

Using XACML and SAML for Authorisation messaging and assertions:

XACML and SAML standards overview and usage examples

Draft version 0.1. - January 12, 2005

Yuri Demchenko <demch@science.uva.nl>

Abstracts

This document provides general overview of the XACML and SAML standards features that can be used for authorisation decision request and authorisation assertion expression. Example of custom AuthzTicket used in combined pull-push authorisation model is described and its mapping to both XACML Request/Response messages format and SAML Authorisation assertion is discussed.

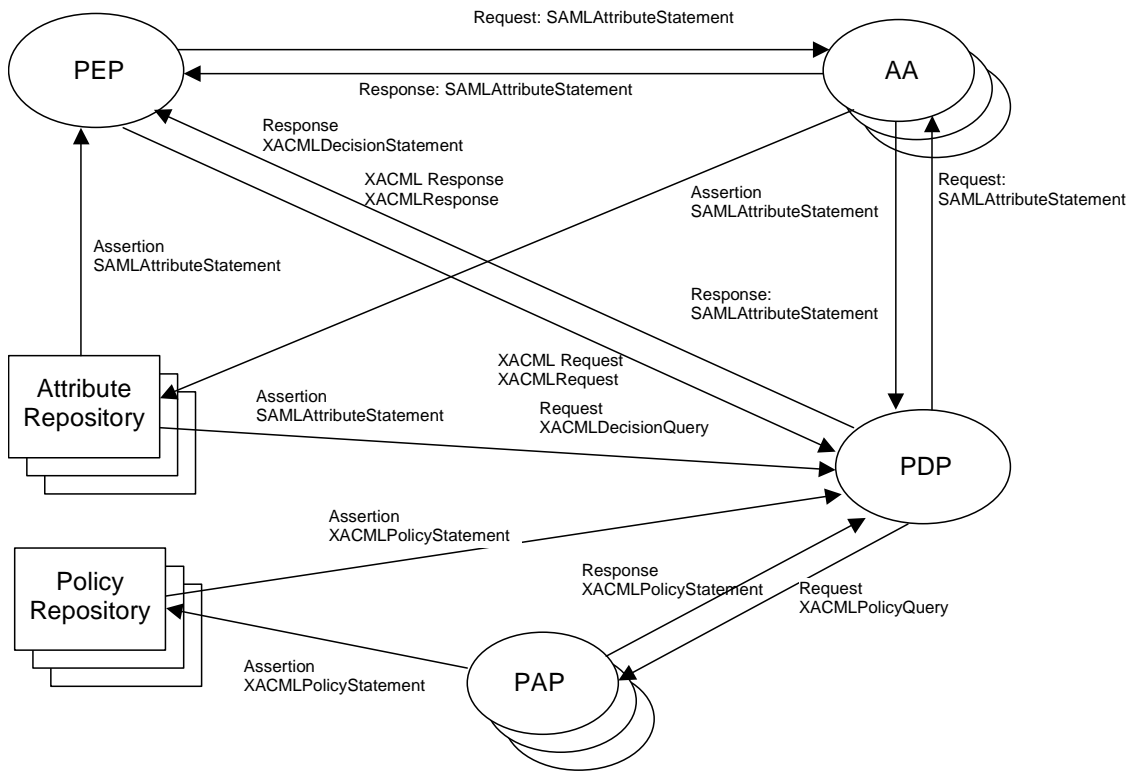
The document provides also in-deep analysis and comparison between SAML 1.1 and SAML 2.0 functionality for authorisation. Current version of the document provides general overview of XACML 2.0 core specification and short overview of the XACML 2.0 special profiles for RBAC and multiple and hierarchical resources. The latter will be extended in further versions.

1 Using XACML and SAML for AuthZ decision request and security tokens exchange	1
2 Example mapping between custom/OCE AuthzTicket and XACML/SAML messages/assertions formats	5
3 SAML security tokens expression and exchange format	8
4 XACML policy expression and messaging format	20
4.1 Generic RBAC functionality	20
4.2 XACML Core specification	22
4.3 XACML 2.0 special profiles	29
4.3.1 XACML 2.0 RBAC profile	29
4.3.2 XACML 2.0 Profile for Multiple Resources	29
4.3.3 XACML Multiple Resources profile	29
5 References	29

1 Using XACML and SAML for AuthZ decision request and security tokens exchange

This section explains how the two standards XACML and SAML are used in the discussed above AuthZ use cases and scenarios for policy expression and security assertions exchange. Some other information is also provided here to understand the overall structure of XACML and SAML. More detailed information about these standards is provided in the Appendixes.

The diagram below illustrates where SAML protocol and assertions and protocol and XACML Request/Response messages can be used in a typical policy based decision making [XACML-SAML].



- Note:
- All messages and statements semantics relates to SAML 2.0 core specification and SAML profile for XACML.
 - XACML specific messages are marked explicitly with

Figure 1.1. Using SAML and XACML for messaging and assertions

Although XACML defines XACML Request/Response messages format, it doesn't provide any suggestions about using one or another transport container or protocol. Using XACML messages directly as AuthZ assertions impose some security/integrity problems because they don't have mechanisms to bind authority (trust) or express/imply security restrictions as they are provided by the such SAML elements as Issuer or Conditions.

So, a solution proposed in the XACML profile of the SAML 2.0 provides a good combination between XACML policy expression and evaluation functionality and SAML security assertion management functionality.

Recently published SAML 2.0 specification provides even better security and improved functionality comparing to SAML 1.1:

1) features improving SAML security (via better integrity and secure context management):

- Issuer element is now obligatory top level element under root element <Assertion>, it is moved from the attribute in <Assertion> element

- <Subject> element is an (optional) top element and it is removed from the (Authn/Authz/Attribute)Statement elements as in SAML 1.1

- main sensitive elements Subject/NameID, Advice/Assertion, AttributeStatement/Assertion now have an option of encrypted elements correspondingly EncryptedID, Encrypted Assertion, EncryptedAttribute

2) better flexibility in secure context management:

- added new conditions OneTimeUse and ProxyRestriction instead of old DoNotCacheCondition

- Assertions in Advice and AuthzDecisionStatement now can be referenced by also AssertionURIRef in addition to previous AssertionIDRef only

- old element AuthorityBinding in SAML 1.1 is replaced now with new element AuthnContext that includes AuthnContextClassRef, AuthnContextDecl, AuthnContextDeclRef, or AuthenticatingAuthority

3) number of special AuthN context profiles are defined including X.509, Kerberos, PGP, XMLdsig, SSL, IP, Smartcard, mobile telephony, timesynch, etc.

4) XACML based AuthZ profile is defined by introducing element XACMLAuthzDecisionStatement/Query, XACMLPolicyStatement/Query

The picture below shows major differences between SAML 1.1 and SAML 2.0.

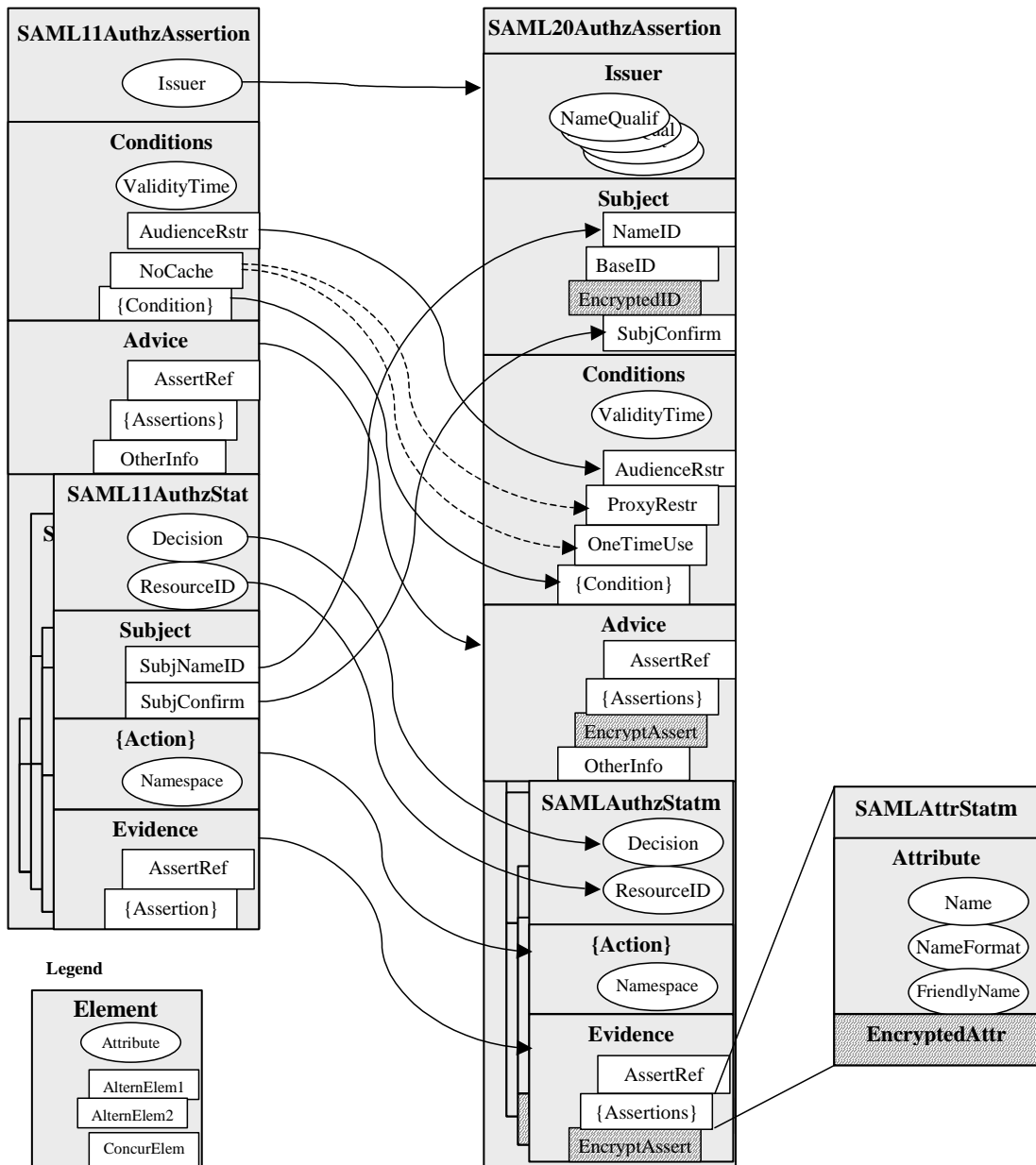


Figure 1.2. Comparison between SAML 1.1 and SAML 2.0 (for the Authorization Assertion as an example)

The picture below provides more detailed information about mapping between XACML Request/Response messages and SAML 2.0 Authorisation assertion than it was discussed in the context of using SAML 2.0 for the CNLAuthzTicket expression.

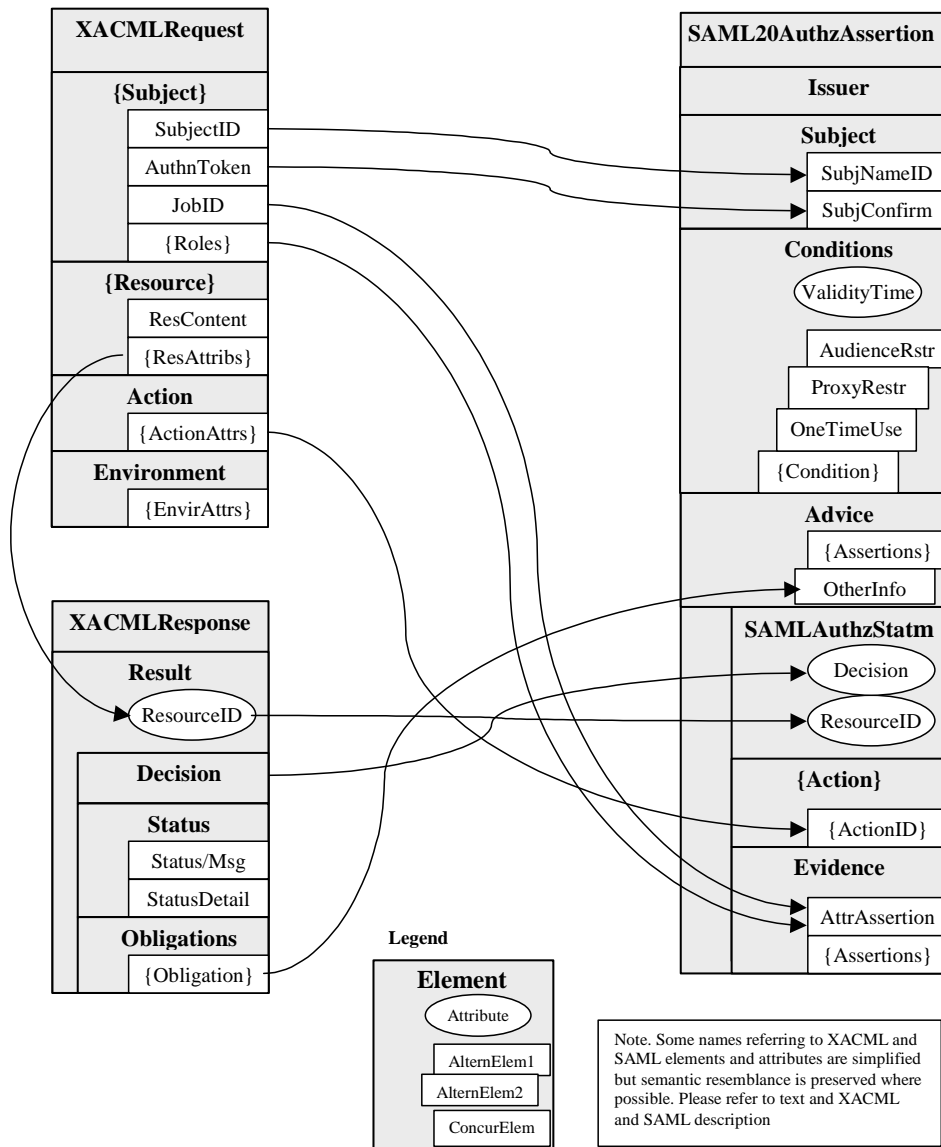


Figure 1.3. Mapping between XACML Request/Response messages and SAML 2.0 Authorisation assertion.

2 Example mapping between custom/OCE AuthzTicket and XACML/SAML messages/assertions formats

The figure below provides the graphical presentation of mapping between proposed/suggested Authorisation ticket OCEAuthzTicket format for Open Collaborative Environment (OCE) use case, XACML Request and Response messages, and SAML 2.0 Authorisation Assertion format.

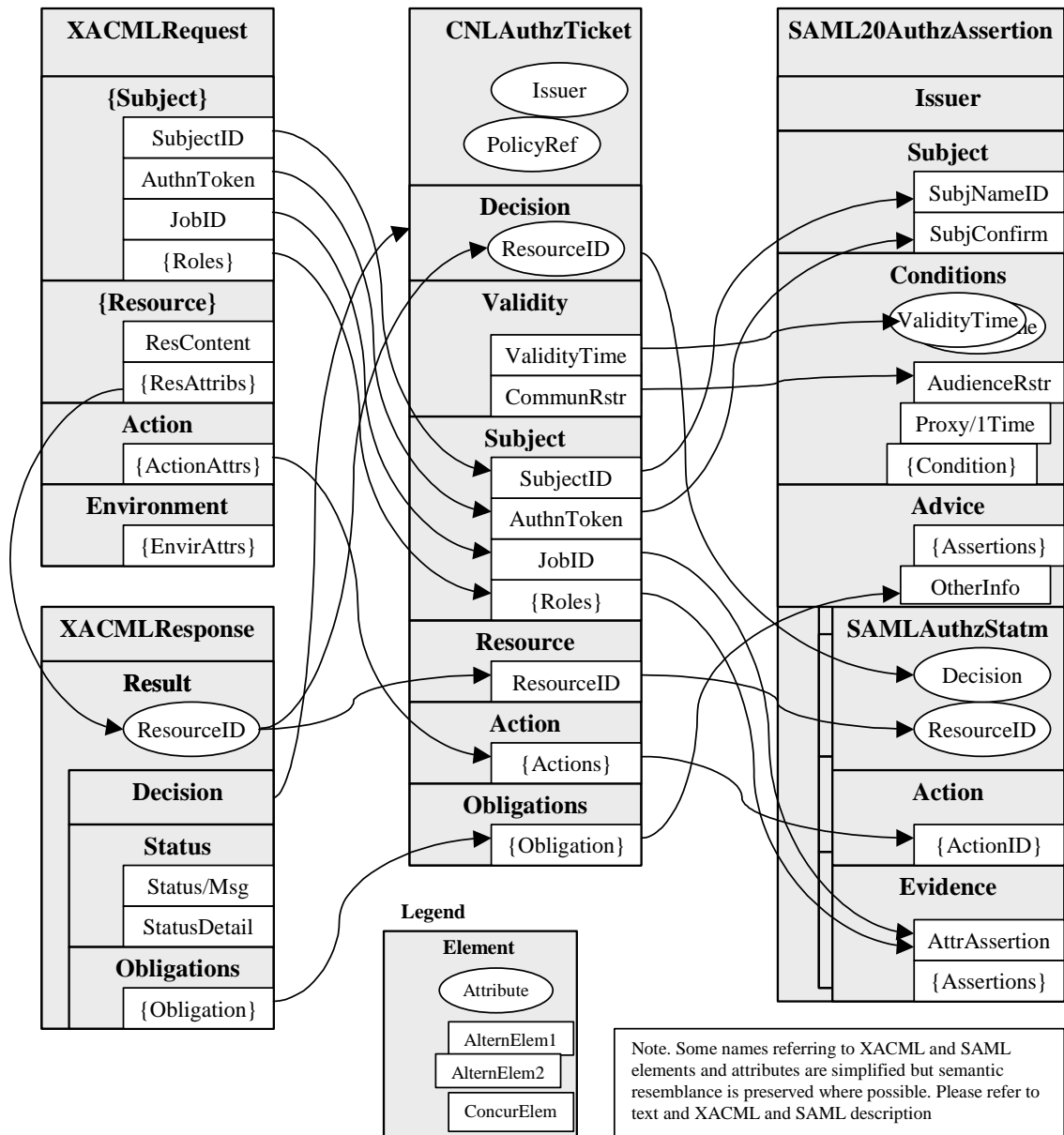


Figure 2.1. Mapping between XACML Request/Response, OCEAuthzTicket and SAML2.0 Authorization Assertion.

Due to some limitations of the XACML and SAML formats the following issues need to be discussed:

- XACML Request format allows only one Action element, although multiple Action/Attribute sub-elements. Sub-elements can provide some additional information about an action but only one Action can be evaluated in one request against a particular Action Rule in a XACML policy and XACML Response is supposed to provide a decision only for one action. No mean how to possibly combine individual decisions are available. SAML AuthzStatement may have multiple actions but they can be bound to a single ResourceID and a single Decision.
- In case of multiple actions evaluation by XACML PDP these actions should be evaluated individually and actions/results combination algorithms should be derived from the policy. Note, this case may require further research how the relations between multiple actions can be described in the request and transferred into final Authorisation decision statement.

- Multiple Resource elements provide flexibility when implementing hierarchical or multiple resources model. However, both XACML Response and SAMLAuthzDecisionStatement generally allow only ResourceID reference.
- Multiple Subject elements allow flexibility when implementing hierarchical RBAC model when some actions require superior subject/role approval, or in case when initiator, requestor and receiver of Action are different entities/subjects.
However, SAML Assertion format allows only one Subject. In case when there is a need to keep information about other subjects, this information can be placed into SAMLAssertion/Advice element or into SAMLAuthzDecisionStatement/Evidence element in form of SAML Assertions.
- Obligation which is an optional element of the Policy and a possible component of the XACML Response can be placed into Condition simple element or into extension sub-element under SAML Advice element

Suggested OCEAuthzTicket validity parameters include ValidityTime defining validity period “NotBefore” and “NotOneOrAfter”, CommunityRestriction defining trusted community can be mapped in the following way:

- ValidityTime containing two attributes “NotBefore” and “NotOneOrAfter” can be mapped directly into related attributes of the SAMLAssertion/Conditions element
- Because of SAML Conditions element may have only one of elements AudienceRestriction, ProxyRestriction, OneTimeUse sub-elements or multiple Condition sub-element, CNLAuthzTicket validity parameters should limited to have only CommunityRestriction or both CommunityRestriction and NumberOfUse should be represented in a simple/textual form and placed into multiple Condition elements.

All other mapping issues doesn't have problems and can be achieved in the following way:

- SubjectID and Subject AuthenticationToken can be placed into Subject/(NameID or BasedID) element and SubjectConfirmation/ SubjectConfirmationData element correspondently
- Subject attributes such as JobID and roles can be placed into SAMLAuthzDecisionStatement/Evidence element in an a form of SAML Attribute Assertion.

More information about XACML special profiles is provided in the XACML section.

Another solution can be to use SAML profile of the XACML that defines XACMLAuthzDecisionStatement/Query, XACMLPolicyStatement/Query that allow to include original XACML Request and Response messages directly into SAML Assertions (see Figure below).

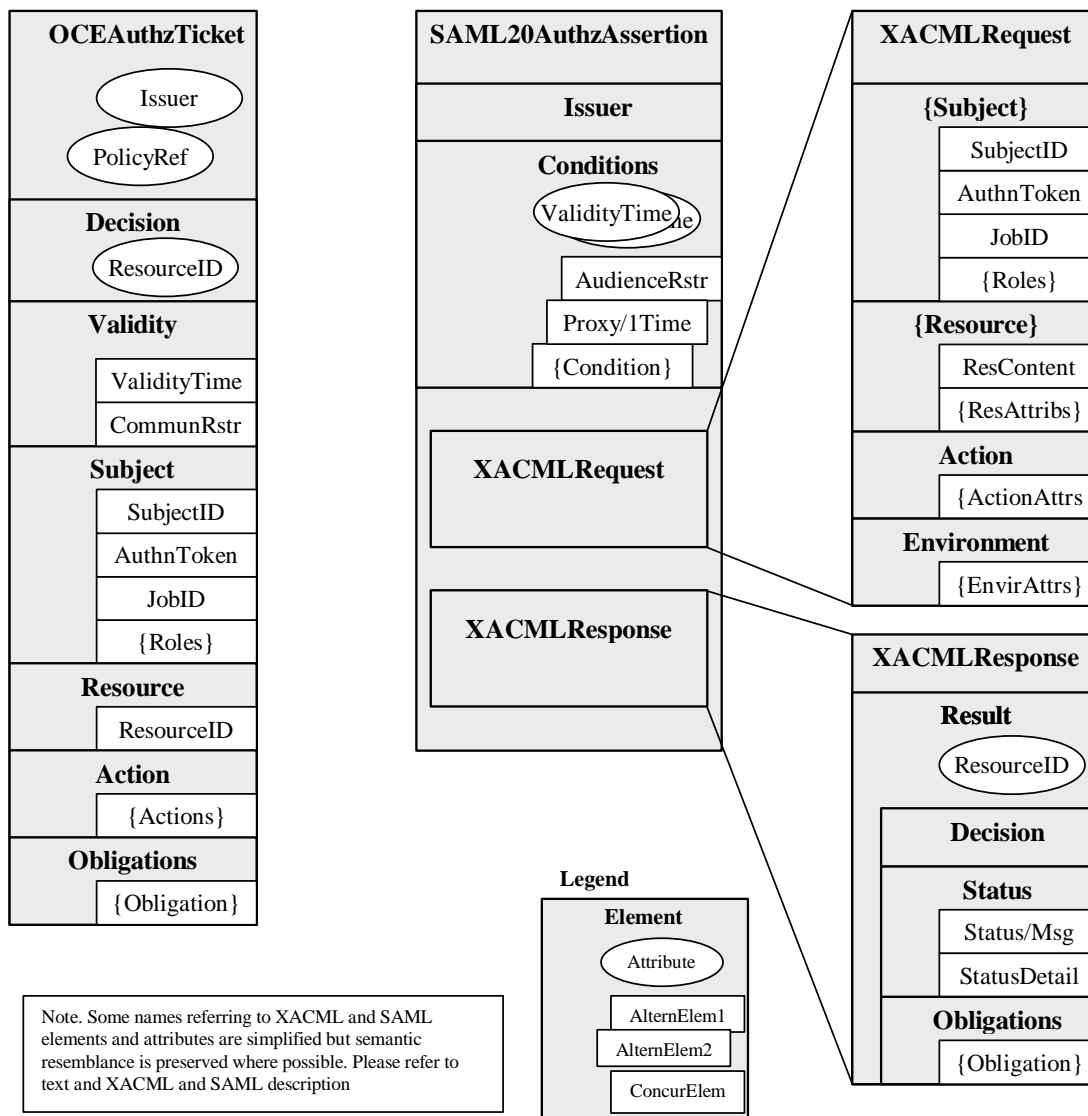


Figure 2.2. Mapping between OCEAuthzTicket and SAMLAuthzAssertion using XACML profile.

3 SAML security tokens expression and exchange format

3.1 General information and comparison between SAML 1.1 and SAML 2.0

This section provides basic information about the structure and elements of the SAML 2.0 format. Examples are provided as an illustration to the discussed above CNL Authorisation token format.

Comparison between currently used SAML 1.1 and recently published SAML 2.0 specifications is provided for reference purposes only:

- 1) features improving SAML security (via better integrity and secure context management):

- Issuer element is now obligatory top level element under root element <Assertion>, it is moved from the attribute in <Assertion> element

- <Subject> element is an (optional) top element and it is removed from the (Authn/Authz/Attribute)Statement elements as in SAML 1.1

- main sensitive elements Subject/NameID, Advice/Assertion, AttributeStatement/Assertion now have an option of encrypted elements correspondingly EncryptedID, Encrypted Assertion, EncryptedAttribute

2) better flexibility in secure context management:

- added new conditions OneTimeUse and ProxyRestriction instead of old DoNotCacheCondition

- Assertions in Advice and AuthzDecisionStatement now can be referenced by also AssertionURIRef in addition to previous AssertionIDRef only

- old element AuthorityBinding in SAML 1.1 is replaced now with new element AuthnContext that includes AuthnContextClassRef, AuthnContextDecl, AuthnContextDeclRef, or AuthenticatingAuthority

3) number of special AuthN context profiles are defined including X.509, Kerberos, PGP, XMLdsig, SSL, IP, Smartcard, mobile telephony, timesynch, etc.

4) XACML based AuthZ profile is defined by introducing element XACMLAuthzDecisionStatement/Query, XACMLPolicyStatement/Query

Picture below shows major differences between SAML 1.1 and SAML 2.0

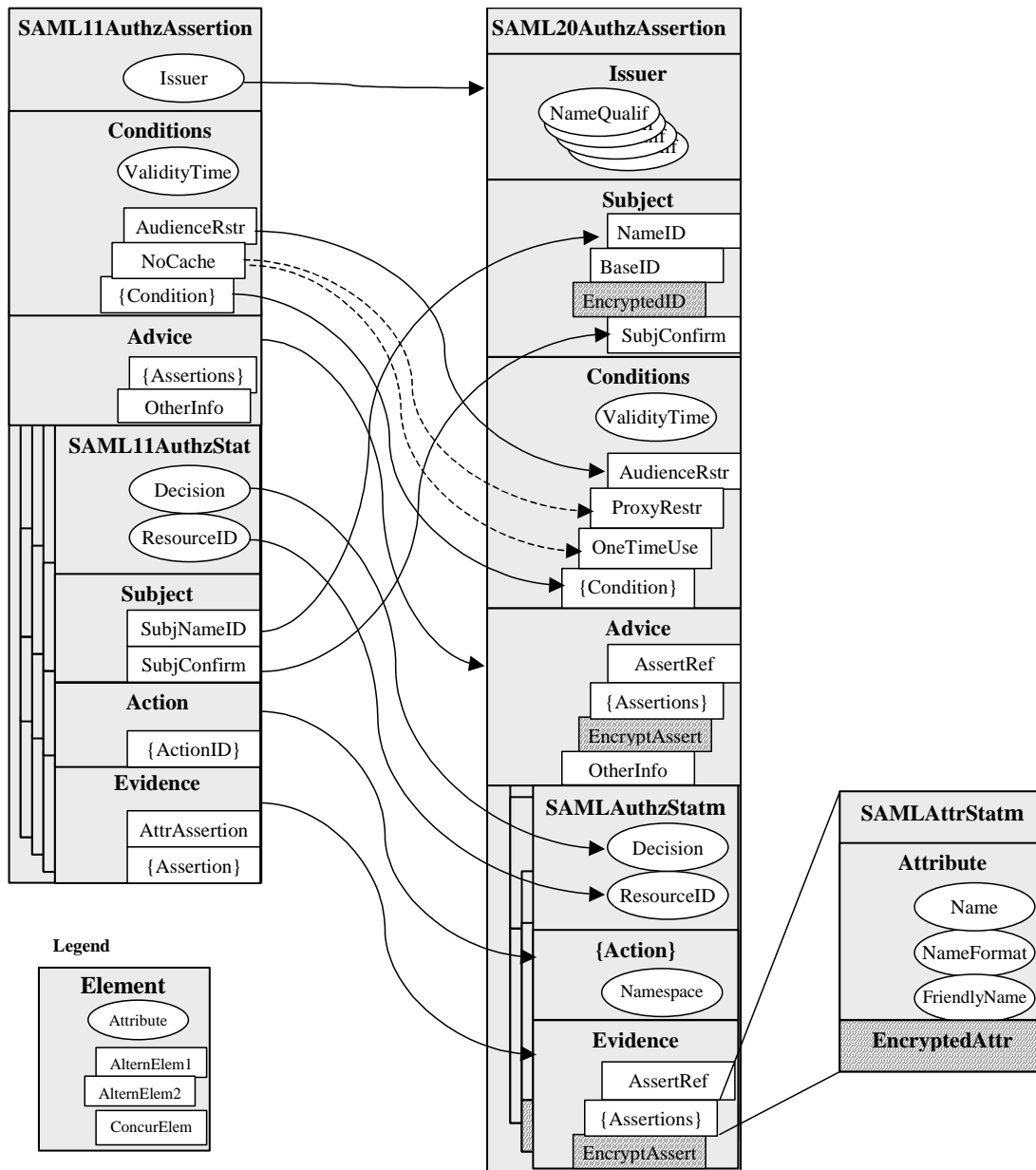


Figure 3.1. Comparison between SAML 1.1 and SAML 2.0 (for the Authorization Assertion)

Figures below provide more detailed breakdown for SAML 2.0 Assertion format. Subject element contains all required information to describe Subject including provided credentials in the SubjectConfirmation element. SAML Assertion provides the facility to describe conditions for assertion/credentials use and validity in the Conditions element that contains auditorium/community limitation, caching/proxy restrictions and time validity constrains. Security context or e.g. credentials delegation and/or usage history can be placed into the Advice element. Both Condition and Advice elements are extendable but in a bit different way. The Condition element can contain extendable condition element as a specific named instance of the abstract Condition element. The Advice element can contain any extendable element using any external namespace.

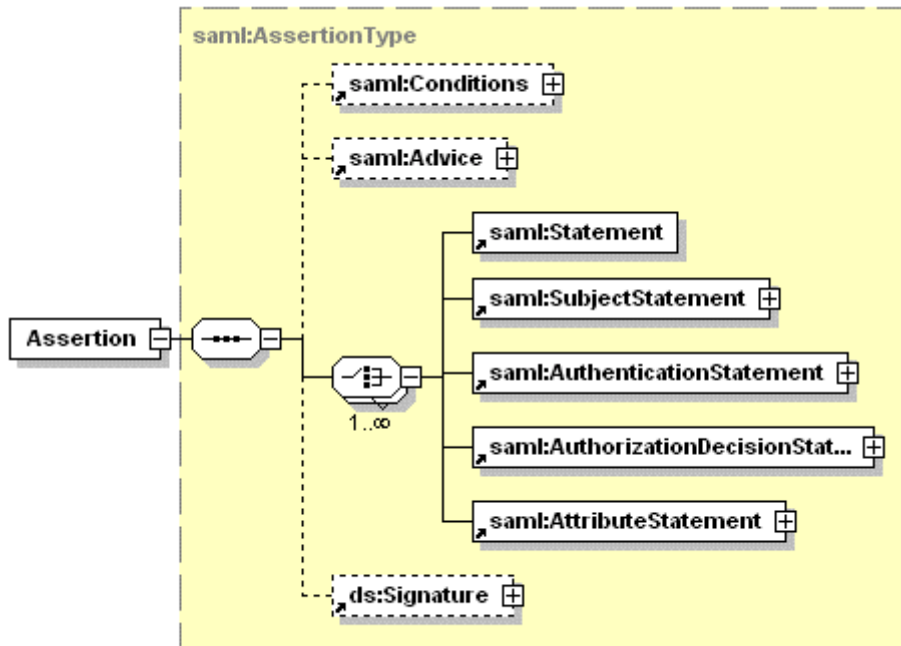
All sensitive SAML components can be encrypted. However, SAML 2.0 defines some encrypted elements directly in the schema.

3.2 SAML 1.1 and SAML 2.0 Top Level Elements

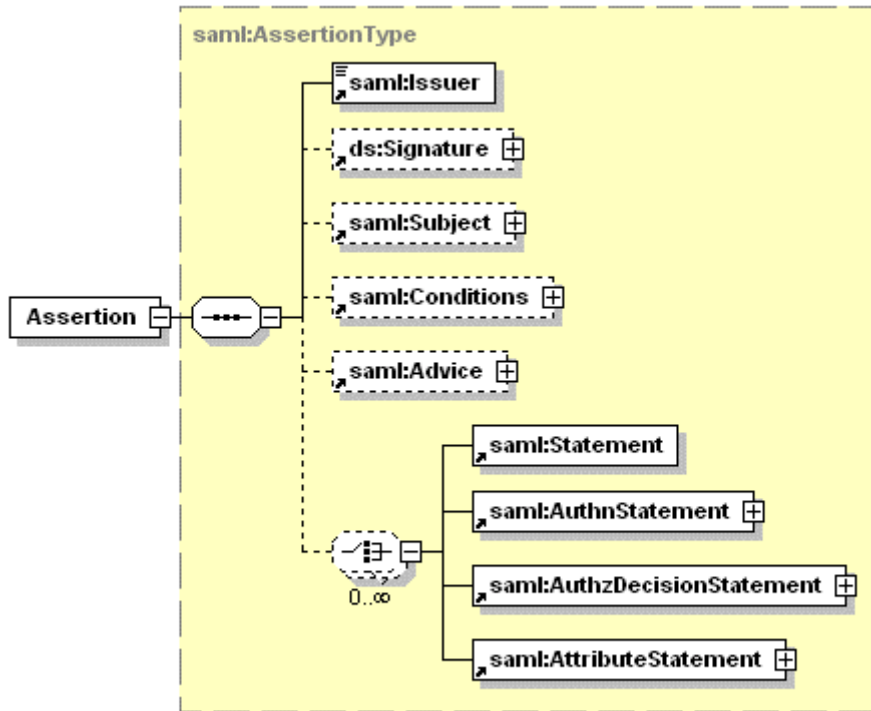
3.2.1 SAML Assertion Element

SAML 2.0 Assertion element content can be expressed in the compact XML DTD format as follows:

```
<!ELEMENT Assertion (Issuer, Signature?, Subject?, Conditions?, Advice?,  
(Statement | AuthnStatement | AuthzDecisionStatement | AttributeStatement)*)>  
<!ATTLIST Assertion  
  Version CDATA #REQUIRED  
  ID ID #REQUIRED  
  IssueInstant CDATA #REQUIRED  
>
```



a) SAML 1.1 Assertion element



b) SAML 2.0 Assertion element

Figure 3.2. SAML 1.1 and SAML 2.0 root element Assertion (comparison).

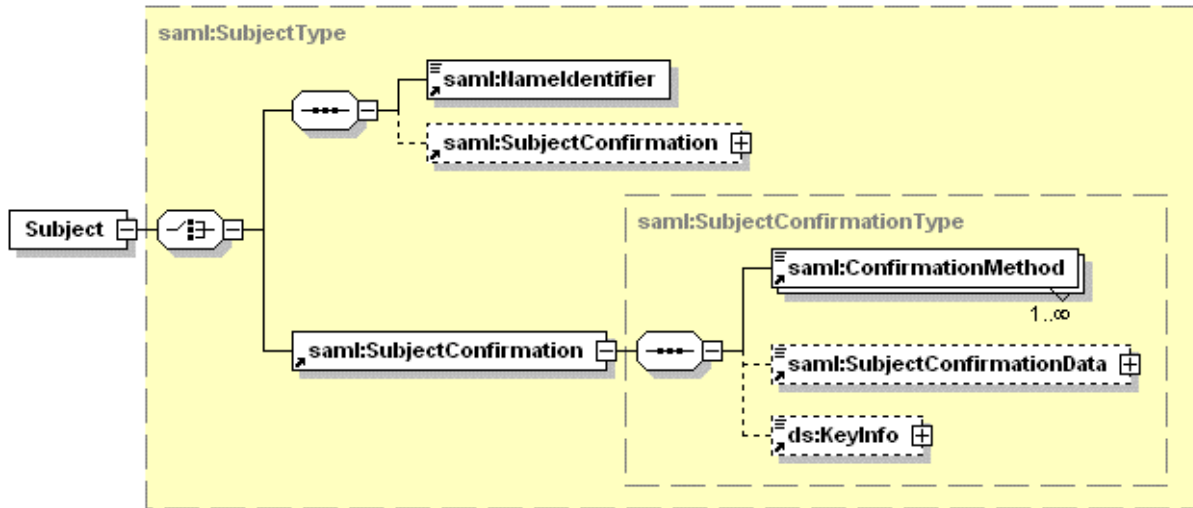
3.2.2 SAML Subject Element

SAML 2.0 Subject element is an optional top level element which contains the following sub-elements:

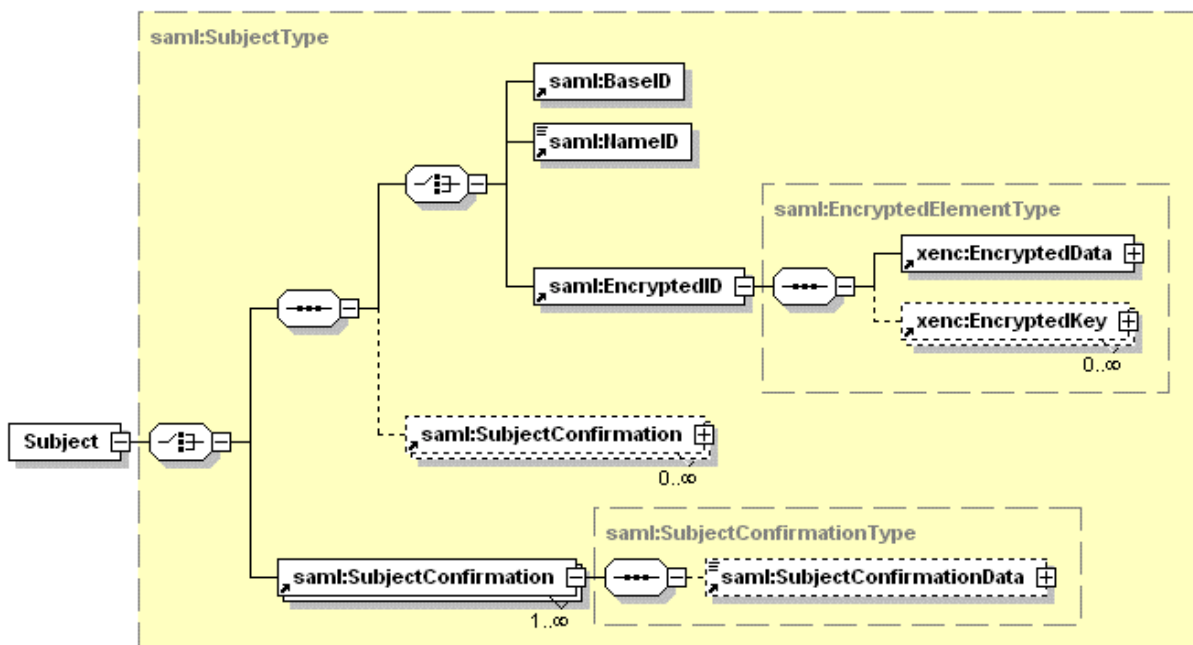
```

<!ELEMENT Subject (((BaseID | NameID | EncryptedID), SubjectConfirmation*) |
SubjectConfirmation+)>
<!ELEMENT SubjectConfirmation (SubjectConfirmationData?)>
<!ATTLIST SubjectConfirmation
    Method CDATA #REQUIRED
>
<!ELEMENT SubjectConfirmationData (#PCDATA | *)*>
<!ATTLIST SubjectConfirmationData
    NotBefore CDATA #IMPLIED
    NotOnOrAfter CDATA #IMPLIED
    Recipient CDATA #IMPLIED
    InResponseTo CDATA #IMPLIED
    Address CDATA #IMPLIED
>
<!ELEMENT SubjectLocality EMPTY>
<!ATTLIST SubjectLocality
    Address CDATA #IMPLIED
    DNSName CDATA #IMPLIED
>

```



a) SAML 1.1 Subject element



b) SAML 2.0 Subject element

Figure 3.3. SAML 1.1 and SAML 2.0 top level element Subject (comparison).

3.2.3 SAML Authentication Statement Element

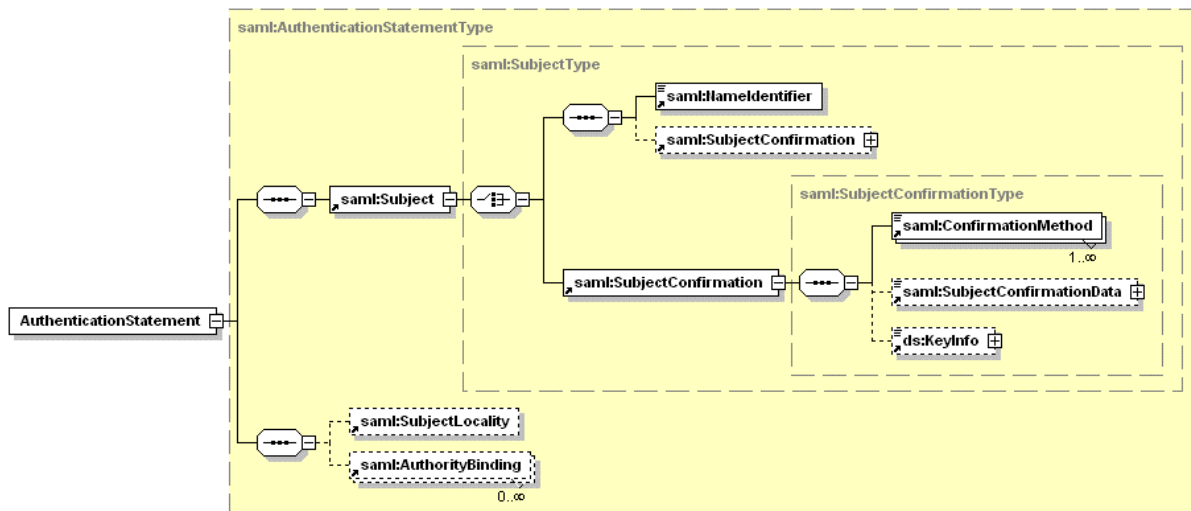
SAML 2.0 AuthnStatement has the following structure:

```
<!ELEMENT AuthnStatement (SubjectLocality?, AuthnContext)>
<!ATTLIST AuthnStatement
    AuthnInstant CDATA #REQUIRED
    SessionIndex CDATA #IMPLIED
    SessionNotOnOrAfter CDATA #IMPLIED
>
<!ELEMENT AuthnContext (((AuthnContextClassRef, (AuthnContextDecl |
AuthnContextDeclRef)?) | (AuthnContextDecl | AuthnContextDeclRef)),
AuthenticatingAuthority*)>
```

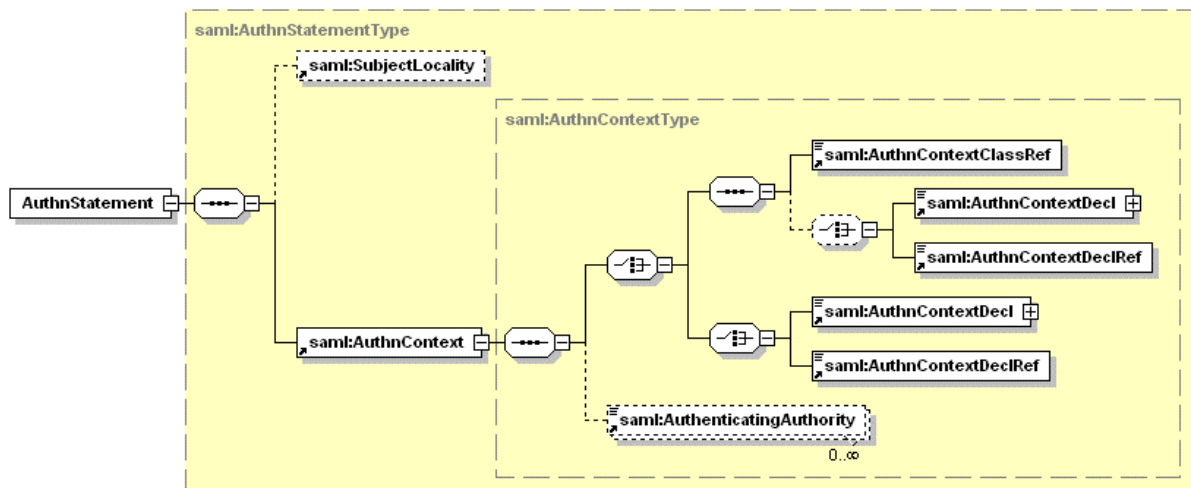
```

<!ELEMENT AuthnContextClassRef (#PCDATA)>
<!ELEMENT AuthnContextDecl (#PCDATA)>
<!ELEMENT AuthnContextDeclRef (#PCDATA)>
<!ELEMENT AuthenticatingAuthority (#PCDATA)>

```



a) SAML 1.1 AuthenticationStatement element



b) SAML 2.0 AuthnStatement element

Figure 3.4. SAML 1.1 AuthenticationStatement and SAML 2.0 AuthnStatement elements (comparison).

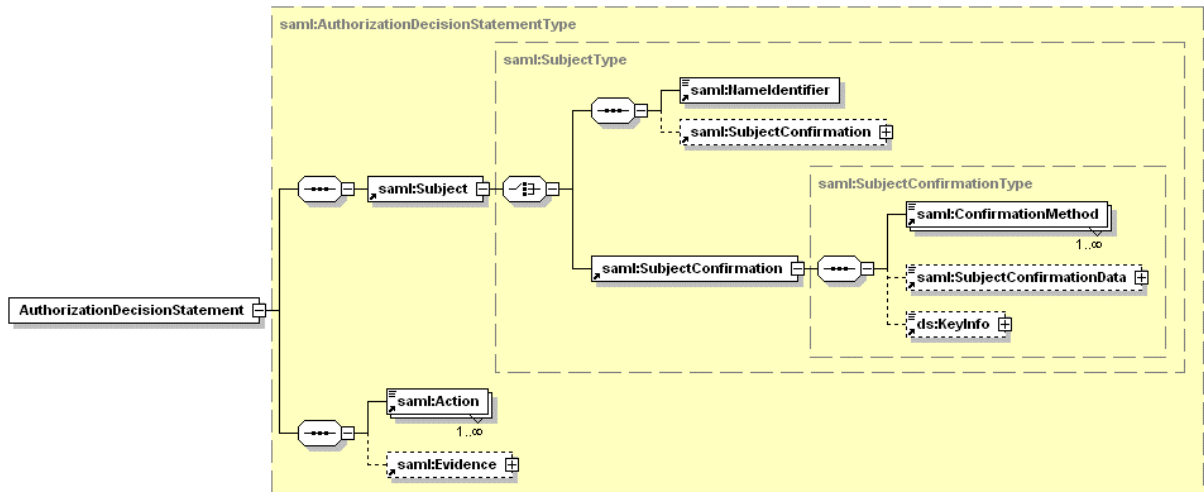
3.2.4 SAML Authorisation Decision Statement Element

SAML 2.0 AuthzDecisionStatement has the following structure:

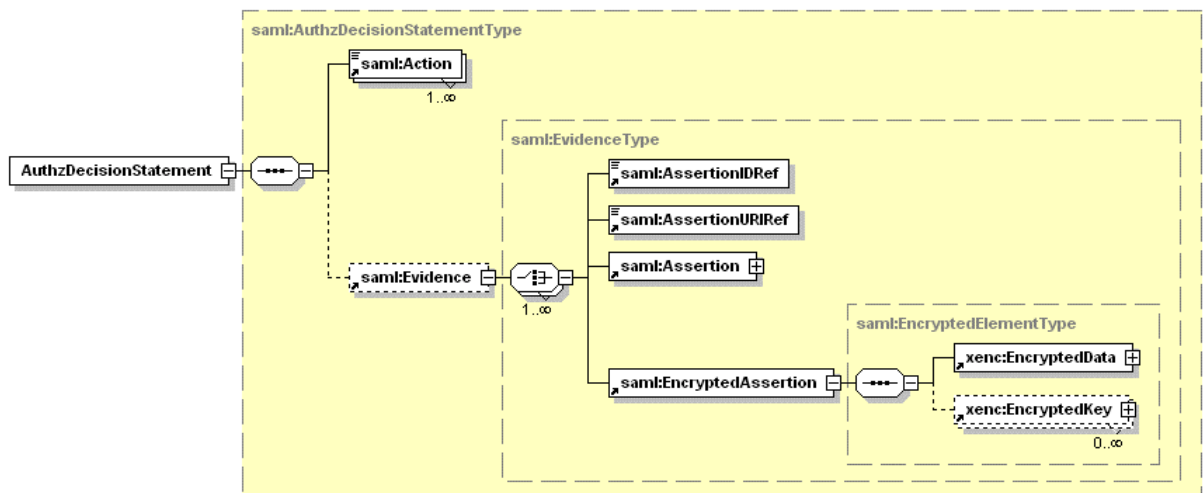
```

<!ELEMENT AuthzDecisionStatement (Action+, Evidence?)>
<!ATTLIST AuthzDecisionStatement
  Resource CDATA #REQUIRED
  Decision (Permit | Deny | Indeterminate) #REQUIRED
>
<!ELEMENT Evidence (AssertionIDRef | AssertionURIRef | Assertion |
  EncryptedAssertion)+>

```



a) SAML 1.1 AuthorizationDecisionStatement element



b) SAML 2.0 AuthzDecisionStatement element

Figure 3.5. SAML 1.1 AuthorizationDecisionStatement and SAML 2.0 AuthzDecisionStatement elements (comparison).

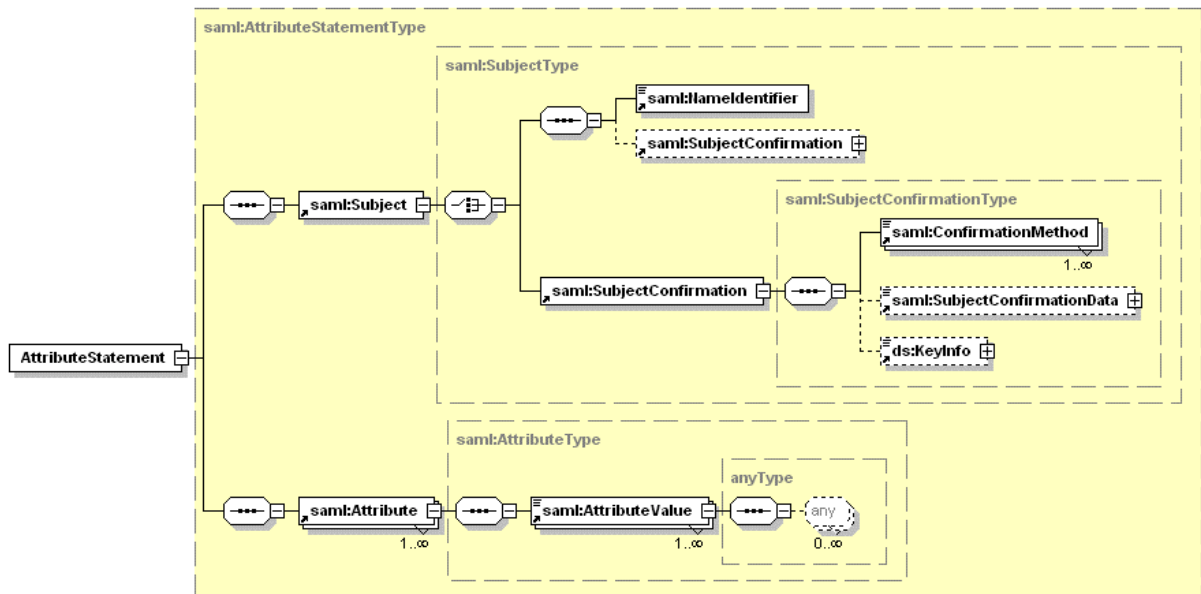
3.2.5 SAML Attribute Statement Element

Figure 3.6 below shows the structure of the SAML AttributeStatement element. It contains the following elements:

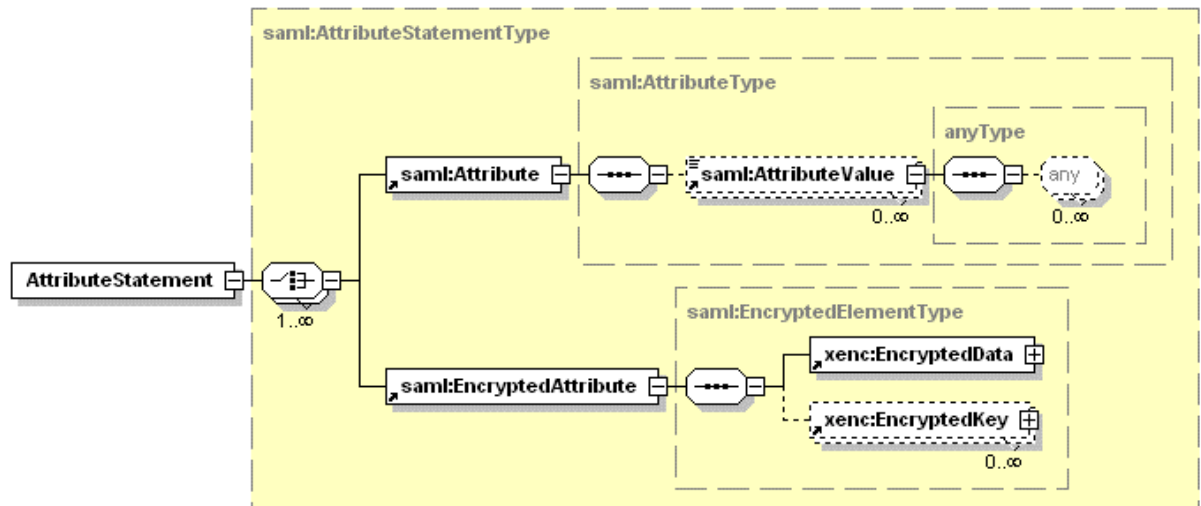
```

<!ELEMENT AttributeStatement (Attribute | EncryptedAttribute)+>
<!ELEMENT Attribute (AttributeValue)*>
<!--
  Name CDATA #REQUIRED
  NameFormat CDATA #IMPLIED
  FriendlyName CDATA #IMPLIED
-->

```



a) SAML 1.1 AttributeStatement element



b) SAML 2.0 AttributeStatement element

Figure 3.6. SAML 1.1 and SAML 2.0 AttributeStatement elements (comparison).

3.3 Examples of SAML 1.1 and SAML 2.0 Assertions

3.3.1 SAML 1.1 and SAML 2.0 Authentication Assertions

The examples of the SAML 1.1 and SAML 2.0 Authentication Assertions created with the OpenSAML 1.1 package and SAML 2.0 extensions will look like follows:

```
<Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
AssertionID="b3edb01c39f5f08897341c84b7181afe" IssueInstant="2004-12-
```

```

29T18:07:23.367Z" Issuer="cnl:subject:CNLAAAauthority" MajorVersion="1"
MinorVersion="1">
  <Conditions NotBefore="2004-12-04T23:00:00.000Z" NotOnOrAfter="2004-12-
22T21:22:22.000Z"/>
  <AuthenticationStatement AuthenticationInstant="2004-12-29T18:07:23.227Z"
AuthenticationMethod="AuthenticationMethod_X509_PublicKey">
    <Subject>
      <NameIdentifier Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress"
NameQualifier="cnl:subject:customer">WHO740@users.collaboratory.nl</NameIdentifier>
      <SubjectConfirmation>
        <ConfirmationMethod>email</ConfirmationMethod>
        <ConfirmationMethod>callback</ConfirmationMethod>
      </SubjectConfirmation>
    </Subject>
    <SubjectLocality DNSAddress="dns.collaboratory.nl" IPAddress="192.30.180.22"/>
  </AuthenticationStatement>
</Assertion>

```

Listing 3.1. Example SAML 1.1 Authentication Assertion

```

<Assertion xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
ID="e0fcd9f023440a05d540ba365e1ed1fe" IssueInstant="2004-12-29T17:14:24.085Z"
Version="2.0">
  <Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:X509SubjectName"
NameQualifier="cnl:subject:subject:AAAAuthority">CN=Agent Smith, O=Matrix,
C=NL</Issuer>
  <Subject>
    <NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-format:emailAddress"
NameQualifier="cnl:subject:customer">WHO740@users.collaboratory.nl</NameID>
    <SubjectConfirmation>
      <ConfirmationMethod>email</ConfirmationMethod>
      <ConfirmationMethod>callback</ConfirmationMethod>
    </SubjectConfirmation>
  </Subject>
  <Conditions NotBefore="2004-12-28T23:00:00.000Z" NotOnOrAfter="2005-01-
29T21:22:22.000Z"/>
  <AuthnStatement AuthenticationInstant="2004-12-29T17:14:23.875Z"
AuthenticationMethod="AuthenticationMethod_X509_PublicKey">
    <SubjectLocality DNSAddress="dns.collaboratory.nl" IPAddress="192.30.180.22"/>
  </AuthnStatement>
</Assertion>

```

Listing 3.2. Example SAML 2.0 Authentication Assertion

3.3.2 SAML 1.1 and SAML 2.0 Authorisation Assertions

The examples of the SAML 1.1 and SAML 2.0 Authorisation Assertions containing also Attribute Assertion in the Evidence element will look like follows:

```

<Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
AssertionID="dd6101a2c7b3d8656f479f8bdd6cfea3" IssueInstant="2005-01-
06T21:04:56.523Z" Issuer="cnl:subject:CNLAAAauthority" MajorVersion="1"
MinorVersion="1">
  <Conditions NotBefore="2004-12-04T23:00:00.000Z" NotOnOrAfter="2004-12-
22T21:22:22.000Z"/>
  <AuthorizationDecisionStatement Decision="@Resource;Permit"
Resource="http://resources.collaboratory.nl/Phillips_XPS1">
    <Subject>
      <NameIdentifier Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress"
NameQualifier="cnl:subject:customer">WHO740@users.collaboratory.nl</NameIdentifier>
      <SubjectConfirmation>
        <ConfirmationMethod>email</ConfirmationMethod>
        <ConfirmationMethod>callback</ConfirmationMethod>

```

```

    </SubjectConfirmation>
  </Subject>
  <Action Namespace="urn:oasis:names:tc:SAML:1.0:action:cnl:action">CNLaction02:
zoom</Action>
  <Action Namespace="urn:oasis:names:tc:SAML:1.0:action:cnl:action">CNLaction01:
2Dscan</Action>
  <Evidence>
    <AssertionIDReference>2355789adcebb</AssertionIDReference>
    <AssertionIDReference>@resourceId;Permit</AssertionIDReference>
    <Assertion AssertionID="aal98b5bbc6f7e16b9dad16c6d5a3d38" IssueInstant="2005-
01-06T21:04:56.392Z" Issuer="cnl:subject:CNLAAAauthority" MajorVersion="1"
MinorVersion="1">
      <Conditions NotBefore="2004-12-04T23:00:00.000Z" NotOnOrAfter="2004-12-
22T21:22:22.000Z"/>
      <AttributeStatement>
        <Subject>
          <NameIdentifier Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress"
NameQualifier="cnl:subject:customer">HEIS007@staff.collaboratory.nl</NameIdentifier
>
          <SubjectConfirmation>
            <ConfirmationMethod>email</ConfirmationMethod>
            <ConfirmationMethod>callback</ConfirmationMethod>
          </SubjectConfirmation>
        </Subject>
        <Attribute xmlns:typens="urn:cnl"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
AttributeName="AttributeSubject" AttributeNamespace="urn:cnl">
          <AttributeValue
xsi:type="typens:subject">@cnl:subject:role:manager</AttributeValue>
          <AttributeValue
xsi:type="typens:subject">cnl:subject:role</AttributeValue>
          <AttributeValue xsi:type="typens:subject">jobID</AttributeValue>
        </Attribute>
      </AttributeStatement>
    </Assertion>
  </Evidence>
</AuthorizationDecisionStatement>
</Assertion>

```

Listing 3.3. Example SAML 1.1 Authorisation Assertion

```

<Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
AssertionID="b444a40244e84092d862089eaff8e878" IssueInstant="2004-12-
29T18:07:41.423Z" Issuer="cnl:subject:CNLAAAauthority" MajorVersion="1"
MinorVersion="1">
  <Conditions NotBefore="2004-12-04T23:00:00.000Z" NotOnOrAfter="2004-12-
22T21:22:22.000Z"/>
  <AuthorizationDecisionStatement Decision="@Resource;Permit"
Resource="http://resources.collaboratory.nl/Phillips_XPS1">
    <Subject>
      <NameIdentifier Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress"
NameQualifier="cnl:subject:customer">WHO740@users.collaboratory.nl</NameIdentifier>
      <SubjectConfirmation>
        <ConfirmationMethod>email</ConfirmationMethod>
        <ConfirmationMethod>callback</ConfirmationMethod>
      </SubjectConfirmation>
    </Subject>
    <Action Namespace="urn:oasis:names:tc:SAML:1.0:action:cnl:action">CNLaction02:
zoom</Action>
    <Action Namespace="urn:oasis:names:tc:SAML:1.0:action:cnl:action">CNLaction01:
2Dscan</Action>
    <Evidence>
      <AssertionIDReference>2355789adcebb</AssertionIDReference>
      <AssertionIDReference>@resourceId;Permit</AssertionIDReference>
      <Assertion AssertionID="dcfb4382637317bd7a8d7844d7e47b09" IssueInstant="2004-
12-29T18:07:41.353Z" Issuer="cnl:subject:CNLAAAauthority" MajorVersion="1"
MinorVersion="1">

```

```

        <Conditions NotBefore="2004-12-04T23:00:00.000Z" NotOnOrAfter="2004-12-
22T21:22:22.000Z"/>
        <AttributeStatement>
            <Subject>
                <NameIdentifier Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress"
NameQualifier="cnl:subject:customer">HEIS007@staff.collaboratory.nl</NameIdentifier
>
                <SubjectConfirmation>
                    <ConfirmationMethod>email</ConfirmationMethod>
                    <ConfirmationMethod>callback</ConfirmationMethod>
                </SubjectConfirmation>
            </Subject>
            <Attribute xmlns:typens="urn:cnl"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
AttributeName="AttributeSubject" AttributeNamespace="urn:cnl">
                <AttributeValue
xsi:type="typens:subject">@cnl:subject:role:manager</AttributeValue>
                <AttributeValue
xsi:type="typens:subject">cnl:subject:role</AttributeValue>
                <AttributeValue xsi:type="typens:subject">jobID</AttributeValue>
            </Attribute>
        </AttributeStatement>
    </Assertion>
</Evidence>
</AuthorizationDecisionStatement>
</Assertion>

```

Listing 3.4. Example SAML 2.0 Authorisation Assertion

3.3.3 SAML 1.1 and SAML 2.0 Attribute Assertions

The examples of the SAML 1.1 and SAML 2.0 Attribute Assertions created with the upgraded OpenSAML 1.1 package will look like follows:

```

<Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol"
AssertionID="aa198b5bbc6f7e16b9dad16c6d5a3d38" IssueInstant="2005-01-
06T21:04:56.392Z" Issuer="cnl:subject:CNLAAAauthority" MajorVersion="1"
MinorVersion="1">
    <Conditions NotBefore="2004-12-04T23:00:00.000Z" NotOnOrAfter="2004-12-
22T21:22:22.000Z"/>
    <AttributeStatement>
        <Subject>
            <NameIdentifier Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress"
NameQualifier="cnl:subject:customer">HEIS007@staff.collaboratory.nl</NameIdentifier
>
            <SubjectConfirmation>
                <ConfirmationMethod>email</ConfirmationMethod>
                <ConfirmationMethod>callback</ConfirmationMethod>
            </SubjectConfirmation>
        </Subject>
        <Attribute xmlns:typens="urn:cnl" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
AttributeName="AttributeSubject" AttributeNamespace="urn:cnl">
            <AttributeValue
xsi:type="typens:subject">@cnl:subject:role:manager</AttributeValue>
            <AttributeValue xsi:type="typens:subject">cnl:subject:role</AttributeValue>
            <AttributeValue xsi:type="typens:subject">jobID</AttributeValue>
        </Attribute>
    </AttributeStatement>
</Assertion>

```

Listing 3.5. Example SAML 1.1 Attribute Assertion

```

<Assertion xmlns="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
ID="b4d00e1500d2a10a43d3d2fb5a578028" IssueInstant="2004-12-29T17:17:24.164Z"
Version="2.0">
  <Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:X509SubjectName"
NameQualifier="cnl:subject:subject:AAAAuthority">CN=Agent Smith, O=Matrix,
C=NL</Issuer>
  <Subject>
    <NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-format:emailAddress"
NameQualifier="cnl:subject:customer">HEIS007@staff.collaboratory.nl</NameID>
    <SubjectConfirmation>
      <ConfirmationMethod>email</ConfirmationMethod>
      <ConfirmationMethod>callback</ConfirmationMethod>
    </SubjectConfirmation>
  </Subject>
  <Conditions NotBefore="2004-12-28T23:00:00.000Z" NotOnOrAfter="2005-01-
29T21:22:22.000Z"/>
  <AttributeStatement>
    <Attribute xmlns:typens="urn:cnl" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
AttributeName="AttributeSubject" AttributeNamespace="urn:cnl">
      <AttributeValue
xsi:type="typens:subject">@cnl:subject:role:manager</AttributeValue>
      <AttributeValue xsi:type="typens:subject">cnl:subject:role</AttributeValue>
      <AttributeValue xsi:type="typens:subject">jobID</AttributeValue>
    </Attribute>
  </AttributeStatement>
</Assertion>

```

Listing A.6. Example SAML 2.0 Attribute Assertion

4 XACML policy expression and messaging format

4.1 Generic RBAC functionality

The generic Authorisation infrastructure consists of

- RBE (Rule Based Engine) as a central policy based decision making point,
- PEP (Policy Enforcement Point) providing Resource specific AuthZ decision request/response handling and policy defined obligations execution,
- PAP (Policy Authority Point) or Policy DB as a policy storage (in general, distributed),
- PIP (Policy Information Point) providing external policy context and attributes to the RBE including subject credentials and attributes verification
- RIP (Resource Information Point) that provides resource context.
- AA (Attribute Authority) that manages user attributes

To allow user access to the resource, Resource Agent requests via a Policy Enforcement Point (PEP) an authorisation decision from a Policy Decision Point (PDP) that evaluates the authorisation

request against the policy defined for a particular job, resource and user attributes/roles. The access policy is defined by the resource owner and stored in the policy repository.

The PEP and PDP may also request specific user attributes or credentials from the Authentication service, or additional information from the Resource/Instrument.

Figure 4.1 illustrates a basic Authorisation functionality that implements the generic RBAC model.

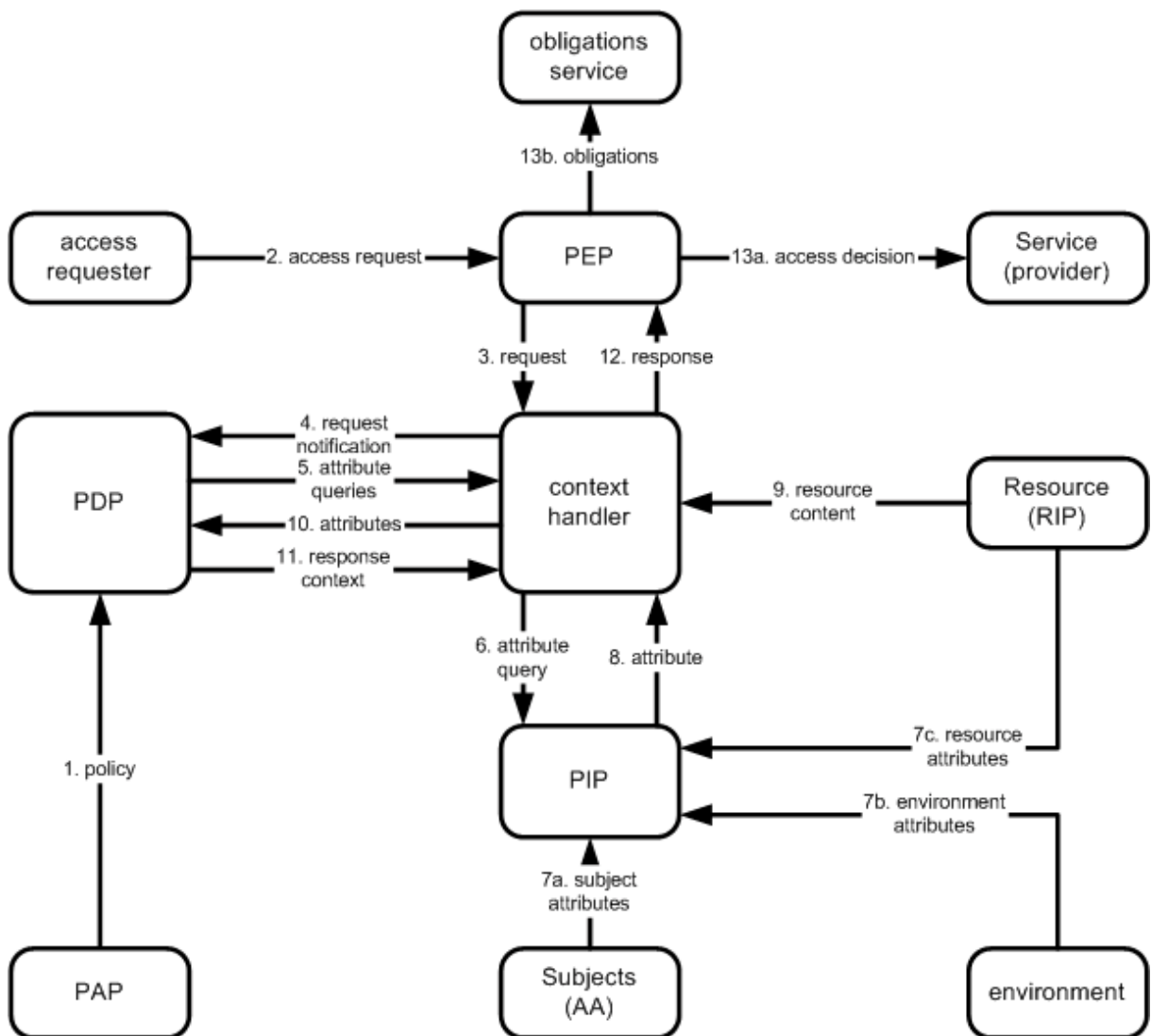


Figure 4.1. RBAC based access control system components and dataflows.

In details, the XACML RBAC model operates by the following steps [XACML 2.0]:

1. **PAPs** write **policies** and **policy sets** and make them available to the **PDP**. These **policies** or **policy sets** represent the complete policy for a specified **target**.
2. The access requester sends a request for access to the **PEP**.

3. The **PEP** sends the request for **access** to the **context handler** in its native request format, optionally including **attributes** of the **subjects**, **resource**, **action** and **environment**.
4. The **context handler** constructs an XACML request **context** and sends it to the **PDP**.
5. The **PDP** requests any additional **subject**, **resource**, **action** and **environment attributes** from the **context handler**.
6. The context handler requests the attributes from a **PIP**.
7. The **PIP** obtains the requested **attributes**.
8. The **PIP** returns the requested **attributes** to the **context handler**.
9. Optionally, the **context handler** includes the **resource** in the **context**.
10. The **context handler** sends the requested **attributes** and (optionally) the **resource** to the **PDP**. The **PDP** evaluates the **policy**.
11. The **PDP** returns the response **context** (including the **authorization decision**) to the **context handler**.
12. The **context handler** translates the response **context** to the native response format of the **PEP**. The **context handler** returns the response to the **PEP**.
13. If **access** is permitted, then the **PEP** permits **access** to the **resource**; otherwise, it denies **access**. The **PEP** fulfils the **obligations**, generally, for both cases of possible PDP solutions.

4.2 XACML Core specification¹

XACML provides a format for expressing policy for the generic RBAC used by PDP and define a simple Request/Response messages format.

XACML (eXtensible Access Control Markup Language) defines reach policy format for access control based on "Subject-Resource-Action" triad attributes. XACML defines format for policy and request/response messages.

Decision request sent in a Request message provides context for policy-based decision. The complete policy applicable to a particular **decision request** may be composed of a number of individual **rules** or **policies**. Few policies may be combined to form the single policy applicable to the request.

¹ Currently this section provides text and example for the XACML 1.0 version. Next updates will add also comparison with the XACML 2.0.

XACML defines three top-level policy elements: <Rule>, <Policy> and <PolicySet>. The <Rule> element contains a Boolean expression that can be evaluated in isolation, but that is not intended to be accessed in isolation by a **PDP**. So, it is not intended to form the basis of an **authorization decision** by itself. It is intended to exist in isolation only within an XACML **PAP**, where it may form the basic unit of management, and be re-used in multiple **policies**.

The <Policy> element contains a set of <Rule> elements and a specified procedure for combining the results of their evaluation. It is the basic unit of **policy** used by the **PDP**, and so it is intended to form the basis of an **authorization decision**.

The <PolicySet> element contains a set of <Policy> or other <PolicySet> elements and a specified procedure for combining the results of their evaluation. It is the standard means for combining separate **policies** into a single combined **policy**.

XACML defines a number of Rule and Policy combining algorithms that define a procedure for arriving at an **authorization decision** given the individual results of evaluation of a set of **rules** or **policies**, in particular:

- Deny-overrides,
- Permit-overrides,
- First applicable,
- Only-one-applicable.

XAML Policies are based (or bound) to subject and resource attributes that are different from their identities. XAML allows multiple subjects and multi-valued attributes. XAML also allows policies based on resource content what means that authorisation decision may be based on content of the requested resource or its status.

Information security **policies** operate upon **attributes of subjects**, the **resource** and the **action** to be performed on the **resource** in order to arrive at an **authorization decision**. In the process of arriving at the **authorization decision**, **attributes** of many different types may have to be compared or computed. XACML includes a number of built-in functions and a method of adding non-standard functions. These functions may be nested to build arbitrarily complex expressions. This is achieved with the <Apply> element. The <Apply> element has an XML attribute called FunctionId that identifies the function to be applied to the contents of the element. Each standard function is defined for specific argument data-type combinations, and its return data-type is also specified.

Figures 4.2 and 4.3 shows the structure of Policy element and Rule element. Policy is bound to the Target that is described by Subject, Resource and Action. Policy may contain a number of rules defined by multiple Rule elements.

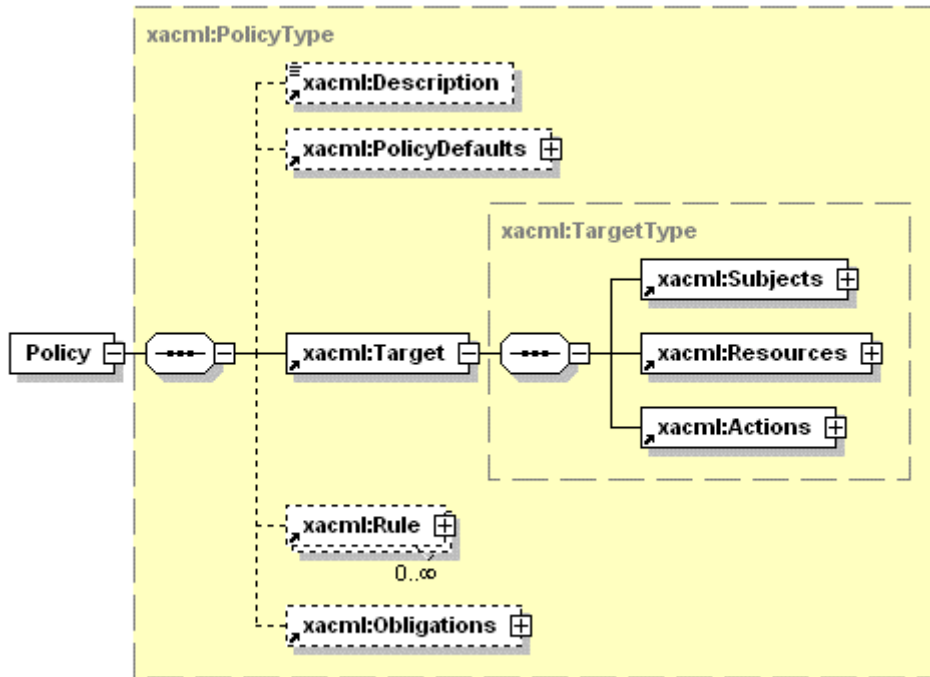


Fig. 4.2. Definition of the Policy element in XACML 1.0 that binds access rules to the Target (Subject, Resource, Action).

A rule is the most elementary unit of policy. The main components of a rule are target, condition that are represented by subelements and effect which is included as an attribute of the Rule element.

The <Condition> element is a boolean function over **subject, resource, action** and **environment attributes** or functions of **attributes**. If the <Condition> element evaluates to "True", then the enclosing <Rule> element is assigned its Effect value. The <Condition> element is of **ApplyType** complex type.

The <Apply> element denotes application of a function to its arguments, thus encoding a function call. The <Apply> element can be applied to any combination of <Apply>, <AttributeValue>, <SubjectAttributeDesignator>, <ResourceAttributeDesignator>, <ActionAttributeDesignator>, <EnvironmentAttributeDesignator> and <AttributeSelector> arguments.

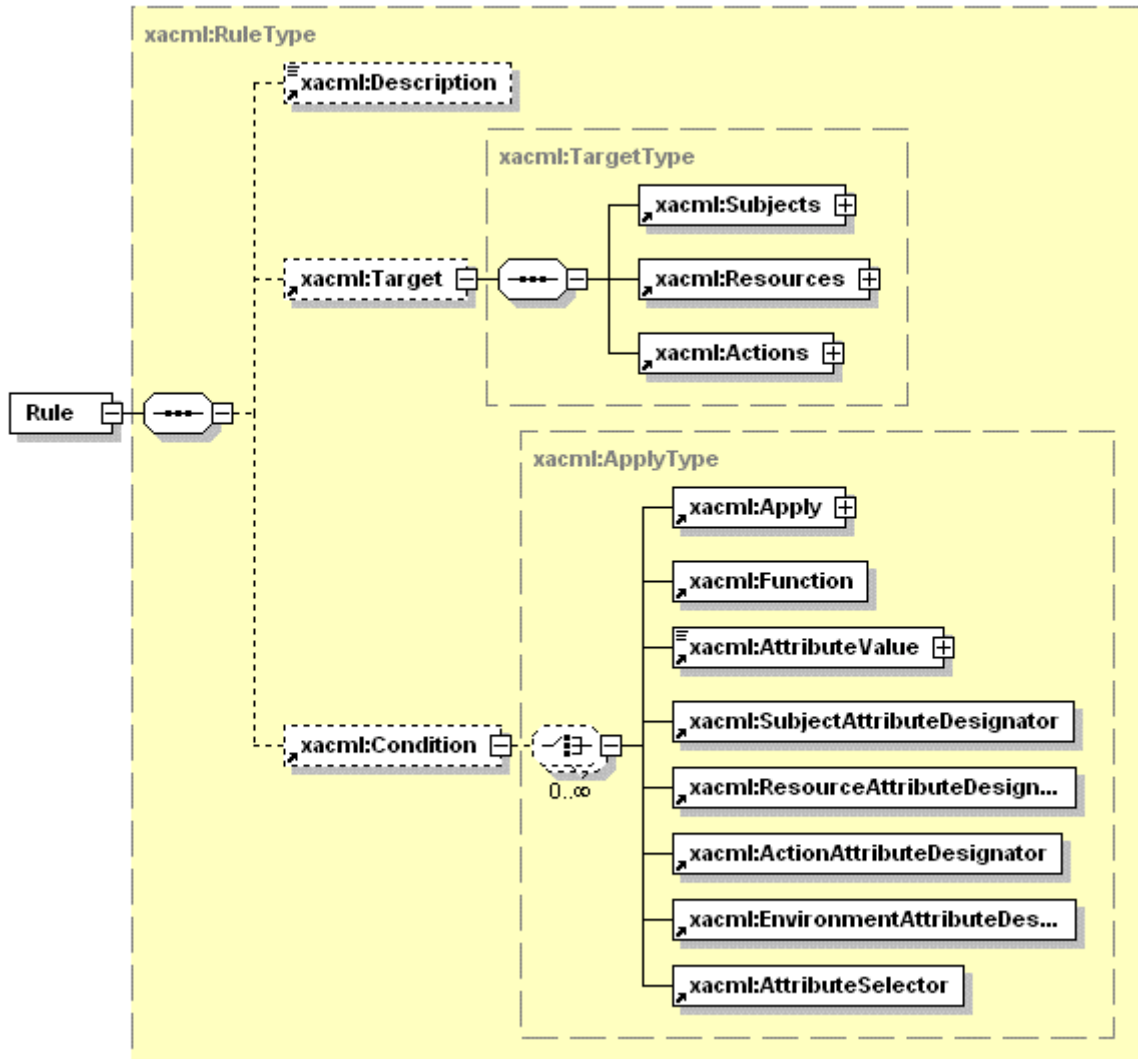


Fig. 4.3. Definition of the Rule element in XACML 1.0 that defines the access Conditions to the Target (Subject, Resource, Action).

XACML re-uses enumerated list of functions and operations defined in XPath 2.0 and XQuery 1.0 used in the `FunctionId` attribute of the `<Apply>/<Condition>` element. Element Target contains matching specification for the attributes of the Subject, Resource and Action.

The EGEE site Authorisation service will use standard XACML messaging format to ensure future compatibility with new and emerging products. XAML defines format for the Request message that provides context for the policy-based decision. Request may contain multiple Subject elements and multiple attributes of the Subject, Resource and Action.

The request message consists of three mandatory elements Subject, Resource, Action (so called Target triad Subject, Resource, Action), and optionally may contain the Environment element. The Subject element normally consists of Subject attributes, Subject authentication token and may contain subject ID sub-elements. The Resource element contains ResourceID sub-element that specifies the CNL resource or instrument, and may contain multiple ResourceAttribute sub-elements that may define resource subsystem or content related attribute. The Action element contains only one sub-element ActionID. It will be also possible to request multiple actions, however handling of such requests should be defined by the policy. The Environment element provides additional context

information for the Request and can be used for Requestor's policy reference in case of mutual Authorisation.

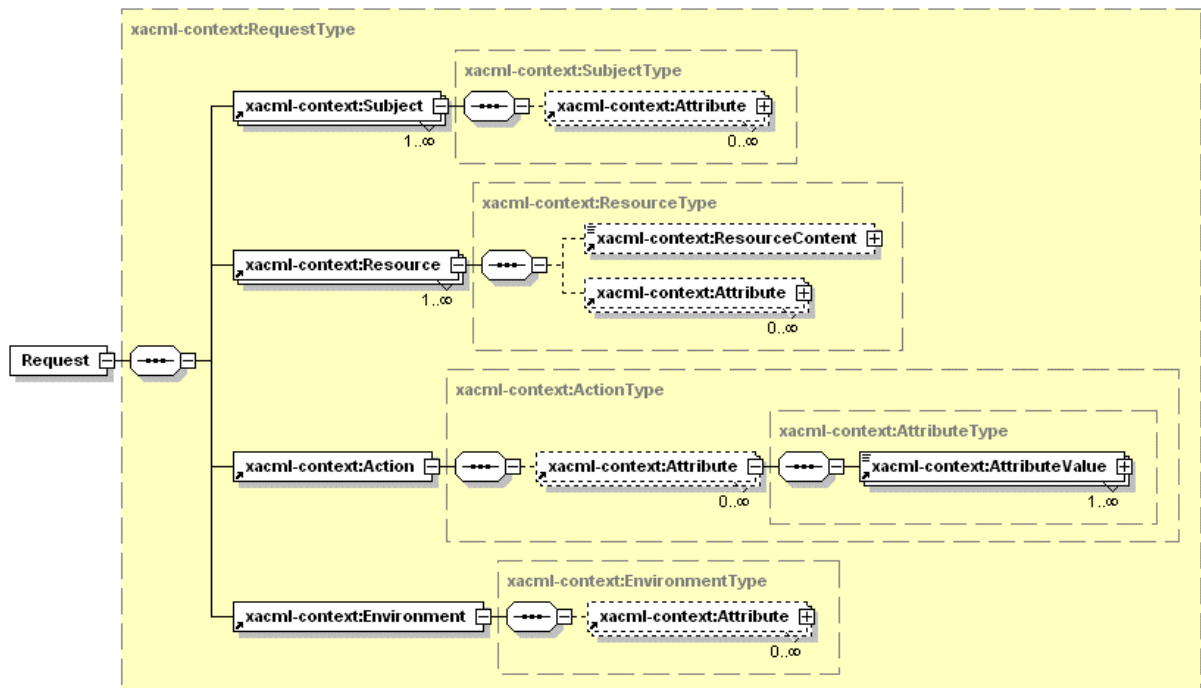


Fig. 4.4. High-level elements of the XACML 1.0 Request (XACML 2.0 Request is the same).

Response message defined by XACML provides format for conveying Decision (“Deny” or “Permit”) and Status of the decision making process. The Response message format may contain multiple Result elements as defined by the request message and resource policy. The Result element contains a Decision element, which may contain either “Permit” or “Deny” or “Intermediate”. The Status element may contain a simple status code (e.g., “OK”, “request-info”, etc.) and additional status information in the StatusMessage and StatusDetail sub-elements.

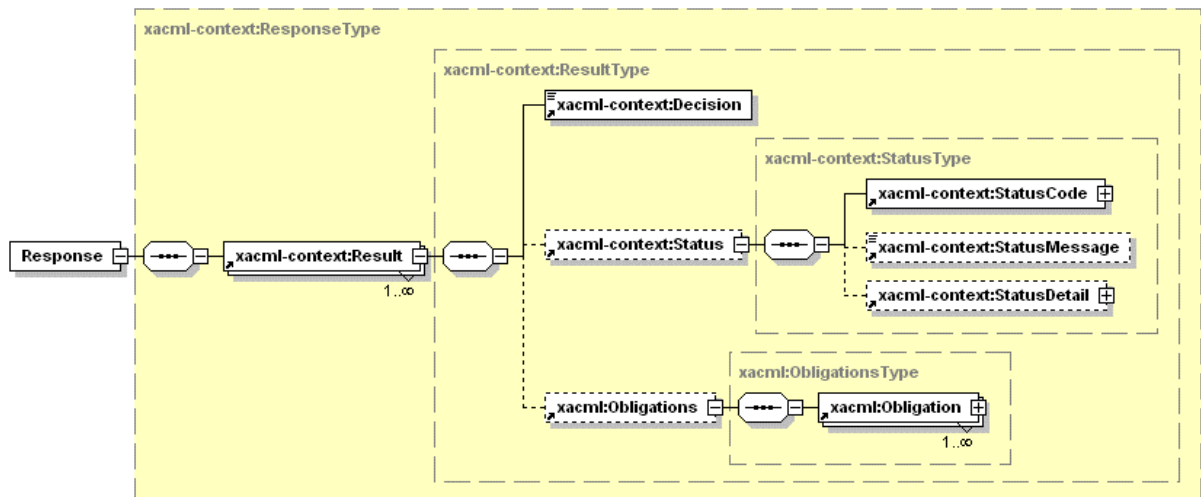


Fig. 4.5. High-level elements of the XACML 1.0 Response (XACML 2.0 Request is the same).

A request message sent by a user or an application over is an XML message sent with SOAP HTTP channel. Such a request contains information about the Requestor/Subject, Resource and requested Action. The response message contains a Decision result that may be either “Permit” or “Deny” for a final decision (or “Intermediate” for an intermediate communication).

Example below shows a XACML request message send to the PDP and requesting permission to perform ControllInstrument action on resource XPS1. Subject is represented by its SubjectID, authentication token and attributes role and JobID.

```
<?xml version="1.0" encoding="UTF-8"?>
<xacml-context:AAAResponse xmlns:xacml="urn:oasis:names:tc:xacml:1.0:policy"
xmlns:xacml-context="urn:oasis:names:tc:xacml:1.0:context"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="urn:oasis:names:tc:xacml:1.0:context data/schemas/aaa-cn1-msg-
xacml-01.xsd">
  <!-- CNL AAAResponse message contains triad (Subject, Resource, Action),
Environment is left as optional element for XACML compatibility -->
  <xacml-context:Subject Id="subject"
SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
    <!-- In CNL Subject = (SubjectId, Token, JobId, Role) are defined by
AttributeId designator -->
    <!-- Note: Policy will use Role and Token to evaluate permissions, SubjectID is
used for possible re-authentication or other purposes-->
    <xacml-context:Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
DataType="http://www.w3.org/2001/XMLSchema#string"
Issuer="admin@gaaa.collaboratory.nl">
      <xacml-context:AttributeValue>WHO740@users.collaboratory.nl</xacml-
context:AttributeValue>
    </xacml-context:Attribute>
    <xacml-context:Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:token"
DataType="http://www.w3.org/2001/XMLSchema#string"
Issuer="admin@gaaa.collaboratory.nl">
      <xacml-context:AttributeValue>2SeDFGVHYTY83ZXxEdsweOP8Iok)yGHxVfHom90</xacml-
context:AttributeValue>
    </xacml-context:Attribute>
    <xacml-context:Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:job-
id" DataType="http://www.w3.org/2001/XMLSchema#string"
Issuer="admin@gaaa.collaboratory.nl">
      <xacml-context:AttributeValue>JobID-XPS1-212</xacml-context:AttributeValue>
    </xacml-context:Attribute>
  <!-- Roles defined for CNL Demo2: Analyst, Customer, Guest, Admin -->
```

```

    <xacml-context:Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:subject:role"
DataType="http://www.w3.org/2001/XMLSchema#string"
Issuer="admin@gaaa.collaboratory.nl">
    <xacml-context:AttributeValue>Analyst</xacml-context:AttributeValue>
    </xacml-context:Attribute>
</xacml-context:Subject>
<!-- Policy will be defined for any of resources in CNL: Phillips XPS 1, Phillips
TEM 1, FEI TEM 1, CNL Job Management, CNL User Management, CNL DataStorage -->
    <xacml-context:Resource>
    <xacml-context:Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#string"
Issuer="admin@gaaa.collaboratory.nl">
    <xacml-context:AttributeValue>http://xps1.resources.collaboratory.nl/
Phillips_XPS1</xacml-context:AttributeValue>
    </xacml-context:Attribute>
</xacml-context:Resource>
<!-- Suggested list of action in CNL: ControlInstrument, ControlExperiment,
ViewArchive, ViewExperiment, AdminTask -->
    <xacml-context:Action>
    <xacml-context:Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:action:token"
DataType="http://www.w3.org/2001/XMLSchema#string"
Issuer="admin@gaaa.collaboratory.nl">
    <xacml-context:AttributeValue>ControlExperiment</xacml-
context:AttributeValue>
    </xacml-context:Attribute>
</xacml-context:Action>
    <xacml-context:Environment>
    <xacml-context:Attribute
AttributeId="urn:oasis:names:tc:xacml:1.0:environment:issue-time"
DataType="http://www.w3.org/2001/XMLSchema#string"
Issuer="admin@gaaa.collaboratory.nl">
    <xacml-context:AttributeValue>2001-12-17T10:30:00</xacml-
context:AttributeValue>
    </xacml-context:Attribute>
</xacml-context:Environment>
</xacml-context:AAAResponse>

```

Listing 4.1. Example XACML Request message

The following message is a PDP response with the result "Permit":

```

<?xml version="1.0" encoding="UTF-8"?>
<AAA:AAAResponse xmlns:AAA="http://www.AAA.org/ns/AAA_BoD"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.AAA.org/ns/AAA_BoD aaa-cn1-msg-xaml-01.xsd">
    <Result
ResourceId="http://resources.collaboratory.nl/http://resources.collaboratory.nl/Phi
llips_XPS1">
        <Status>
            <StatusCode Value="OK"/>
            <StatusDetail>DecisionID</StatusDetail>
            <StatusMessage>Request is successful</StatusMessage>
        </Status>
        <Decision>Permit</Decision>
    </Result>
</AAA:AAAResponse>

```

Listing 4.2. Example XACML Response message

4.3 XACML 2.0 special profiles

4.3.1 XACML 2.0 RBAC profile

XACML RBAC profile describes how to built Policies requiring multiple Subjects and roles combination to access a resource and perform an action. Multiple Subject elements in XACML allow flexibility when implementing hierarchical RBAC model for such cases when some actions require superior subject/role approval to perform a specific action. One or more `<Subject>` elements are allowed. A **subject** is an entity associated with the **access** request. For example, one **subject** might represent the human user that initiated the application from which the request was issued; another **subject** might represent the application's executable code responsible for creating the request; another **subject** might represent the machine on which the application was executing; and another **subject** might represent the entity that is to be the recipient of the **resource**.

4.3.2 XACML 2.0 Profile for Multiple Resources

The conditions under which multiple `<Resource>` elements are allowed are described in the XACML Profile for Multiple Resources. The hierarchical resource profile specifies how XACML can provide **access control** for a **resource** that is organized as a hierarchy, which examples include file systems, XML documents, and organizations. In this case resource is presented as set hierarchical nodes which are referred to as `resource-parent`, `resource-ancestor`, and `resource-ancestor-or-self`.

4.3.3 XACML Multiple Resources profile

XACML Multiple Resources profile SHALL be interpreted as a request for **access** to all **resources** specified in the individual `<Resource>` elements. For each `<Resource>` element, one **Individual Resource Request** SHALL be created. This **Individual Resource Request** SHALL be identical to the original request **context** with one exception: only the one `<Resource>` element SHALL be present. If such a `<Resource>` element contains a "scope" **attribute** having any value other than "Immediate", then the **Individual Resource Request** SHALL be further processed according to the corresponding enumerated value of this attribute. This processing may involve decomposing the one **Individual Resource Request** into other **Individual Resource Requests** before evaluation by the **PDP**.

5 References

- [1] Role Based Access Control (RBAC) – NIST, April 2003. - <http://csrc.nist.gov/rbac/>
- [2] ISO/IEC 10181-3:1996 Information technology -- Open Systems Interconnection -- Security frameworks for open systems: Access control framework. – Available in "OSG Authorisation API". - <http://www.opengroup.org/online-pubs?DOC=9690999199&FORM=PDF>
- [3] GFD-I.32 GGF Site requirements for Grid Authentication, Authorization and Accounting. – October 13, 2004 <http://www.gridforum.org/documents/GWD-I-E/GFD-I.032.txt>

- [4] Security Architecture for Open Collaborative Environment. By Yuri Demchenko, Leon Gommans, Cees de Laat, Bas Oudenaarde, Andrew Tokmakoff, Martin Snijders, Rene van Buuren. – Paper submitted to EGC2005.
- [5] eXtensible Access Control Markup Language (XACML) Version 1.1 - Committee Draft 01, 24 July 2003 - <http://www.oasis-open.org/committees/xacml/repository/cs-xacml-specification-1.1.pdf>
- [6] eXtensible Access Control Markup Language (XACML) Version 2.0 - Committee Draft 04, 6 December 2004 - http://docs.oasis-open.org/xacml/access_control-xacml-2_0-core-spec-cd-04.pdf
- [7] XACML profile for Web-services (WSPL): - <http://www.oasis-open.org/committees/download.php/3661/draft-xacml-wspl-04.pdf>
- [8] Web-services policy language use-cases and requirements: <http://www.oasis-open.org/committees/download.php/1608/wd-xacml-wspl-use-cases-04.pdf>
- [9] Hierarchical Resource profile of XACML. Committee Draft 01, 11 November 2004 - http://docs.oasis-open.org/xacml/access_control-xacml-2.0-hier_profile-spec-cd-01.pdf
- [10] Multiple Resource profile of XACML. Committee Draft 01, 11 November 2004 - http://docs.oasis-open.org/xacml/access_control-xacml-2.0-mult_profile-spec-cd-01.pdf
- [11] Core and Hierarchical Role Based Access Control (RBAC) profile of XACML, Version 2.0. Committee Draft 01, 11 November 2004 - http://docs.oasis-open.org/xacml/access_control-xacml-2.0-rbac_profile1-spec-cd-01.pdf
- [12] Security Assertion Markup Language (SAML) v1.0 - OASIS Standard, 5 November 2002 - <http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf>
- [13] Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. Committee Draft 03, 14 December 2004 - <http://www.oasis-open.org/committees/download.php/10627/sstc-saml-core-2.0-cd-03.pdf>
- [14] SAML 2.0 profile of XACML. Draft 02, 11 November 2004. - http://docs.oasis-open.org/xacml/access_control-xacml-2.0-saml_profile-spec-cd-02.pdf
- [15] Web Services Policy Framework (WS-Policy). Version 1.1. - <http://msdn.microsoft.com/ws/2002/12/Policy/>
- [16] Web Services Policy Attachment (WS-PolicyAttachment). Version 1.1. - - <http://msdn.microsoft.com/ws/2002/12/PolicyAttachment/>
- [17] Web Services Federation Language (WS-Federation) Version 1.0 - July 8 2003 – <http://msdn.microsoft.com/ws/2003/07/ws-federation/>
- [18] Liberty Alliance Phase 2 Final Specifications - <http://www.projectliberty.org/specs/>
- [19] Web Services Secure Conversation Language (WS-SecureConversation) - <http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-secureconversation.asp>