhxdBmitkcfSy7tOpBgrPXRhdXohbG1huZPAOMvkWwDMf8x7Yc4k7F319ua67w5Z2Q
```

```

cDf8NBq5iYM3TkB+2qpmn16L7Pbp5qlAoIcB409+6VwxHiHQgBHOPGsPlxHNYGYyKcfr4VxaUUXf
5G18b5N0nx3S2VCBA9fJGXlHqW3RmtlMEP4dEQdCbhH7jw7jd5E10NabRAOfCBTAYR61vYa90v7S
DOIefy6NpDffg9sFltOa36ag==</xenc:CipherValue>
    </xenc:CipherData>
  </xenc:EncryptedData>
</RequestedSecurityToken>
<wsp:AppliesTo/>
</RequestSecurityTokenResponse>
</soapenv:Body>
</soapenv:Envelope>

```

Figure 10: Example Authenticate 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" Id="xxxx"
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/TR/2001/REC-xml-c14n-20010315" />
                <ds:SignatureMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
                <ds:Reference URI="#xxxx">
                    <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>oOkdc3KB2HwPB6YzhEa9MHx5yolu/xqHp81wPj68uf5Ypet/5wHHEvfFuN
hxD+S3ejT2f4lKIGkVDCsRNYqAn60CnJiN4RBpwcjCwQSuj5/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 11: Security Token Service WSDL

7.2 *ServiceRequest*

Through the implementation of this interface to the ServiceRequest (i.e. the service operations such as the catalogue GetRecords, the programming GetFeasibility etc.) authenticated clients will send requests to a server controlling access to the final service. The request is made using WS-Security containing the SAML token previously returned in the AuthenticationResponse.

N.b. The service requests from a client to the Federating Entity or from the Federating Entity to a Service Provider are the same.

7.2.1 Request

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

7.2.2 XML encoding

The following XML-Schema fragment defines the XML encoding of an example ServiceRequest operation

```

<?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+nBkutOsmP9Lm6v4gpPvtJqx00JLE0oKCQfE4Q7qplYObkKRlu
j9zb7Y07cNdJf80hzGaHFGz70IfM9TPl5QEEntkoeOT9wg/PsYqlIaAsCRoDsYjJoQrqpHdIpv3w1
Cck8iysQus4LpqdK
Hc6pWRk0Gk9022z/3U=</xenc:CipherValue>
    </xenc:CipherData>
    </xenc:EncryptedKey>
    </ds:KeyInfo>
    <xenc:CipherData>

    <xenc:CipherValue>+X00FMae+fv8zOr0pPA02icglYf4AKcaml/jNfP8gdmjIh/dB/utVIC
YxKtarBRSAIptozGoI92r+bwUwmAyIY3D7gX0h6EC0P3LqhojKiMRrNbvCaotOPWmherMplSubx
eYgxdZVlpXa77mNHHEkjhcmNXHydggz4DJoLxzhUIDWm9Lv+UTuFH+D680Jic00SCnGdIC6KEpM68
h+3x/PRRNdmy
QRpS9WZ03DAADBokRmW8IbG5Y1UG4NjZDhkPDR1FhaHBTn4ZDP4LEd98KXZclwAlAB9XIICteNFh
9t0itufclExX0zu/icAM8ws/sEAh7NmLyw+k8MRI7lMXeldnqftBYyJ5NzYZqUd87XXqTe6ytnS3
SwbZXgdkgKylqdp0p0FcJVOGb4obfP/6lrwzfluJk2DMJb+9+mTQzfdNIXimegV5Y2r1Wsg7Xt
xiVU6TFMI6VA5CH1MkmgYyYFqgI2MoiNXW3c3sgAs6+Q1RoPMR3uNzvtB7NKy0m9BET+zqxCRgPt
GPstjX5ATvJ7tbcAlSKGyHubEIE9Am1Q2nGv3ChZgPPw+rwtow1D8xeSnxWOKpp/whmXcN9AEQ/z
5HtDCmbw1+ExRTM8Xy1NWp135If/6ooxcJtOf5vavo5Mx1OQt08TieF35+5FXA0rUiCn//yJzJRz
2mXEMJfo01HfwPpfGWxId4yuhWeylNAA4sKwt2OVdc/zkZpRTIH0oOWuut2LdsZS1fBz+RMnpt/
u8tIvsRITLyd2htIILXKIenNpdRWeUD4d23RxcFFt+bGh
yrbHnsrT+DIzJD0Pfs0zxigXu+NG4wy+Plhjl1h4pn2AosIP5v0ZXN/tObsQonwKyjFwgqGH8js
Ik/96PLnu1ODRRYVIBOglcV9K7NrHeqCkM1157HCwu/rflTXK61jzRsZ8/hzC8ADiOXQnpk8K4EO
AFs/6LY54A8MFQndHkAHN6NEK0nbAkqhOTur99PhrXQtYfsf26Mrd8rTKhkP5zd3pdfzvhnqOnx
Ose2FuWX3WHwUTgZAMB1JC2PRzHM4Q4q/pHFYk+UrLE2QLYsBn6VzouHfCI3dikr/0d2RQPsRQKQ
PB8WXMjJxK7v05jRBjZaNYpmsFk08zweM1OWuVB8L57zSzAfb7CKpPgsgRk4ezLwrPVK7Z2UuQ+z
/UH66S3a9dYsneQMadMh7wYQtLe3fEeUhbBYrjRBZWHrriOnhx0N7R118bnK0XoJUJmZqyZyvN7q
HTLUxG98KUncu3HYwKSIGZEog
ZhHsfbq9jee0tx4MhW+t8z9Upsh7TPhWcEvaFxp5pfz4c704nsM7Tmcq4IijlnW8m3kX/mBR/O
TFcjWh2mV8XZoa/Hro7Rj69HVjELBTlF/W+S7pNsN+hoErrDlWxuHqC6v7KDMakaLF/Cekz
WA27ozxm7z1+6a/BeiXSTNojoodOybmV7hxjObWg53RUp5H3rejnN52+7IDHJwik2DONIERqL5GP
Soy3+adE5mSnAQklZ4zeErs1/A82ySgovaQNskuaq+6aBLvhHQ9CpeQmlddofyU6HebMSN2mE8OZ
2yejwhujnt5ha4KoeFrE21bwi5xWkNDOBjbfBPXgg7Wdf4M6n3zsRTT6ixugXrkdRhnYyTzprJ6E
PpL5Cduh2gyQHiyJbcCh37rzTcsx0CiWHfak0ObqDRUeJ8tPyOS5PhyxcNknQ4p3RCI0QJTxUYG
3jpwntAK7ZU3d0Bmk+DAaPGGGJ44fKZ3HZTjwnFTcQWxYOXxsiJ/kKE8ZVcDJ6IL4pf4dTnJHaa
8ht09LvtVZJrcYbI5q/QLlhcMclPBByoap4EmFPxX3dppap0uf2qbX80G+jjVtshd9rhEmyoc6
tJjj27Z2B0ANPAI53AMDXXGF5HHHzzfifizN0vK48EO/xk34EEy1SmCIInrf72m54f7wh8Rojgo0
zIWZkIU+tpCf02HTcxRUT/rd6Wfb624YE9ov20+T1U0Yc9nyj
zoNDNBjCXh7+tr4FkUoZGyzqm50lYkfkKvkwk6p6rH8RzUHQKewHjWb7zdlYHEH8XkMtcHneYgeQ
pDQ/E8bshTuLILoELUtRODjszaTXyrm4x1hCzJ7mWlZa+viTV4PzHdRQCvByGxOsFfhJGtrOOrx
SUnyUNVDBbKx3Te7tzuTtZ9demhJAE17svhC5htIWg5NMJ3FeG4HzOL9Uoah4gwyvFa/0azdt
1ZWaYc6SPufQIwK9I2HWRb1m4waiA24LkLBXvYmJWtto+DsVWPP3WQXtPaSBntnd/VEbM+2bbPER
Al+drMTui6Gv1/iHNQ9uq+UrmXPOR9ntfSEAh/M5BdSH75zGifd369R1eFJqBuBwile2R7ryqnPP
BbVf89md0nhYe3RzdD0DKbJen5/r3eicrc3Pcu1W8cGJDqtuEI9kC1xULyXuhWmpEACgTabmNmW3
T+3LnzEuKU97DtLpgqlJoAqXHBBImDsPmzX
JID0oNc7J8ouh53L8ZM6jZwFXGqQBteV0HsDPwxTBSGE+tuPQHj/cWvovOBeltewUuBskG0EwsDt
kYwmP5yNlbY4vn0ouL+4tlprZr2oNSitv3Lle/usE5ps70ALpQYvzG369DAqf2T+m3Ld5HaKZ/N+
cWtQSt/EJGjjBTodrzbkAe7Mkz/euMxU0Pj5Gv4kyLysfivPPuvar+ZuRos/jm5N+mHUQUzWd2i
zk4BvBuyWHNe3Jq45H/ELAND00EZR0XCRbpbz0io+a9c6T44B1OECJyXI1gp/m9lsbb5iSv0HpmSv
4xsLpM1AuXLPkmeqdsHO/zEnLU0hR2Dpk6hoqpnPvK+QbVO0c/YdJ+lkeGIz6C20srbb5io+cUou
1+7mJ6Wg7VqiEJWNX8mzd6RklCnt0WlCgBk93vzOspDjEnvBkHSZ1VmuiWLWpEstUrYcwF77lc
SLDR3rvqa29hUORrV4BRH11LAuf9ofAyg9r3vb2TjlrG58FOekzRxomjP/rTL0jteYiBf6YwoMEw
g2chC31hRoatpzTpMSbAoByuI3VCSqMN81JwEAU0tXmjrFg/C8Cnc1/Zg8tYfntIymNbzH9S5Mmo
Qy2oEUAM/RmAUIyhqEE+J1q4wBfJb933T8oPOQBgBnjntcDsJXwQt1xa/QxDs7d1TaGMkzby6YxZ
Qxt0s6cTwyda2Xwq06Bd9pugluF4c41218g+42PTzwhWpTcXbqOaQDLVITfM5LWK7JvAEK5fai9X
c2dooFiNiR+QUG3SwRrZ4GZ1d4wWgOJsPexgEkYOKQuwo6S1vl0WNZtuYC7/+ct4qiHzA2tk+5HB

```



```

vMoZlWyxUQNxo/sfhRx5xK7lT2vHqBx0klgwoZW718/IPpbo0frpdKT2iOGLB/YH46GkWztp0Ft8
+7uPbFebu76teSF04ei3utFf9h+UmcxgNmtGRlBuILes5ERKI2KDLfr90+ltKhDZu6gOBrCOWxik
x+bhojouvj0o+LdZt9zjSaNPZTKym0lzpPoFv24xyAA30UAc1WESHKcuPbw018LIopmURoEROB3dY
N8veuTfekmYPv3tHOaLdZaL66oJklarJBcHWYlr/ob0/gElFmn+20y/kK7oq9vEP/oOSMYgtiyCW
mBEgcnm6rIQvklFxz9FFMz06+2I5/WOOSRnr371Pb1nukHFXXHC5bDRMbnR7JobKhPacDibz12i
Nt/uWNX3K7L3Ddh1hCHFF/Dl+won2HJsfiTovbfXVoL3fs1Rk6+FXvO5QRqcrQVOKN/z2cn2Y8N/
bLlr86AH3+J7r4fGAspyqx985VMKzz15OHvi+DzGDmmuzgtHpB3/R0NRbWgbW5
ebpeduehzmzGQ4vL3KbrrbH+QkU2DIlcp+DOYsynVNtDx1fJVSDfvHxBaeOYwm9sjzvrslpHMqkl
tSmiqOnuU/shfPtAdYyxoTTDV11R+TJNQo80Mq7cJUd9NeiYi+Tjorpn5qtJW9/XQIPQjO
riuWkK3d/mv4dWGWQkS14CJ/5Em4ONdEJnJzwU4FndrLGh76IWczBM+3grhCVWBWF5EohxV8rMEJ
D7m3HeP6koPo4uxTylRkhsFO8GgP0aFR5cEGsjnIhyPVcf7ad1L9t+A3ajzpPW9m+pcdgWqvamCT
47B/Uc6S/nN8VA+7bIdXVCqTsNiyynNSEplk8Qi97nz2ZF6/UcxdoD6aVD/HvJA53usmluCKyul
nFBFX9eIyOF0DGppo3RsP8ka6lpSt+jXrn95xkis0lu/Efmt81b0bhrPET+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="urn:ogc:def:e
bRIM-Slot:OGC-06-
131:multiExtentOf";]/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="urn:ogc:def:e
bRIM-Slot:OGC-06-
131:parentIdentifier";]/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 12: Service Request Example

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 *ServiceResponse*

7.3.1 Synchronous

The service response is as defined in the corresponding catalogue, ordering and programming specifications.

7.3.2 Use Case: User logs in at client and makes Synchronous Service Request to Federating Entity Service

In this case the sequence is as follows:

1. A user already registered in the local (Federating Entity) user registry logs in at the client (Federating Entity Client or an external client).
2. The client authenticates the user through the Federating Entity STS.
3. The Federating Entity STS:
 - validates the user,
 - creates the SAML assertions from the information held in the user registry
 - signs the SAML assertion with the Federating Entity's private key.
 - encrypts the signed SAML assertion with the Federating Entity's public key.
4. The Federating Entity STS returns the RSTR containing the encrypted SAML token.
5. The user is given confirmation of login.
6. The user selects a service.
7. The client constructs the service request and inserts the encrypted SAML assertion in the request. (N.b. The client should manage the token validity as it is possible that the token has expired and will therefore not pass the PEP checks. To ensure the token has not expired an RST could be requested at each service request).
8. The service request is sent.
9. The Federating Entity PEP decrypts the token using the Federating Entity private key and applies policy checks.
10. The request is forwarded to the Federating Entity Web service.
11. Service request for the external entity is constructed and the SAML token in the header is encrypted with the external entity's public key.
12. The service request is sent to the external entity PEP.
13. The external entity PEP decrypts the token using the external entity's private key and applies policy checks.

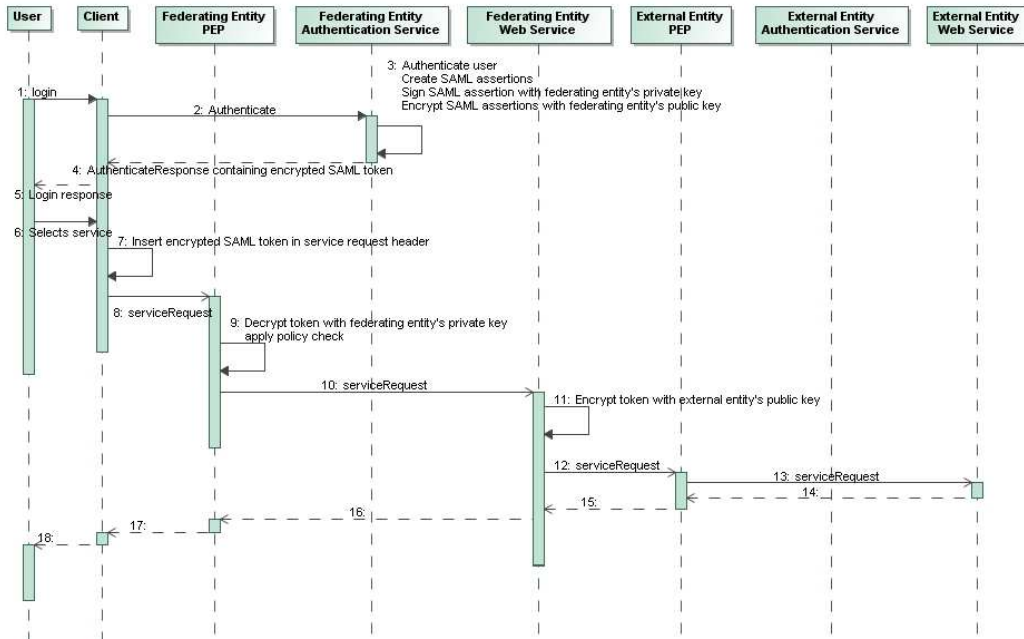


Figure 13 Federating Entity Sequence Diagram Showing Client Request with synchronous Response

7.3.3 Asynchronous

The asynchronous service response is as defined in the corresponding catalogue, ordering and programming specifications. This response may be protected by the same encryption and signature as defined for the service request and authentication. The sequence is as follows:

1. The SP prepares the response to the endpoint mentioned in the WS_Addressing.
2. The Service Provider creates a token authenticating himself i.e. external entity and signs it with his private key. This is then encrypted with the public key of the Federating Entity and inserted into the asynchronous response in the same way as previously described for a service request
3. The asynchronous response is returned to the address provided in the ws_addressing of the request. This will normally be the address of a PEP.

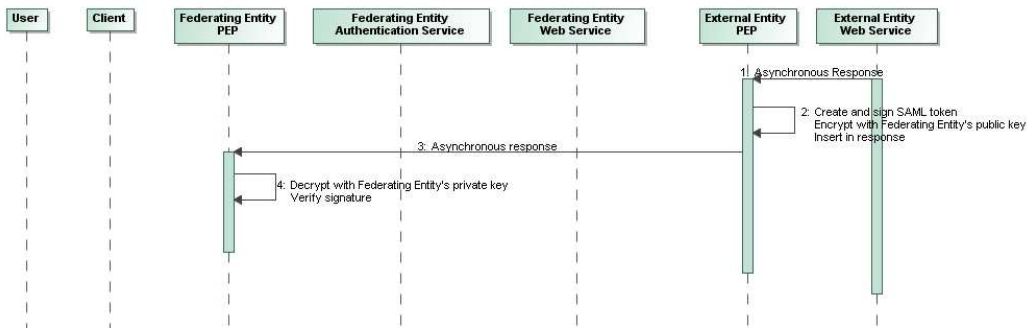


Figure 14 Sequence Diagram showing asynchronous request

8 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
<i>Identity Spoofing</i>	<p>If the IdP is the Federated Entity or an External Entity complying to the present specification (see cases 6.4.3.1, 6.4.3.2 and 6.4.3.3), 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).</p> <p>If the IdP is an External Entity not complying with the present specification (see case 6.4.3.4), 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 Federating Entity STS. The action of registering the public key of such External entity on the Federating Entity STS means that this STS <i>trusts</i> 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.</p> <p>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.</p>
<i>Man-in-the-middle</i>	<p>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.</p> <p>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.</p>
<i>Data integrity</i>	<p>The signature protocol enforced on SAML Token allows for the verification of its own data integrity, at the PEP level.</p> <p>The data integrity of the message payload may be checked by another signature mechanism on the SOAP body. Such signature</p>

	<p>should be bound in some way with the SOAP header, in order to avoid the risk of forged service request (see "identity spoofing" topic).</p>
<p><i>Data confidentiality / privacy violation</i></p>	<p>Encryption of SAML token (both for RSTs and service requests) guarantees that no entity excepting the target PEP can read conveyed user attributes.</p> <p>For service request, the data confidentiality of the message payload may be enforced by using HTTPS protocol or by encryption of the SOAP body.</p> <p>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.</p>
<p><i>Replay attack</i></p>	<p>For RSTs: the transport protocol is HTTPS, which is based on SSL; SSL includes a "nonce" mechanism to protect against replay attack.</p> <p>For service requests (if simple HTTP is used): the risk of unauthorized 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.</p>
<p><i>Denial of Service</i></p>	<p>Web service is susceptible to message flooding denial of service attacks from message replay. "replay detection" mechanisms can be used.</p>
<p><i>Password Disclosure</i></p>	<p>If the IdP is the Federated Entity or an External Entity complying to the present specification (see cases 6.4.3.1, 6.4.3.2 and 6.4.3.3), then RST with password is used, which relies on HTTPS. The password is therefore encrypted during transmission from client to IdP.</p> <p>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.</p> <p>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).</p> <p>If the IdP is an External Entity not complying with the present specification (see case 6.4.3.4), 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 specification.</p>
<p><i>Password Cracking / Guessing</i></p>	<p>If the IdP is the Federated Entity or an External Entity complying to the present specification (see cases 6.4.3.1, 6.4.3.2 and 6.4.3.3), 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).</p> <p>If the IdP is an External Entity not complying with the present specification (see case 6.4.3.4), 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 specification.</p>

<i>Unauthorized access</i>	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
<i>SQL injection</i>	<p>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.</p> <p>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).</p>
<i>LDAP injection</i>	<p>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</p> <p>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).</p>

9 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.

9.1 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.

9.2 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.

9.3 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.

9.4 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>
  </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.

9.5 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.

9.6 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.

9.7 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” specification, 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.

9.8 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]	<p>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.</p> <p>Messages should conform to SOAP 1.2. The following assertions shall hold:</p> <ul style="list-style-type: none"> • The SOAP Header holds the authentication token [if applicable], embedded in a WS-Security element. • The SOAP Body holds the message payload.
Test method	<p>Send a request embedded in a SOAP Envelope over the HTTP[S] protocol; verify that a response is returned (<u>even in case of failure</u>) embedded in a SOAP Envelope over the HTTP[S] protocol.</p> <p>The SOAP Envelope shall be compliant with version 1.2 of SOAP (namespace http://schemas.xmlsoap.org/soap/envelope/)</p>
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]	<p>SAML 1.1 is proposed to encode the user authentication token. WS-Security is proposed to encode the SAML assertions in the SOAP header.</p> <p>A SAML token is made of the following statements:</p> <ul style="list-style-type: none"> • 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. <p>The set of attribute statements returned in a SAML token shall be defined arbitrarily.</p>
Test method	<ul style="list-style-type: none"> • Send a valid RST to [the Identity Provider of] the Federating Entity; the response shall contain a SOAP message whose SOAP Body holds an encrypted SAML token. • Decrypt the SAML token using the Orchestrating Service Provider private key, and verify that the SAML token has the expected statements covering the (arbitrarily) defined set of attributes. <p>Pre-condition:</p> <p>For carrying out this test, the client needs a copy of the Orchestrating Service Provider private key. Since the Orchestrating Service Provider is a component of the Federating Entity, the client ultimately holds a copy of the private key of the Federating Entity.</p> <p>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 Federating Entity itself</p>
Reference	Clauses 6.2 and 6.4
Test type	Capability
Test case identifier	urn:ogc:cite:ats:um:0.1.0:07-118r5:encryption

1.1.3 ATC-1.3 Encryption algorithm for SAML token

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

Decryption and encryption of SAML token is performed by [the PEP of] the Orchestrating Service Provider during authorisation request orchestration [assuming that an Orchestrating Service Provider receives all of the incoming authorisation requests].

Decryption of SAML token is performed by [the PEP of] the final Service Provider when handling the authorisation request.

Test assertion [purpose]	<p>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:</p> <ul style="list-style-type: none"> • 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	<ol style="list-style-type: none"> 1. Send a valid RST to [the Identity Provider of] the Federating Entity; 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 Orchestrating Service Provider private key. 3. Decrypt the SAML token using the AES-128 symmetric key and check that the result contains a valid SAML Assertion <p>Pre-condition:</p> <p>For carrying out this test, the client needs a copy of the Orchestrating Service Provider private key. Since the Orchestrating Service Provider is a component of the Federating Entity, the client ultimately holds a copy of the private key of the Federating Entity.</p> <p>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 Federating Entity itself.</p>
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]	<p>The secure hash SHA-1 digital signature message digest algorithm is proposed. The SAML Token is signed before encryption.</p> <p>The XML signature <ds:Signature> element of can be used for signature, according to WS-Security specification.</p>

Test method	<ol style="list-style-type: none"> 1. Send an RST to [the Identity Provider of] the Federating Entity 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. <p>Pre-condition:</p> <p>For carrying out this test, the client needs a copy of the Orchestrating Service Provider private key. Since the Orchestrating Service Provider is a component of the Federating Entity, the client ultimately holds the private key of the Federating Entity.</p> <ul style="list-style-type: none"> • 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 Federating Entity
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 Federating Entity 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 Federating Entity explicitly indicating the Federating Entity as the Identity Provider in charge of fulfilling the request;
- The first test case is related to the following scenario: the client issues an RST with password to the Federating Entity explicitly indicating an external entity as the Identity Provider in charge of fulfilling the request;

A final test case (to be split into more specific test cases when the present specification will be completed) is devoted to handing of request failure

1.2.1.1 ATC-2.1.1 No request designated IdP - Federating entity resolved as IdP

Test case identifier

“urn:ogc:cite:ats:um:0.1.0:07-118r5:authentication-1”

Test assertion [purpose]	The Federating Entity 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: <ul style="list-style-type: none"> • 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 Federating Entity is request designated Id	
Test case identifier	“urn:ogc:cite:ats:um:0.1.0:07-118r5:authentication-2”
Test assertion [purpose]	The Federating Entity is the request designated IdP. In this use case the RST contains an identifier for the Federating Entity STS.
Test method	The client issues an RST with: <ul style="list-style-type: none"> • mandatory username/password information; • an identifier for the Federating 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 6.4.3.1
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	<p>The client issues an RST with:</p> <ul style="list-style-type: none"> • mandatory username/password information; • an identifier for the External Entity. <p>Verify that the client receives a SAML token which is signed and encrypted according to ATC-1.4.</p> <p>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.</p>
--------------------	--

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 Federating Entity 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 Federating entity, 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 RSTs 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	<p>The client issues an RST with:</p> <ul style="list-style-type: none"> • mandatory username information and signature <p>Verify that the client receives a SAML token which is signed and encrypted according to ATC-1.4.</p> <p>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.</p>
--------------------	--

Reference	Clause 6.4.3.4
------------------	----------------

Test type	Capability
1.2.2.2 ATC-2.2.2 unsuccessful 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	<p>The client issues an RST with:</p> <ul style="list-style-type: none"> • mandatory username information and signature <p>Verify that the client receives an error message reporting that the signature is invalid</p> <p>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.</p>
Reference	Clause 6.4.3.4 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 authorisation requests and responses.

Two abstract test cases are defined for authorisation requests, either for synchronous or asynchronous responses. In both test cases, the authorisation 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.

A final test case (to be split into more specific test cases when the present specification will be completed) is devoted to handing of request failure

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]	<p>Only an authorized client can access a requested protected service.</p> <p>The service request header contains a SAML Token returned by a previous successful RST.</p>

Test method	<p>Verify that the service to be invoked is protected, i.e. its WSDL specifies WS-Security policies.</p> <p>The client issues a request containing a SAML token previously obtained through authentication.</p> <p>Verify that the client is authorized to access the protected service, that is a successful response shall be returned.</p>
Reference	Clauses 7.3.2.
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-authorization”
Test assertion [purpose]	<p>Only an authorized client can access a requested protected service.</p> <p>The service request header contains a SAML Token returned by a previous successful RST and WS-Addressing information to allow dispatching of the response.</p>
Test method	<p>Verify that the service to be invoked is protected, i.e. its WSDL specifies WS-Security policies.</p> <p>The client issues a request containing a SAML token, previously obtained through authentication.</p> <p>The Service Provider shall return a service response according to the following format:</p> <ul style="list-style-type: none"> • 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. <p>Pre-condition:</p> <p>The IUT shall support the asynchronous communication for the requested service.</p>
Reference	Clauses 7.3.3
Test type	Capability

1.3.3 ATC-3.3 Authorisation request failure

Test case identifier	“urn:ogc:cite:ats:um:0.1.0:07-118r5:authorization-failure”
-----------------------------	--

Test assertion [purpose]	The Service provider shall return a SOAP fault message if an authorisation 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 authorized to access the protected service. Verify that a SOAP fault response is returned, such that: <ul style="list-style-type: none">• 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>

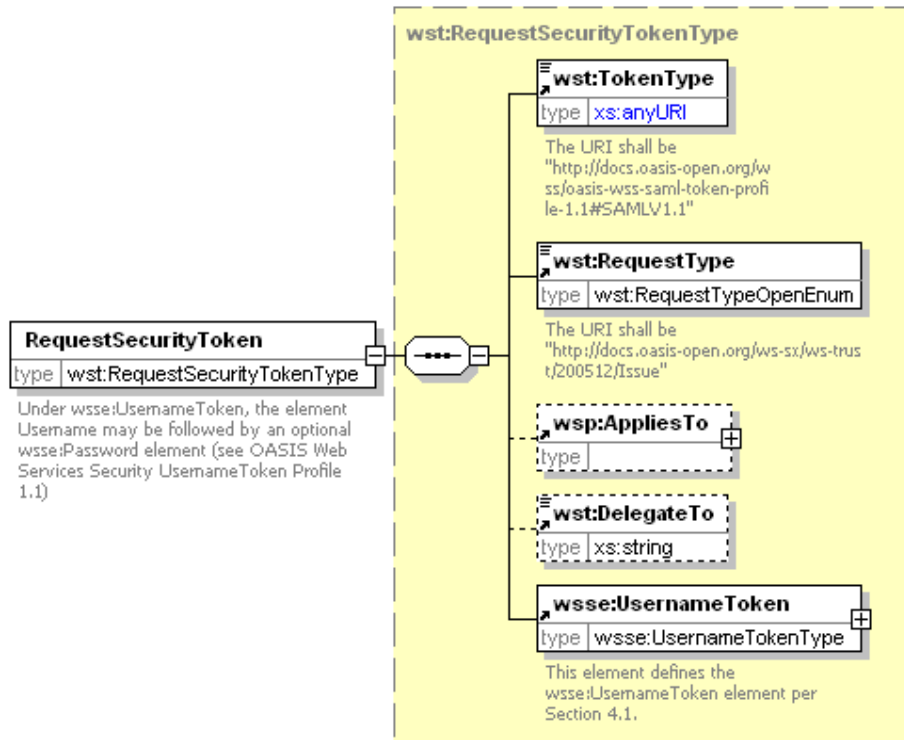
The constrained schema is compatible with the standard WS-Trust 1.3 (i.e. any service implementation conforming to constrained files shall also conform to the standard ones).

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)



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 (§6.4.5.2).

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 shall be ignored for the moment. In the future, it can be used to make the STS more general, by providing SAML tokens that can be used by specified relying party, not only the main federating entity. This is described in the "Extension Points" section (§6.4.5.1).

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 specification.

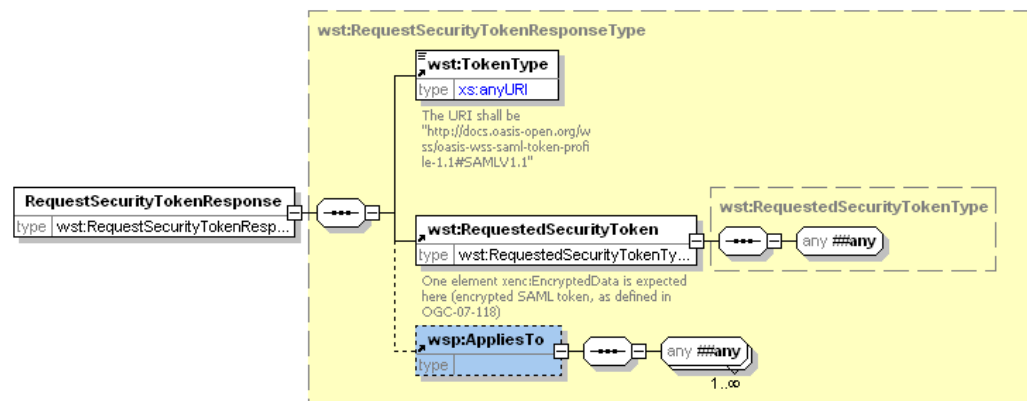
In case of *RST with signature*, it is REQUIRED to put in the SOAP header a wsse:Security element containing a ds:Signature element. The ds:Signature shall contain the digital signature of the SOAP body (that contains the wst:RequestSecurityToken element), as a detached signature. The following

- 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)



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 and shall contain 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 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:complexType>
</xs:schema>
```



```

        <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:RequestTypeEnum">
        <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="RequestTypeEnum">
        <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)

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>
    </Apply>
  </Condition>

```

```

    <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>
  </Rule>
</Policy>

```



```

    </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>
  </Rule>
</Policy>

```

```

</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: ESA UM-SSO / EO-DAIL Integration

UM-SSO deployed at ESA is an operational Single Sign-On system for ESA Web-based Applications which could be adopted by other EO Providers as well. It features typical UM functions (login-authentication, registration-account maintenance, access control). It is based on Shibboleth system ([OR1]). It allows ESA Web applications to outsource the sign-on process and offers the user access to several ESA EO Portals with one single sign-on on his browser (“WEB-SSO”).

The present section provides specific information of usage the present interface in the context of integration of ESA UM-SSO with EO-DAIL.

ESA UM-SSO defines a specific security domain, which is separated from the security domain defined by EO-DAIL, which complies with the present interface. In concrete terms, the UM-SSO IdP authenticates ESA UM-SSO users without providing the SAML token defined by the present interface.

In this specific context, only the RST with signature shall be used (see 6.4.3.4).

In order to establish a trust relationship between the two security domains, a given Client *C* of ESA UM-SSO security domain shall provides its public key to the EO-DAIL STS. The trust relationship between *C* and EO-DAIL IdP is established as soon as the EO-DAIL security administrator has put this public key in the keystore of EO-DAIL IdP. From that point, the client *C* can obtain SAML token for any UM-SSO authenticated users by issuing RST with signature. The sequence of steps is as follows:

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