



NextGRID Security Profile

Editors:	Joris Claessens	EMIC
	Thomas Leonard	IT Innovation
	Mehran Ahsant	KTH

Date	Author	Comments	Version	Status
12 Oct 2006	Joris Claessens	Initial outline	V0.0	Draft
09 Jan 2007	Thomas Leonard	Draft of second part	V0.2	Draft
11 Jan 2007	Joris Claessens	Further input	V0.3	Draft
17 Jan 2007	Joris Claessens	Adding notes	V0.4	Draft
14 Feb 2007	Thomas Leonard	Updated management interface	V0.5	Draft
28 Feb 2007	Thomas Leonard	Added WSDL and description	V0.6	Draft
05 Mar 2007	Thomas Leonard	Added WS-Policy section	V0.7	Draft
08 Mar 2007	Thomas Leonard	Added WS-PolicyAttachment	V0.8	Draft
17 May 2007	Sven v d Berghe	Clean up	V0.9	Draft
05 Jun 2007	Thomas Leonard	Specified container element	V0.10	Draft
03 Jul 2007	Thomas Leonard	Updates from 19 th Jun meeting	V0.11	Draft
23 Aug 2007	Thomas Leonard	Specified SAML token format	V0.12	Draft
13 Sep 2007	Mehran Ahsant	Krb profile NS added	V0.13	Draft
14 Sep 2007	Joris Claessens	Added and fixed references Split-up STS profiles	V0.14	Draft
01-Oct-2007	Thomas Leonard	Added a section on the use of AdditionalSecurityToken	V0.15	Draft
04 Oct 2007	Mehran Ahsant	Added Sign-on STS profile	V0.16	Draft
12 Dec 2007	Joris Claessens	Added section with further	V0.17	Draft

		considerations		
19 Dec 2007	Joris Claessens	Some polishing	V0.18	Draft
02 Jan. 2008	Thomas Leonard	Improved example WSDL	V0.19	Draft
10 Jan. 2008	Thomas Leonard	Comments from review	V0.20	Draft
15 Jan. 2008	Thomas Leonard	Clarify SAML validation	V0.21	Draft
18 Jan 2008	David Snelling	Final Proof	V1.0	Final



1 INTRODUCTION..... 5

1.1 Profile Overview 5

1.2 Relationships to Other Profiles..... 5

1.3 Notational Conventions 6

1.4 Profile Identification and Versioning..... 6

2 PROFILE CONFORMANCE 7

2.1 Conformance Targets 7

2.2 Claiming Conformance 7

3 CONTEXTUALIZED SOAP MESSAGE SECURITY..... 7

3.1 FederationContext header 7

3.2 SAML Assertion..... 8

3.3 Additional tokens in an EPR..... 9

3.4 SOAP Message Security 10

4 CROSS-ORGANIZATIONAL SECURITY MANAGEMENT AND METADATA 11

4.1 FederationContext creation/triggering and content 11

4.2 Policy Management Operations..... 11

4.3 FederationContext-related service metadata 18

5 WS-TRUST TOKEN ISSUANCE AND VALIDATION 19

5.1 Sign-on STS profile..... 20

5.2 Membership STS profile 24

6 INTRA-ORGANIZATIONAL SECURITY MANAGEMENT..... 24

7 FURTHER CONSIDERATIONS 24

7.1 Relation to WS-SecureConversation..... 24

7.2 Relation to WS-SecurityPolicy 25

7.3 Relation to OGSA Security Profile 2.0..... 25



8 REFERENCES..... 25



1 Introduction

This document defines the NextGRID Security Profile 1.0 (hereafter, "the Profile"). This Profile is part of the NextGRID Generalized Specifications which aims to capture NextGRID architectural concepts in a set of composable profiles. These profiles are specified in such a way that they could be implemented in terms of other well known specifications. While overall consistency is achieved at the conceptual level, and captured through the motivating Use Cases accompanying each specification, the implementation in terms of other specifications may not be consistent between different profiles. Thus each profile defines an implementable realisation of the underlying concept, but implementers of the full NextGRID architecture may need to support multiple competing underlying specifications.

Section 1 introduces the Profile, and explains its relationships to other profiles.

Section 2, "Profile Conformance," explains what it means to be conformant to this Profile.

Each subsequent section addresses a component of the Profile, and consists of two parts: an overview detailing the component specifications and their extensibility points, followed by subsections that address individual parts of the component specifications. Note that there is no relationship between the section numbers in this document and those in the referenced specifications.

1.1 Profile Overview

This Profile is intended for use when securing services that are concerned with distributed computing in line with the concepts of NextGRID [1]. A service implementation that uses a security approach in a manner conformant with the Profile may be said to be an "implementation of the NextGRID Security Profile 1.0."

The primary issues addressed in the profile are as follows:

- *Contextualized SOAP Message Security*. This Profile mandates the use of SOAP Message Security [5], suggests the use of the SAML Token Profile [6], and introduces a "FederationContext" SOAP header.
- *Cross-organizational security management and metadata*. This Profile introduces specific cross-organizational security management web services operations, and suggests the use of WS-Policy [14] for exchanging federation/security metadata.
- *Security token issuance and validation*. This Profile describes the use of WS-Trust [9] and WS-Federation [12], and suggests an approach for claims-based security in contextualized federations similar to the TrustCoM Framework V2 [16].

1.2 Relationships to Other Profiles

This Profile is concerned with message-layer security. Transport-layer security issues are covered by the NextGRID Basic Profile [2].

This Profile is intended to be composable with the NextGRID Basic Profile, but does not mandate its full use. The message security approach specified in this Profile is intended to be applicable for both NextGRID Services and ordinary web services.

1.3 Notational Conventions

The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119 [3].

Normative statements of requirements in the Profile are presented in the manner detailed in the WS-I Basic Profile 1.1 Conformance Requirements section [4].

Both requirement statements and extensibility statements can be considered namespace-qualified. This specification uses a number of namespace prefixes throughout; their associated URIs are listed below. Note that the choice of any namespace prefix is arbitrary and not semantically significant.

Table 1 Namespaces used by NextGRID Security Profile 1.0

Prefix	Namespace
soap	http://schemas.xmlsoap.org/soap/envelope
wsdl	http://schemas.xmlsoap.org/wsdl
xsd	http://www.w3.org/2001/XMLSchema
xsi	http://www.w3.org/2001/XMLSchema-instance
wsa	http://www.w3.org/2005/08/addressing
wsi	http://ws-i.org/schemas/conformanceClaim
ngsec	http://nextgrid.org/2007/security
ngpolicy	http://nextgrid.org/2007/security/policy
wsp	http://www.w3.org/ns/ws-policy

1.4 Profile Identification and Versioning

Profile identification and versioning uses the style described in WS-I Basic Profile 1.1 [4] and abides by the normative descriptions contained therein. The name of this Profile is "NextGRID Security Profile" and version number is "1.0."



2 Profile Conformance

Conformance to the Profile is defined normatively in WS-I Basic Profile 1.1 [4]. This Profile abides by those definitions.

2.1 Conformance Targets

The following conformance targets are used in this Profile:

- **ENDPOINTREFERENCE** – the serialization of the `wsa:EndpointReference` element and its content, (from OGSA WSRF Basic Profile).
- **ENVELOPE** – the serialization of the `soap:Envelope` element and its content (from WS-I Basic Profile 1.1).
- **DESCRIPTION** – descriptions of types, messages, interfaces and their concrete protocol and data format bindings, and the network access points associated with Web services (e.g., WSDL descriptions) (from WS-I Basic Profile 1.1) .
- **INSTANCE** – software that implements a `wsdl:port` (from WS-I Basic Profile 1.1, without “bindingTemplate” from the namespace `urn:uddi-org:api_v2`) .
- **CONSUMER** – software that invokes an INSTANCE (from WS-I Basic Profile 1.1).
- **SENDER** – software that generates a particular message according to the protocol(s) associated with that message (from WS-I Basic Profile 1.1).
- **RECEIVER** – software that consumes a message according to the protocol(s) associated with that message (e.g., SOAP processors) (from WS-I Basic Profile 1.1).
- **SAML TOKEN** – the serialisation of a SAML token (from Web Services Security: SAML Token Profile 1.1).

2.2 Claiming Conformance

Claims of conformance to the Profile and the attachments mechanisms are the same as normatively described in WS-I Basic Profile 1.1.

The conformance claim URI for this Profile is <http://www.nextgrid.org/securityprofile/v1>.

3 Contextualized SOAP Message Security

This section of the Profile incorporates the following specifications by reference, and/or defines extensibility points within it:

- OASIS SOAP Message Security [5]
- OASIS SAML Token Profile [6]

3.1 FederationContext header

When NextGRID Services perform a secure message exchange, they can – when already established – explicitly indicate in which federation context/scope the secure message exchange should take place. Such federation context should be referenced or described in a separate FederationContext SOAP header.

R0311 The **ENVELOPE** of a SOAP message MAY contain one or more `<ngsec:FederationContext>` header elements to state the context(s) in which the message is to be processed. Each header is of type `EndpointReferenceType`, as defined in WS-Addressing [13].

R0312 Each `FederationContext` header element in the **ENVELOPE** MUST be included in the message's signature.

Note that a federation context is different from a resource context/identifier. If there is a resource context, this just means a specific resource is being accessed, and not the stateless service; the resource context is only a matter of addressing; both may need a federation context.

The security token attached to the message (see sections 3.2 and 3.3) should contain sufficient attributes showing that the use of the specified federation context is authorized by the requesting and/or receiving domain. The TrustCoM Framework [16] for example proposes to include a federation context as a specific constraint inside tokens issued to services. This Profile does not define any mandatory attributes or constraints.

Section 3.3 shows an example of a SOAP message with a `FederationContext` header including attached security token.

3.2 SAML Assertion

This Profile leverages the SAML assertion format – as used by [6] – for asserting attributes associated with a cryptographic key, which are relevant to the security policy set by the participating organizations in the specified federation context.

R0321 The subject of each **SAML TOKEN** MUST include the X.509 certificate of the subject of the assertion, allowing this to be cross-checked against the certificate used by the subject to sign their SOAP messages.

R0322 For maximum interoperability, the value of each attribute in a **SAML TOKEN** SHOULD be a plain string

R0323 Since the **SAML TOKENs** are forwarded to services via un-trusted clients, they MUST be signed by the issuer of the token.

The attribute name and value are chosen by the issuer and can be arbitrary; whenever a policy is set up to require such a token, the same name and value is then given in the policy rule. A sample SAML token is shown below:

```
<saml:Assertion xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"
  xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  AssertionID="_701188607134c9a97712c51a32d5feb0"
  MajorVersion="1"
  IssueInstant="2007-08-23T10:27:37.749Z"
  MinorVersion="1"
  Issuer="https://example.com/NextGRID/services/MembershipGroup">
```

```

<saml:Conditions NotOnOrAfter="2007-08-23T13:14:17.736Z"/>
<saml:AttributeStatement>
  <saml:Subject>
    <saml:NameIdentifier NameQualifier="-">-</saml:NameIdentifier>
    <saml:SubjectConfirmation>
      <saml:ConfirmationMethod>
        urn:oasis:names:tc:SAML:2.0:cm:holder-of-key
      </saml:ConfirmationMethod>
      <ds:KeyInfo>
        <ds:X509Data>
          <ds:X509Certificate>MIIDLjCCApegAwIBAgIBB...g94Grc=</ds:X509Certificate>
        </ds:X509Data>
      </ds:KeyInfo>
    </saml:SubjectConfirmation>
  </saml:Subject>
  <saml:Attribute AttributeName="member-of-group"
    AttributeNamespace="http://example.com/my-namespace">
    <saml:AttributeValue xsi:type="xsd:string">ff808181-...-0001</saml:AttributeValue>
  </saml:Attribute>
</saml:AttributeStatement>
<ds:Signature>
  <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
    <ds:Reference URI="#_701188607134c9a97712c51a32d5feb0">
      <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#">
          <ec:InclusiveNamespaces
            PrefixList="code ds kind rw saml samlp typens #default xsd xsi" />
        </ds:Transform>
      </ds:Transforms>
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
      <ds:DigestValue>O21fi4M6DsFED39HirIi8fm6yi0=</ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>hOwEQp...0auiJM=</ds:SignatureValue>
</ds:Signature>
</saml:Assertion>

```

3.3 Additional tokens in an EPR

When a SENDER references some resource using an EPR, they may need to provide additional proof that they are permitted to use that resource/EPR. For example, if a CONSUMER is invoking an INSTANCE in the context of an SLA resource, and the SLA is available only to members of a particular group, the CONSUMER can include in the EPR a SAML token asserting that they are a member of the required group. When the INSTANCE checks the user's access to the resource in the EPR, it will pass along these extra required tokens to the validation service for that EPR.

The CONSUMER attaches such tokens to the EPR rather than placing them in the WS-Security header because there may be many EPRs (including federation contexts) in a message, and the service INSTANCE should not pass all of the extra tokens to each validation service.

Note that the tokens passed in this way may be the same additional tokens that the CONSUMER would have included in the WS-Security header if they were invoking operations on the federation resource directly.

R0331 An *ENDPOINTREFERENCE* MAY contain extra security tokens in its Metadata section. Each extra token is placed in an `<ngsec:AdditionalSecurityToken>` element. When checking whether the original SENDER is permitted to use the EPR, these extra tokens SHOULD be taken into consideration.

It is up to the validator to establish that the tokens can be trusted. For example, if the SENDER includes a SAML assertion into the EPR, then the validator MUST check that the subject (X.509 certificate) of the SAML assertion is the same as the sender of the message, and the validator MUST check that it trusts the issuer of the assertion, in order to deduce that the sender of the message has the specified attribute. To allow this, the service INSTANCE MUST pass the SENDER's X.509 certificate to the validator, along with any extra tokens.

The sample SOAP message outline below shows an example of the use of this element. The sender of the message wishes to invoke the "createSampleResource" operation on the target, in the context of an SLA. The SLA may be used by members of "Project X", and the client includes a SAML assertion (in the format described above) proving that they are in this project:

```
<soapenv:Envelope>
  <soapenv:Header>
    <ngsec:FederationContext>
      <wsa:Address>http://sla.service/sla</wsa:Address>
      <wsa:Metadata>
        <ngsec:AdditionalSecurityToken>member of project X</ns1:AdditionalSecurityToken>
      </wsa:Metadata>
    </ngsec:FederationContext>
  </soapenv:Header>
  <soapenv:Body>
    <impl:createSampleResource/>
  </soapenv:Body>
</soapenv:Envelope>
```

3.4 SOAP Message Security

This Profile mandates the use of the SOAP Message Security specification [5] for securing the messages across NextGRID Services.

Any FederationContext headers (Sect. 3.1) MUST be included in the SOAP message signature.

This Profile does not mandate encryption at the message layer. Confidentiality MAY be handled at the transport layer as defined by the NextGRID Basic Profile.

This Profile recognizes that various alternative Token Profiles can be adopted, and the following options are suggested:

- *X.509 Certificate Token Profile* [7]. The SOAP message is signed using a normal X.509 certificate, and a SAML token is separately added to the signed message (i.e. inside the FederationContext); the Subject of the SAML token MUST refer to the X.509 certificate. This is the approach taken in Sect. 3.3 where the SENDER includes an additional SAML assertion into the FederationContext EPR.
- *SAML Token Profile* [6]. The SOAP message is signed directly with a SAML token (which contains appropriate key material), and no other tokens are required.

4 Cross-organizational security management and metadata

This section of the Profile incorporates the following specifications by reference, and defines extensibility points within them:

- WS-Trust 1.3 [9]
- WS-Federation 1.1 [12]
- WS-Policy (September 2004) [14]
- WS-PolicyAttachment (25 April 2006) [15]
- TrustCoM Framework V2 [16]

NextGRID uses a security model in which tokens are issued by a WS-Trust STS in the consumer domain, and then validated using another WS-Trust STS in the service provider domain, as described in WS-Federation (see figure 9 of that specification).

4.1 FederationContext creation/triggering and content

A federation can be represented by a resource, which is used to manage the federation.

This Profile does not define how this resource is created. It is expected that different types of federation management services will have their own operations for this, taking arguments specific to the type of federation being created. Federation resources may also be created out-of-band, rather than using SOAP operations.

The result of creating a federation is a FederationContext EPR. Membership of the federation MAY be partially managed using the policy management operations described below.

4.2 Policy Management Operations

Service providers may grant clients some ability to control who has access to the resources clients have established with the service providers. This Profile specifies one possible approach where service providers expose a cross-organizational management interface, allowing clients to map security tokens that clients may present to “access rights” or other roles and attributes defined by the service provider, that these clients should be given.

This specification defines a port type for basic policy management, modelled on access control lists. It is desirable to keep the format simple to reduce the implementation burden, to have a

model that is easy to understand, and to reduce the chance of denial-of-service attacks (where a client specifies an extremely complex policy).

More complex policies can be supported by defining a resource's policy as requiring a token from some STS, and then implementing whatever complex policy is required there. This also allows clients to keep the details of their policies private, or to share a single policy between many resources.

The WSDL for the interface is given in full below. Informally, the policy management operations are:

- String[] getValidRoles()
- void addPolicyRule(PolicyRule)
- void removePolicyRule(PolicyRule)
- PolicyRule[] getPolicyRules()

These operations are used to get the list of roles supported by the service, to add a new rule, to remove an existing rule, or to fetch the current list of rules, respectively.

Note that, as always, an instance is free to implement any security policy necessary to control access to these policy-mapping operations. This access may also be controlled by the policy: the policy may allow one client to update the policy to allow another client to manage the policy, for example.

R0421 An **INSTANCE** MAY filter the roles and policies returned to the **CONSUMER**, so that the **CONSUMER** sees only a subset of the complete policy and role set.

Note that no operation is provided to replace a complete policy in one go. This is because there may be several requestors modifying the policy at the same time, and a process where each one downloads the policy, modifies it, and then uploads the new policy would suffer from race conditions.

PolicyRules

Since the tokens are issued and validated in different domains, interoperability requires a set of supported token types. This profile defines two token types that must be supported. Additional types MAY be supported but clients cannot assume this.

R0422 An **INSTANCE** MUST support mappings using X.509 v3 tokens.

R0423 An **INSTANCE** MUST support mappings using SAML tokens.

This extract from the WSDL shows the schema for a PolicyRule:

```
<complexType name="PolicyRule">
  <sequence>
    <element name="matchPattern" type="ngsec:MatchPattern"/>
    <element name="type" type="ngsec:PolicyRuleType"/>
    <element name="role" type="xsd:string"/>
  </sequence>
</complexType>
```

```
</sequence>
</complexType>
```

The `matchPattern` element determines whether a given subject is affected by the rule or not. When a subject attempts to use a resource, any rules not matching the subject's credentials are ignored.

Match patterns

A match pattern can be applied to a set of credentials presented by a subject to determine whether the subject matches the pattern or not. The schema defines the `MatchPattern` type as:

```
<complexType name="MatchPattern">
  <sequence>
    <element maxOccurs="1" minOccurs="0" name="issuerCertificate" type="xsd:string"/>
    <element maxOccurs="1" minOccurs="0" name="subjectDN" type="xsd:string"/>
    <element maxOccurs="1" minOccurs="0" name="attributeName" type="xsd:string"/>
    <element maxOccurs="1" minOccurs="0" name="attributeValue" type="xsd:string"/>
  </sequence>
</complexType>
```

The `subjectDN` element gives the X.509 *Distinguished Name* which a subject must have in their X.509 certificate to match this pattern

The `attributeName` and `attributeValue` elements give a SAML attribute which a subject must have in a SAML assertion. If one of these two fields is present then the other must be as well.

The `subjectDN` field must be present if and only if the attribute fields are not present.

The `issuerCertificate` is the Base64-encoded form of an X.509 certificate. This identifies the authority authorised to make the claim that the pattern checks. The public key found in this certificate must be the one used to sign the user's credential (X.509 certificate or SAML token) for it to match.

If both the `issuerCertificate` and `subjectDN` fields are the special string "*" then this rule matches all subjects.

Roles

Each type of resource has a set of roles which a subject may have. Each policy rule affects membership of only a single role, given by the role element. Unlike the tokens matched by the match patterns, which are chosen by and meaningful only to the client, roles must be meaningful to both client and service.

The set of available roles for a resource is defined by the instance.

Rule types

Three types of rule are defined in the schema:

```
<simpleType name="PolicyRuleType">
```

```

<restriction base="xsd:string">
  <enumeration value="NECESSARY"/>
  <enumeration value="SUFFICIENT"/>
  <enumeration value="DENY"/>
</restriction>
</simpleType>

```

Determining whether a subject has a role

A simple algorithm for determining membership of a role is as follows:

1. List all of the policy's rules for the desired role.
2. The subject has the role if and only if all the following are true:
 - a. It matches at least one rule of type "SUFFICIENT".
 - b. It matches all rules of type "NECESSARY".
 - c. It matches no rules of type "DENY".

Example

As an example, this SOAP fragment shows a request message to a resource, requesting the resource to add a rule. In the absence of any other DENY or NECESSARY rules for the resource, this rule will grant anyone the *reader* role on the resource if they have a signed SAML assertion with themselves as the subject, asserting that they have an attribute called 'member-of-project' with a value of 'NextGRID'. The SAML assertion must be signed by the private key corresponding to the public key extracted from the certificate in the *issuerCertificate* element.

```

<addPolicyRule xmlns="http://nextgrid.org/2007/security">
  <rule>
    <matchPattern>
<issuerCertificate>MIIDaTCCAtKgAwIBAgIBATANBgkqhkiG9w0BAQQFADBxMRYwFAYDVQQDEw1BcnRlbWlzlIENBIEl
...
K3e/kCjY9zHpGXCgTC6S6fTiQ60wnhj6hPcgxkq8qqhfH15Zi9++P+E6/FvWQ==</issuerCertificate>
      <attributeName>member-of-project</attributeName>
      <attributeValue>NextGRID</attributeValue>
    </matchPattern>
    <type>SUFFICIENT</type>
    <role>reader</role>
  </rule>
</addPolicyRule>

```

WSDL

The full WSDL for the PolicyManagement interface is:

```

<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://nextgrid.org/2007/security"
  xmlns:ngsec="http://nextgrid.org/2007/security"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"

```

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<wsdl:types>
  <schema elementFormDefault="qualified"
    targetNamespace="http://nextgrid.org/2007/security"
    xmlns="http://www.w3.org/2001/XMLSchema">
    <element name="addPolicyRule">
      <complexType>
        <sequence>
          <element name="rule" type="ngsec:PolicyRule"/>
        </sequence>
      </complexType>
    </element>
    <complexType name="MatchPattern">
      <sequence>
        <element maxOccurs="1" minOccurs="0" name="issuerCertificate" type="xsd:string"/>
        <element maxOccurs="1" minOccurs="0" name="subjectDN" type="xsd:string"/>
        <element maxOccurs="1" minOccurs="0" name="attributeName" type="xsd:string"/>
        <element maxOccurs="1" minOccurs="0" name="attributeValue" type="xsd:string"/>
      </sequence>
    </complexType>
    <simpleType name="PolicyRuleType">
      <restriction base="xsd:string">
        <enumeration value="NECESSARY"/>
        <enumeration value="SUFFICIENT"/>
        <enumeration value="DENY"/>
      </restriction>
    </simpleType>
    <complexType name="PolicyRule">
      <sequence>
        <element name="matchPattern" type="ngsec:MatchPattern"/>
        <element name="type" type="ngsec:PolicyRuleType"/>
        <element name="role" type="xsd:string"/>
      </sequence>
    </complexType>
    <element name="addPolicyRuleResponse">
      <complexType/>
    </element>
    <element name="removePolicyRule">
      <complexType>
        <sequence>
          <element name="rule" type="ngsec:PolicyRule"/>
        </sequence>
      </complexType>
    </element>
    <element name="removePolicyRuleResponse">
      <complexType/>
    </element>
    <element name="getPolicyRules">
      <complexType/>
    </element>
  </schema>
</wsdl:types>
```

```

<element name="getPolicyRulesResponse">
  <complexType>
    <sequence>
      <element maxOccurs="unbounded" name="getPolicyRulesReturn" type="ngsec:PolicyRule"/>
    </sequence>
  </complexType>
</element>
<element name="getValidRoles">
  <complexType/>
</element>
<element name="getValidRolesResponse">
  <complexType>
    <sequence>
      <element maxOccurs="unbounded" name="getValidRolesReturn" type="xsd:string"/>
    </sequence>
  </complexType>
</element>
</schema>
</wsdl:types>

<wsdl:message name="getValidRolesResponse">
  <wsdl:part element="ngsec:getValidRolesResponse" name="parameters"/>
</wsdl:message>
<wsdl:message name="addPolicyRuleRequest">
  <wsdl:part element="ngsec:addPolicyRule" name="parameters"/>
</wsdl:message>
<wsdl:message name="getValidRolesRequest">
  <wsdl:part element="ngsec:getValidRoles" name="parameters"/>
</wsdl:message>
<wsdl:message name="removePolicyRuleRequest">
  <wsdl:part element="ngsec:removePolicyRule" name="parameters"/>
</wsdl:message>
<wsdl:message name="getPolicyRulesRequest">
  <wsdl:part element="ngsec:getPolicyRules" name="parameters"/>
</wsdl:message>
<wsdl:message name="addPolicyRuleResponse">
  <wsdl:part element="ngsec:addPolicyRuleResponse" name="parameters"/>
</wsdl:message>
<wsdl:message name="getPolicyRulesResponse">
  <wsdl:part element="ngsec:getPolicyRulesResponse" name="parameters"/>
</wsdl:message>
<wsdl:message name="removePolicyRuleResponse">
  <wsdl:part element="ngsec:removePolicyRuleResponse" name="parameters"/>
</wsdl:message>

<wsdl:portType name="PolicyManagement">
  <wsdl:operation name="addPolicyRule">
    <wsdl:input message="ngsec:addPolicyRuleRequest" name="addPolicyRuleRequest"/>
    <wsdl:output message="ngsec:addPolicyRuleResponse" name="addPolicyRuleResponse"/>
  </wsdl:operation>
  <wsdl:operation name="removePolicyRule">

```

```
<wsdl:input message="ngsec:removePolicyRuleRequest" name="removePolicyRuleRequest"/>
<wsdl:output message="ngsec:removePolicyRuleResponse" name="removePolicyRuleResponse"/>
</wsdl:operation>
<wsdl:operation name="getPolicyRules">
  <wsdl:input message="ngsec:getPolicyRulesRequest" name="getPolicyRulesRequest"/>
  <wsdl:output message="ngsec:getPolicyRulesResponse" name="getPolicyRulesResponse"/>
</wsdl:operation>
<wsdl:operation name="getValidRoles">
  <wsdl:input message="ngsec:getValidRolesRequest" name="getValidRolesRequest"/>
  <wsdl:output message="ngsec:getValidRolesResponse" name="getValidRolesResponse"/>
</wsdl:operation>
</wsdl:portType>

<wsdl:binding name="PolicyManagementSoapBinding" type="ngsec:PolicyManagement">
  <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="addPolicyRule">
    <wsdlsoap:operation soapAction="http://nextgrid.org/2007/security/addPolicyRule"/>
    <wsdl:input name="addPolicyRuleRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="addPolicyRuleResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="removePolicyRule">
    <wsdlsoap:operation soapAction="http://nextgrid.org/2007/security/removePolicyRule"/>
    <wsdl:input name="removePolicyRuleRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="removePolicyRuleResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="getPolicyRules">
    <wsdlsoap:operation soapAction="http://nextgrid.org/2007/security/getPolicyRules"/>
    <wsdl:input name="getPolicyRulesRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="getPolicyRulesResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="getValidRoles">
    <wsdlsoap:operation soapAction="http://nextgrid.org/2007/security/getValidRoles"/>
    <wsdl:input name="getValidRolesRequest">
      <wsdlsoap:body use="literal"/>
    </wsdl:input>
    <wsdl:output name="getValidRolesResponse">
      <wsdlsoap:body use="literal"/>
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```

```
</wsdl:binding>
</wsdl:definitions>
```

4.3 FederationContext-related service metadata

In order to invoke an INSTANCE, a CONSUMER needs to know whether they must include a FederationContext SOAP header, and what this header should contain if so. This information is attached to the service's WSDL document using WS-PolicyAttachment. A service's WSDL document is retrieved as specified in the NextGRID Basic Profile [2].

R0431 *If an **INSTANCE** requires a FederationContext to be passed by the CONSUMER when invoking certain operations, then it **MUST** specify this in the resource's WSDL, using WS-PolicyAttachment.*

An example WSDL document is reproduced below. This states that the example getResources SOAP operation can be invoked without any federation context, while the example createSampleResource operation requires either an "account" federation context or an "SLA" federation context EPR, from the specified service.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsp="http://www.w3.org/ns/ws-policy"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
  1.0.xsd"
  xmlns:ngpolicy ="http://nextgrid.org/2007/security/policy"
  ...>
  [...]
  <wsdl:portType name="SampleService">

    <wsdl:operation name="createSampleResource">
      <wsdl:input message="impl:createSampleResourceRequest"
        name="createSampleResourceRequest"/>
      <wsdl:output message="impl:createSampleResourceResponse"
        name="createSampleResourceResponse"/>
      <wsp:PolicyReference URI="#BillingHeaders" wsdl:required="true"/>
    </wsdl:operation>

    <wsdl:operation name="getResources">
      <wsdl:input message="impl:getResourcesRequest" name="getResourcesRequest"/>
      <wsdl:output message="impl:getResourcesResponse" name="getResourcesResponse"/>
    </wsdl:operation>

  </wsdl:portType>

  <wsp:Policy wsu:Id="BillingHeaders">
    <wsp:ExactlyOne>
      <ngpolicy:FederationContext EPR-address="https://example.com/TradeAccountService"/>
      <ngpolicy:FederationContext EPR-address="https://example.com/SLAService"/>
    </wsp:ExactlyOne>
  </wsp:Policy>
```

```
</wsdl:definitions>
```

This specification defines one new WS-Policy primitive assertion:

- `<ngpolicy:FederationContext EPR-address='...'/>`

A SOAP request message satisfies this assertion if there is a federation context SOAP header EPR whose Address field is the one in the assertion.

The schema for this element is as follows:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://nextgrid.org/2007/security/policy"
  elementFormDefault="qualified">
  <xs:element name="FederationContext">
    <xs:complexType>
      <xs:attribute name="EPR-address" use="required" type="xs:anyURI"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

5 WS-Trust token issuance and validation

A consumer may need additional tokens to prove that it is permitted to use a resource or FederationContext. These tokens are attached to the SOAP message as described above. There are various reasons why a client may include a token in a SOAP message, including:

1. The client is given an identity or group-membership token when they log in to their local system. They include this in all messages sent.
2. The client explicitly gets a token asserting that they have some attribute (for example, that they are a member of some project). They include this in all messages sent.
3. The client knows that a particular token is needed to use a particular resource or FederationContext. The client may be given the token, or told where to get it, when they discover the FederationContext or resource.
4. The client is told by an instance to get a token from a particular STS (in the instance's WS-Policy document). The client includes this token in messages sent to this service.

In all cases, the consumer gets any required token from a WS-Trust STS, using the WS-Trust issuance binding.

R0501 The **ENVELOPE** for token issuance **MUST** conform to the WS-Trust Issuance Binding.

R0502 The **ENVELOPE** for token validation **MUST** conform to the WS-Trust Validation Binding.

The token contents are typically opaque to the client. The token must be attached to the message as specified in Sect. 3.4.

R0504 *A RECEIVER MUST ignore any unrecognised security tokens attached to the message.*

Clients may use multiple Security Token Services within their own domain. For example, a client might use one STS to get an X.509 identity token from a Kerberos ticket and then use a membership service to get a SAML token asserting they are a member of some group. WS-Trust must also be used for this (R0501).

Finally, a service provider may require clients to get a token from an STS of the service provider's choosing. In this scenario, the STS to use is public knowledge, and is given in the WS-Policy.

R0505 *If an INSTANCE requires a CONSUMER to get present a token from an STS, then it SHOULD specify this in the resource's WS-Policy.*

5.1 Sign-on STS profile

A Sign-on STS is a kind of WS-Trust Security Token Service that issues security tokens as defined by the WS-Trust specification. It can issue users, who already hold a Kerberos ticket asserting their identity, with an X.509 certificate. The SOAP messaging for issuing tokens between users and a Sign-on STS is fully compliant with the WS-Trust specification and more specifically with the “issuance binding” of this specification. The rest of this section describes a request/response message pair that should be exchanged between a Requester and a Sign-on STS in order to issue an X509 certificate given a valid Kerberos ticket.

Request Message:

The requester makes a WS-Trust RST message (“RequestSecurityToken”) and includes a <RequestType> element which is populated with an appropriate value proposed by the WS-Trust “Issuance Binding”. It also includes a <TokenType> element to specify the type of requested security token.

R0506 *When requesting a security token in format of X509CERTIFICATE, the ENVELOPE MUST specify the <TokenType> as ["http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"](http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3).*

The RST message SHOULD also attach a PKCS#10 Certificate Request as a “claim” of request according to what is proposed by the WS-Trust specification.

R0508 *When requesting a security token based on a claim in format of X509 CERTIFICATE REQUEST, the ENVELOPE MUST specify the claim type as ["http://nextgrid.org/2007-security-profile-x509-token-profile#PKCS10"](http://nextgrid.org/2007-security-profile-x509-token-profile#PKCS10).*

The Requester SHOULD also include a Kerberos Service Token (ST) as a proof-of-possession for the RST message. This should be attached as the second “claim” of the request message in order to prove that the holders is a valid and authenticated user in her local domain.

R0507 *When requesting a security token based on a claim in format of Kerberos Tokens, the ENVELOPE MUST specify the claim type as “http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ”.*

Response Message:

A Sign-on STS generates a WS-Trust RSTR message (“RequestSecurityTokenResponse”) and attaches the requested security token (an X509 Certificate) to it according to the WS-Trust specification.

R0506 *When responding a security token in format of X509CERTIFICATE, the ENVELOPE MUST specify the <TokenType> as “<http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3>”.*

Security consideration:

The RST message which is sent from a Requester to a Sign-on STS SHOULD be signed and encrypted using the shared secret in a Kerberos Service Token according to the WS-Security specification. A Sign-on STS SHOULD also use its private key to sign and encrypt the RSTR message when sending a requested security token to A Requester.

Sample SOAP messages:

A sample request SOAP message:

```
<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">
  <soapenv:Header>
    <wsse:Security
      soapenv:mustUnderstand="1"
      xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <wsse:BinarySecurityToken
        EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
        ValueType="http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ"
        wsu:Id="KrbID-6885751"
        xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
        YYH8MIH.....Z4yUYmcWl0r/8Wsttn
      </wsse:BinarySecurityToken>
```

```

<ds:Signature Id="Signature-5076660"
  xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
    <ds:Reference URI="#id-28623319">
      <ds:Transforms>
        <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>qy+xc3zmJXg/uD5W4Yu8x/xKL0w=</ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>P7cg0LmLDb85oBdf1f9dRZEFxUs=</ds:SignatureValue>
  <ds:KeyInfo Id="KeyId-9144903">
    <wsse:SecurityTokenReference
      wsu:Id="STRId-12470752"
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility1.0.xsd">
      <wsse:Reference
        URI="#KrbID-6885751"
        ValueType="http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-
profile-1.1#GSS_Kerberosv5_AP_REQ"/>
    </wsse:SecurityTokenReference>
  </ds:KeyInfo>
</ds:Signature>
</wsse:Security>
</soapenv:Header>
<soapenv:Body
  wsu:Id="id-28623319"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd">
  <wst:RequestSecurityToken xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
    <wst:RequestType>http://schemas.xmlsoap.org/ws/2005/02/trust/Issue</wst:RequestType>
    <wst:TokenType>
      http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0
    </wst:TokenType>
    <wst:Claims Dialect="http://DialectNameSpace">
      <wsse:BinarySecurityToken
        EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soap-message-security-1.0#Base64Binary"
        ValueType="http://nextgrid.org/2007-security-profile-x509-token-
profile#PKCS10"
        xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">
          MIIBPD.....ny3Rg==
        </wsse:BinarySecurityToken>
      <wsse:BinarySecurityToken
        EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soap-message-security-1.0#Base64Binary"
        ValueType="http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-

```

```

profile-1.1#GSS_Kerberosv5_AP_REQ"
  xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">
    YYH8MIH ...8+g4xw9y9jxQnRFJYjPs7hGpK2ryNSAehpGuEaSqRr10RYFX
    9tD1/5vA57
  </wss:BinarySecurityToken>
</wst:Claims>
</wst:RequestSecurityToken>
</soapenv:Body>
</soapenv:Envelope>

```

A sample response SOAP message:

```

<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">
  <soapenv:Header>
    <wss:Security
      soapenv:mustUnderstand="1"
      xmlns:wss="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
1.0.xsd">
      <ds:Signature
        Id="Signature-6059828"
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
          <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#hmac-sha1" />
          <ds:Reference URI="#id-16765237">
            <ds:Transforms>
              <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>M5+nbOa1ZkVlFr3P1CLB5/NlJGw=</ds:DigestValue>
          </ds:Reference>
          <ds:Reference URI="#SigConf-27785692">
            <ds:Transforms>
              <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>lpP2608mk4yNTw8AnydxvYpTXAc=</ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>m8/h0LEoGwyHsZ6G+ZpmJMv8TPk=</ds:SignatureValue>
      </ds:Signature>
      <wsse11:SignatureConfirmation
        Value="P7cg0LmLDb85oBdf1f9dRZEFxUs="
        wsu:Id="SigConf-27785692"
        xmlns:wsse11="http://docs.oasis-open.org/wss/2005/xx/oasis-2005xx-wss-
wssecurity-secext-
1.1.xsd"

```

```

    xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-          1.0.xsd"/>
  </wsse:Security>
</soapenv:Header>
<soapenv:Body
  wsu:Id="id-16765237"
  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-
1.0.xsd">
  <wst:RequestSecurityTokenResponse
    xmlns:wst="http://schemas.xmlsoap.org/ws/2005/02/trust">
    <wst:TokenType>
      http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0
    </wst:TokenType>
    <wst:RequestedSecurityToken>
    <wsse:BinarySecurityToken
      EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
soap-message-security-1.0#Base64Binary"
      ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-
token-profile-1.0#X509v3"
      xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">
        MIIB6DC.....iq7U
        MB7shK7b
      </wsse:BinarySecurityToken>
    </wst:RequestedSecurityToken>
  </wst:RequestSecurityTokenResponse>
</soapenv:Body>
</soapenv:Envelope>

```

5.2 Membership STS profile

R0508 When requesting or validating a SAML v1.1 token, the **ENVELOPE MUST** specify the token type as "<http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1>" (from TrustCoM)

6 Intra-organizational security management

How clients and service providers manage their internal security policies is out of scope of this profile, although they may choose to use the inter-domain operations specified above for all or part of this.

7 Further considerations

7.1 Relation to WS-SecureConversation

The WS-SecureConversation specification [10] defines “mechanisms for establishing and sharing security contexts”. “The security context is defined as a new WS-Security token type that is obtained using a binding of WS-Trust.” “Parties that wish to exchange multiple messages typically

establish a security context in which to exchange multiple messages. A security context is shared among the communicating parties for the lifetime of a communications session.” This Profile does not cover any issues regarding composability with WS-SecureConversation. However, this Profile does not prevent, and even recommends e.g. for performance reasons, to establish a lightweight security context to be able to authenticate and refer to an initially authenticated FederationContext, in a sequence of service requests that are to be performed in the scope of that same FederationContext.

7.2 Relation to WS-SecurityPolicy

The WS-SecurityPolicy specification [11] defines “a set of security policy assertions for use with the WS-Policy framework with respect to security features provided in SOAP Message Security”, and more particularly defines “a base set of assertions that describe how messages are to be secured”. The use of SOAP Message Security as prescribed by this Profile in Sect. 3.4, could therefore in principle be expressed by such a security policy assertion. For example, when the FederationContext header must be covered by the message signature, then this header should be included in the <sp:SignedParts> elements.

Note that this Profile defines a new WS-Policy primitive assertion to indicate that a FederationContext header is required (see Sect. 4.3). This is in line with the WS-SecurityPolicy specification which notes that “specifications are expected to provide domain specific assertions that specify which headers are expected in a message”. If this Profile had not specified such a new assertion, then the required presence of a FederationContext header would have needed to be indicated as part of the <sp:RequiredElements>.

7.3 Relation to OGSA Security Profile 2.0

While the OGSA Security Profile 1.0 mainly covers transport layer security, the OGSA WG is working towards a OGSA Security Profile 2.0, including a profile of SOAP message layer security within the OGSA context [17]. The fundamental principles of this profile are fully aligned with those in this document, namely the recommendation to protect integrity and confidentiality of messages, particularly when there are messaging intermediaries. The Profile in this document adds specific concepts which are not covered by the OGSA security profile, namely the requirement of a FederationContext header, and the support for policy management operations across trust boundaries.

8 References

- [1] NextGRID White Paper
http://www.nextgrid.org/download/publications/NextGRID_Architecture_White_Paper.pdf.
- [2] NextGRID Basic Profile www.nextgrid.org/GS/management/NextGRID_Basic_Profile/
- [3] S. Bradner (ed.): Key words for use in RFCs to Indicate Requirement Levels, The Internet Engineering Task Force Best Current Practice, March 1997.
<http://www.ietf.org/rfc/rfc2119>
- [4] K. Ballinger, D. Ehnebuske, C. Ferris, M. Gudgin, C.K. Liu, M. Nottingham, and P. Yendluri (ed.): Basic Profile Version 1.1, Web Services Interoperability Organization Final Material, 24 August 2004. <http://www.ws-i.org/Profiles/BasicProfile-1.1.html>

- [5] IBM, Microsoft, Sun, VeriSign (ed.): Web Services Security: SOAP Message Security 1.1 (WS-Security 2004), OASIS Standard, 1 February 2006. <http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>
- [6] IBM, Microsoft, Sun, VeriSign (ed.): Web Services Security: SAML Token Profile 1.1, OASIS Standard, 1 February 2006. <http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLSecurityProfile.pdf>
- [7] IBM, Microsoft, Sun, VeriSign (ed.): Web Services Security: X.509 Certificate Token Profile 1.1, OASIS Standard, 1 February 2006. <http://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf>
- [8] IBM, Microsoft, Sun, VeriSign (ed.): Web Services Security: Kerberos Token Profile 1.1, OASIS Standard, 1 February 2006. <http://www.oasis-open.org/committees/download.php/16788/wss-v1.1-spec-os-KerberosTokenProfile.pdf>
- [9] IBM, Microsoft, Nortel, VeriSign (ed.): WS-Trust 1.3. OASIS Standard, March 2007. <http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-os.pdf>
- [10] IBM, Microsoft, Nortel, VeriSign (ed.): WS-SecureConversation 1.3. OASIS Standard, March 2007. <http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.pdf>
- [11] IBM, Microsoft, Nortel, VeriSign (ed.): WS-SecurityPolicy 1.2. OASIS Standard, July 2007. <http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.pdf>
- [12] BEA, BMC Software, CA Inc., IBM, Layer 7 Technologies Inc, Microsoft, Novell Inc., VeriSign: Web Services Federation Language (WS-Federation). Version 1.1, December 2006. <http://specs.xmlsoap.org/ws/2006/12/federation/ws-federation.pdf> (WS-Federation 1.2 is drafted by OASIS WSFED TC. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsfed)
- [13] Microsoft, Sun, Computer Associates International: Web Services Addressing 1.0 - Core. W3C Recommendation 9 May 2006. <http://www.w3.org/TR/ws-addr-core>
- [14] BEA Systems, IBM, Layer 7 Technologies, Microsoft, Nokia, SAP, webMethods (ed.): Web Services Policy 1.5 – Framework (WS-Policy). W3C Recommendation, 4 September 2007. <http://www.w3.org/TR/ws-policy/>
- [15] BEA Systems, IBM, Layer 7 Technologies, Microsoft, Nokia, SAP, webMethods (ed.): Web Services Policy 1.5 – Attachment (WS-PolicyAttachment). W3C Recommendation, 4 September 2007. <http://www.w3.org/TR/ws-policy-attach/>
- [16] Michael D. Wilson, Alvaro Arenas, Lutz Schubert (ed.): The TrustCoM Framework for Trust, Security and Contract management, V2. TrustCoM Deliverable D29/D35/D36, January 2006. (Section IV.2, pages 84 and following) <http://www.eu-trustcom.com/DownDocumentation.php?tipo=docu&id=243>
- [17] Duane Merrill (ed.): OGSA Security Profile 2.0 – Secure SOAP Messaging. OGF OGSA WG Draft Recommendation. <https://forge.gridforum.org/sf/go/doc14551>