

CNL2 Detailed Security design: Authorisation and Policy Enforcement¹

Yuri Demchenko <demch@science.uva.nl>

Abstracts

This technical design document provides all background technical information for building Collaboratory.nl Project Phase 2 (CNL2) Authorisation service as a component of the overall CNL2 Security infrastructure. The document describes CNL Job-centric security model as overall framework for building customer driven security infrastructure/services and provides technical details about its demo implementation, in particular, CNL Authorisation token definition and its mapping to both used in CNL XACML messages format and SAML assertions format.

Proposed CNL2 Authorisation and Policy enforcement service/infrastructure is based on and provides further development to the Generic AAA framework for flexible/extended policy and security context management, in particular for the generic Role-based Access Control (RBAC) functionality. Implementation suggestions and details are provided for the three basic AAA pull, push and agent models that can be used in different CNL use cases. The document also explores the possibilities of the dynamic policy binding using WS-Security and WS-PolicyAttachment mechanisms to provide customer-driven/job-centric security model being developed in CNL project. Initial suggestions are provided for multiple policies combination by using Policy Enforcement Points (PEP) and/or Policy Decision Points (PDP) sequencing or recursive policy evaluation, and for the mutual authorisation required in cases when information confidentiality and privacy is of high concern.

The document also undertakes an attempt to analyse trust relations in distributed access control infrastructure in application to the proposed OCE/CNL Job-centric security model. The suggested outcome of this analysis will be recommendations for the policy management and key management and distribution.

Proposed solutions are built upon extended use of existing and emerging standards in area of XML and Web Services security, in particular, WS-Security stack, SAML, XACML, XML Signature and XML Encryption and can provide a good technical basis for building security infrastructure for Open Collaborative Environment (OCE) in general.

The document also describes proposed formats of proprietary CNL Authorisation ticket and token and their mapping to SAML AuthzAssertion and XACML messages format. Additionally, similar information provided for CNL Authentication ticket and token.

¹ Copyright notice: This is a draft version of the document and it is provided for public review and discussion to benefit from solicited comments. Not all content will be used in the final report on CNL. No part of this document can be copied or used as a part of other document or research without notification and prio consent of author. Some parts may require also obtaining permission from the CNL Consortium.

1 CNL Job datamodel and Job-centric security model	2
2 CNL/OCE Security system operation	3
3 Open standards for Authorisation and Policy expression	5
3.1 Introduction and roadmap	5
3.2 Relation to standards in access control and policy expression	6
3.2.1 Policy Expression and Exchange Layer	7
3.2.2 Authorisation and Access Control	7
4 Policy based access control using Generic AAA framework	8
4.1 Generic RBAC functionality	8
4.2 Access Control System operation and components	10
The following implementation suggestions should be considered:	13
4.3 Mutual authorisation model	14
4.4 Policy enforcement in Service Oriented Architecture using WS-Security/WS-Policy mechanisms	15
4.5 Trust relations and Policy management in distributed AAA infrastructure	18
5 CNL Authorisation Service operation in a Demo system	21
6 Using Authorisation tickets and tokens for performance optimisation in the CNL/OCE Authorisation infrastructure	24
6.1 CNL AuthzTicket definition and its mapping to XACML messages and SAML assertions formats	24
6.2 CNL Authorisation tickets and tokens examples	26
7 Using XACML and SAML for AuthZ decision request and security tokens exchange	32
8 Providing Integrity and Confidentiality with XML Digital Signature and XML Encryption	35
8.1 XML Digital Signature format and processing	35
8.2 XML Encryption	37
9 References	39

1 CNL Job datamodel and Job-centric security model

The Job description, as a semantic document, is created based on the signed order and contains all information required to run the experiment on the collaborative infrastructure. The job description contains the following components: Job ID, Job owner and other attributes, assigned users and roles, Policy reference, trust/security anchor (TA) in a form of customer and provider digital signatures. Figure 1 illustrate a structure of the Job description and its relation to other CNL/OCE components and security services.

This kind of job description can also be used as a foundation for creating Virtual Organisation (VO) as an association of designated users and resources, which support all standard security constructs such as users, groups, roles, trust domains, designated services and authorities.

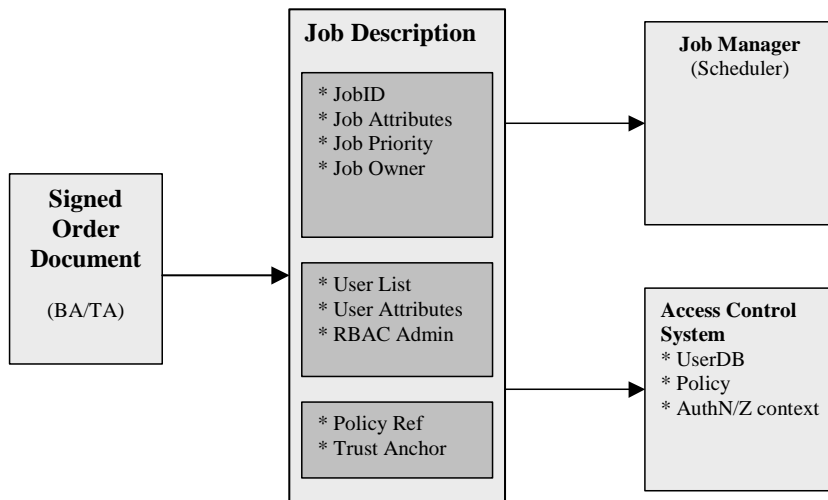


Figure 1. Security built around Job description

The job description (mandatory) must include or reference the Job policy, which defines all aspects of the user, resource and trust management when executing the job. This policy should define the following issues:

- trusted users, VO's, resources and in general trusted credentials (or trusted CA's);
- delegation policy and additionally identity federation/mapping policy;
- privileges/permissions assigned to roles;
- credit limits and conditions of use;
- confidentiality and privacy requirements;
- Job access control or authorisation policy.

It is important to note that a Job policy may be combined with the Resource admission policy and in practice should not be more restrictive than the Resource policy. Otherwise the Job security management service may reject some resources based on Resource policy evaluation as a procedure of mutual authorisation.

Job-centric approach gives organizations complete flexibility in the creation of their security associations and services for the specific tasks or applications.

Practical implementation of the Job-centric security model requires wide spectrum of emerging XML and Web Services Security technologies that altogether constitute a general OCE Security Architecture as described in the next chapter.

2 CNL/OCE Security system operation

Figure 2 below illustrates relations and interactions between major entities and processes in the Job-centric model. The access control system/infrastructure implements the generic RBAC model which operation is defined by the specific attributes of a particular job. All initial information required for proper operation of the AuthN/AuthZ system should be provided in a job description (JobDescr) that binds job attributes, user information and established security/trust relations between the customer

and the provider. This approach to building security services in CNL/OCE may be defined as a job-centric, and it provides the perceived benefit of decoupling security services and simplifying building the distributed security infrastructure.

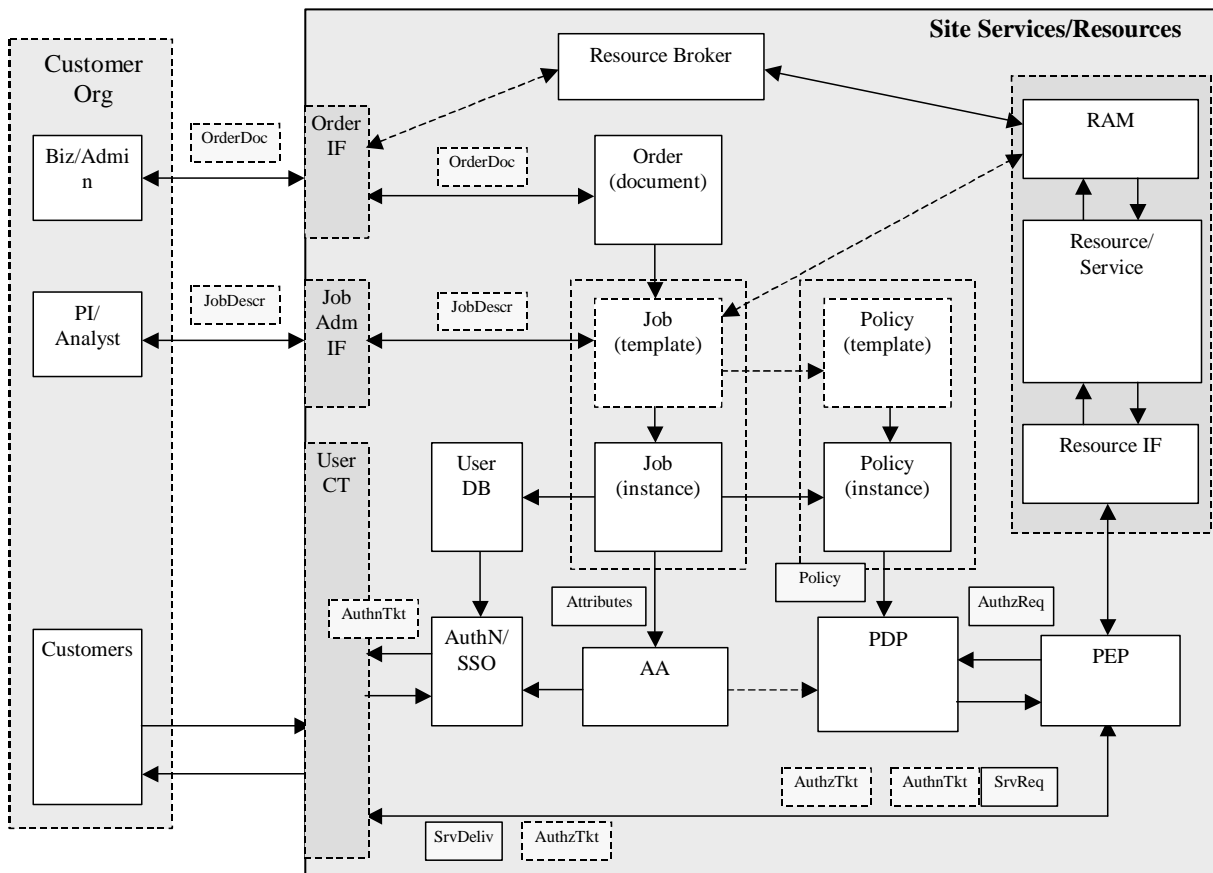


Figure 2. Major entities and processes in the Job-centric model

JobDescr is created as a result of customer and provider negotiation and an agreement that can provide the so-called business and/or trust anchor (BA/TA). During operation, security services will:

- 1) retrieve user information and roles from the JobDescr and put them into the UserDB;
- 2) retrieve job attributes to reference or define the policy of the resource access;
- 3) use TA or BA to verify or sign all future security (or financial) related attributes, claims, tokens and credentials.

Interaction with the user is provided via User Collaborative Tools (UserCT), interaction with the instrument – via Resource Agent. User authentication is requested by UserCT, user authorisation is enforced by the Resource Agent. UserCT and authentication services may include SSO (Single-Sign-On) functionality to provide a single user logon for a particular domain, defined by a business or trust agreement.

The job/order owner may possess the RBAC administrative functions (privileges) that allow him/her to create and modify user accounts and assign roles/privileges for a particular job/experiment.

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.

Currently, CNL uses the basic Authentication (AuthN) service provided by CHEF GroupWare. In the next stage, the project will investigate the use of a user AuthN and identity and credentials management solution that should allow dynamic creation of user and resource associations on the basis of the Virtual Organisations (VO) concept (currently being developed in the general OGSA framework).

3 Open standards for Authorisation and Policy expression

3.1 Introduction and roadmap

This section provides general vision and roadmap how authorisation service can be built using existing emerging standards and technologies in areas of XML and Web Services Security, Privilege Management Infrastructure (PMI) and Role-based Access Control (RBAC).

Common authorisation framework will improve services integration and security. Authorization decisions may include multiple layers and multiple (policy) decision points however coordination by common site/resource policy.

Common AuthZ service will be based on common policy expression and exchange format and common Request/Response protocol that allow intra-site and multiple site scalability. It will use the principle of ownership in respect to the policy and decision making precedence. Policy based access control assurance will be provided by attaching policy to the service description, in particular, in a form of WSDL.

Future/developing AuthZ service will separate Authorisation infrastructure from the policy itself providing only secure environment/mechanism for site/authority controlled policy enforcement. This will simplify policy management and provide more generic infrastructure for services and resources virtualization.

Service request evaluation will be done according to the site/resource access control policy for the major target components Subject, Resource, Action, and additionally Environment.

The local authorization will take this into account by implementing the policy evaluation engine as a separate service that will be able to call external separate decision points. Thus, a local to a Resource (designated) PDP can call other PDPs requesting for evaluating policy components related to their domain of authority to provide a final decision. In this respect the final decision will always reside with the resource owner. The site comprising of a number of resources and/or services normally establishes a common authorisation policy against which all site access requests are evaluated by site's central AuthZ service. It is perceived that AuthZ and/or site access policy are known to all participants of the service request/consuming process and related AuthZ service. The

policy can be distributed off-band or exchanged dynamically during negotiation between service consumer and service provider.

However, in less administratively centralized (less coupled) environment, the resource may have a locally determined policy that may imply additional restrictions on resource usage and/or access. This locally implied policy may be referenced in the (common) site access policy as a resource or environment component evaluated by requesting related information from the resource during the central policy evaluation or applied by the resource locally using local PEP and PDP functions. The examples of such local resources policies may include resource "blacklist", limitation for number of users, etc. However, such general user related requirements as for example user qualification/training level (like in case of complex experimental or medical equipment, their use can only be allowed to certified professionals) should rather be included into the general site access policy. A question how to expose local resource policy to the initial requestor remains open.

3.2 Relation to standards in access control and policy expression

Developing CNL/OCE Authorisation framework is intended to be compatible with the general WSA/OGSA Security Architecture and in particular with the EGEE Global Security Architecture (GSA) [5, 6, 7]. The GGF document GFD-I.032 "Site requirements for Grid Authentication, Authorization and Accounting" is used as a general guidance how to address Grid sites specific security requirements [8].

Recently developed and emerging standards in XML Web Services Security provide a good basis for building flexible scalable AuthZ and policy enforcement infrastructure that implements effective easily manageable Privilege Management Infrastructure (PMI) [9] and Role-based Access Control (RBAC) [10] models.

PMI and RBAC separate authentication and authorisation functions giving the home organisation (or identity provider in other architectures) to authenticate a user and the resource or target to make an authorisation decision. Main components of the distributed AuthZ architecture include Authentication (AuthN) function, Access Enforcement Function (AEF) or Policy Enforcement Point (PEP), Access Decision Function (ADF) or Policy Decision (PDP), and Access Control Information (ACI) or Policy module.

Conceptually the Authorisation service is provided by the resource. The authorisation decision is taken according to the access policy based on provided user identity and attribute credentials. All or part of authorisation context information can be provided by the requestor (push model) or requested by the Authorisation service (pull model). Accordingly, ADF/PDP modules can also work in push or pull modes.

To be scalable and manageable, AuthZ infrastructure needs to separate all functional components of the AuthZ decision making and policy enforcement process from each other and from the resource/site services and re-arrange them by using standard/classical PMI and RBAC models/architectures. In general, AuthZ service should be built as a resource/site independent service that is cryptographically/semantically bound to a site or resource, i.e. trusted by the resource to enforce its access control policy.

3.2.1 Policy Expression and Exchange Layer

Communicating services/principals need to conform to certain requirements in order to interact securely. It is important that service or resource requestors have access and understand policies associated with the target service. As a result, both the service requestor and service provider must select an acceptable security profile. It is also important to mention that the privilege to acquire security policy is given by the hosting environment to authenticated and authorised entities only.

The policy layer provides necessary information for the policy enforcement modules of the Operational Security layer. It is suggested that policy expression should conform to the general WS-Policy framework and may include component policies using different policy expression formats including Generic AAA [12] or XACML [13] policy formats. Policies for end applications and services are described by (a set of) rules for a specific target comprising of a triad (Subject, Resource, Action). Policy may also include common attributes such as security tokens, delegation and trust chain requirements, policy combination algorithms in a form defined by WS-Security and WS-Policy [14, 15].

Policy association with the service can be done using WS-PolicyAttachment that defines extensions mechanisms to attach a policy or policy reference to different components of the service description including PortType, operation/message, or arbitrary element [16]. XACML Web Services profile defines the mapping between Web Service description components and its own policy description hierarchy [17].

3.2.2 Authorisation and Access Control

The Authorisation and Access Control security service is a key part of the managed security in an open service oriented environment. Authorisation is typically associated with a service provider or resource owner. The resource owner controls access to a resource based upon requestor credentials or attributes that define the requestor's privileges or associated roles bound to the requestor's identity. Separation of Authentication and Authorisation services allows dynamic role-based access control (RBAC) management [5] and virtual association between interacting entities, and provides a basis for privacy.

For distributed multi-domain services/applications, an Authorisation service can operate in pull or push modes as defined by the generic AAA architecture [3.4] in which correspondently Authorisation decision is requested by a Resource service, or requestor preliminary obtains the Authorisation token from the trusted Authorisation service. It subsequently presents the token together with the authorisation context to the resource or service. When using the push or the combined pull-push model for complex services requests, the SAML format is considered as a recommended format for security tokens exchange.

4 Policy based access control using Generic AAA framework

This section provides a short description of used technologies and implementation details to illustrate how generic security components can be used for providing basic access control functionality in CNL/OCE.

CNL/OCE should provide basic security services: authentication and identity management, authorisation, information and data confidentiality and integrity, non-repudiation and privacy. Typical CNL/OCE use cases apply some specifics of the collaborative environment that is in general dynamic (changing from experiment to experiment) and may span multiple trust domains what will require to handle different user identities and attributes. For this purpose CNL/OCE security infrastructure should provide Single-Sign-On (SSO) and more general Identity management functionality. Authorisation or access control system uses policy and role based access control approach what will allow for flexible user access management.

Security services may be bound to and requested from any basic CNL/OCE service using a standard request/response format. Security services use must be specified by the policy that provides a mapping between a request context (e.g., action requested by a particular subject on a particular resource) and resource functionality and permissions.

Binding between (basic) services and security services can be defined dynamically at the moment of service deployment or invocation using existing Web services and XML Security technologies for binding/associating security services and policies to the service description as explained above.

4.1 Generic RBAC functionality

Figure 3 illustrates the abstract RBAC model as it is implemented in XACML policy enforcement framework and used in CNL. In general, the CNL Authorisation infrastructure consists of

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

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

In CNL PIP and RIP are implemented as Application Specific Modules (ASM) using protocol and exchange format conventions defined by the Generic AAA framework to provide interface to external services.

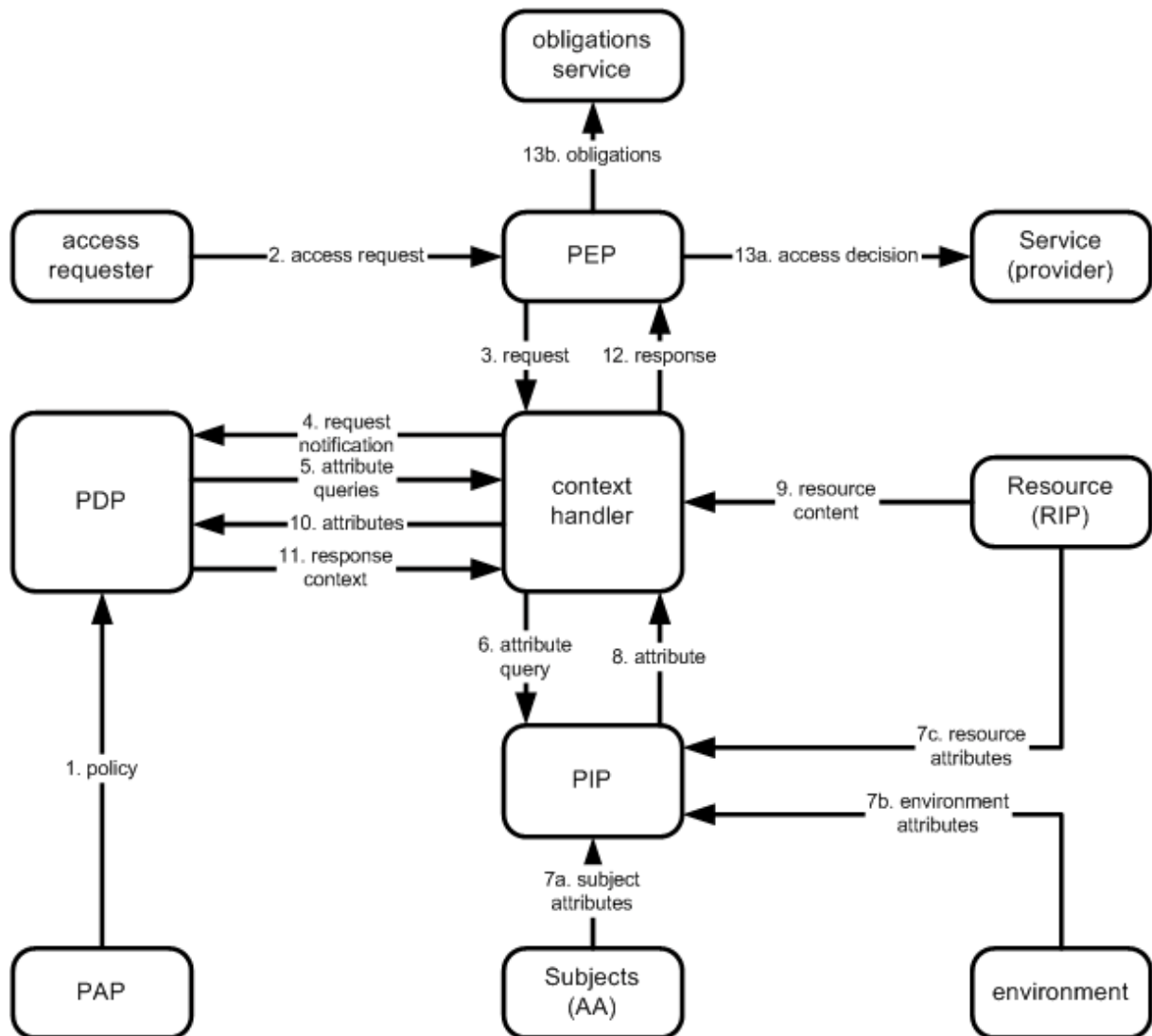


Figure 3. RBAC based access control system components and dataflows.

During the policy evaluation, the RBE invokes external applications via the ASMs. The ASM translate the semantics of the application to some logical /primitive needed by the RBE to make a decision. A user or application sends an AAA request message. This request message 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).

In details, the 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.

The CNL Authorisation service uses standard XACML messaging format to ensure future compatibility with new and emerging products.

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 consists of SubjectID, Role, JobID and Token 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.

The AAA 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.

4.2 Access Control System operation and components

This section explains how the AuthZ components can be placed between a Requestor and a Resource in a more general but more technical form than it is explained in the previous section. Two

basic use cases/models are discussed in this section: combined pull-push model and combined agent-push model.

Figure 4 below illustrates a typical RBAC authorisation model that implements pull model of the generic AAA Authorisation framework and may also use the authorisation ticket “push” functionality to optimise performance. The picture also explains how the policy combination can be done via PEP chaining/sequencing and/or PDP nesting/recursion.

Provided more detailed policy enforcement process analysis allows to formulate security constrains for a typical use case of multiple policies evaluation and multiple PEP’s and PDP’s combination when intending to preserve a site or a resource access control integrity.

This analysis is also undertaken to understand implementation details of the proposed/introduced in OGSA/GT4 and EGEE GSA documents the Authorisation framework that uses configurable PEP/PDP/PIP chaining [OGSA-SEC, GT4 Az, EGEE GSA].

Proposed approach retains integrity of the single/combined policy based decision. Thus, the PDP when evaluating a request from the PEP can call for external evaluation of some policy components but makes its own final decision and returns it to a calling PEP, which acts as a gateway to the initial request.

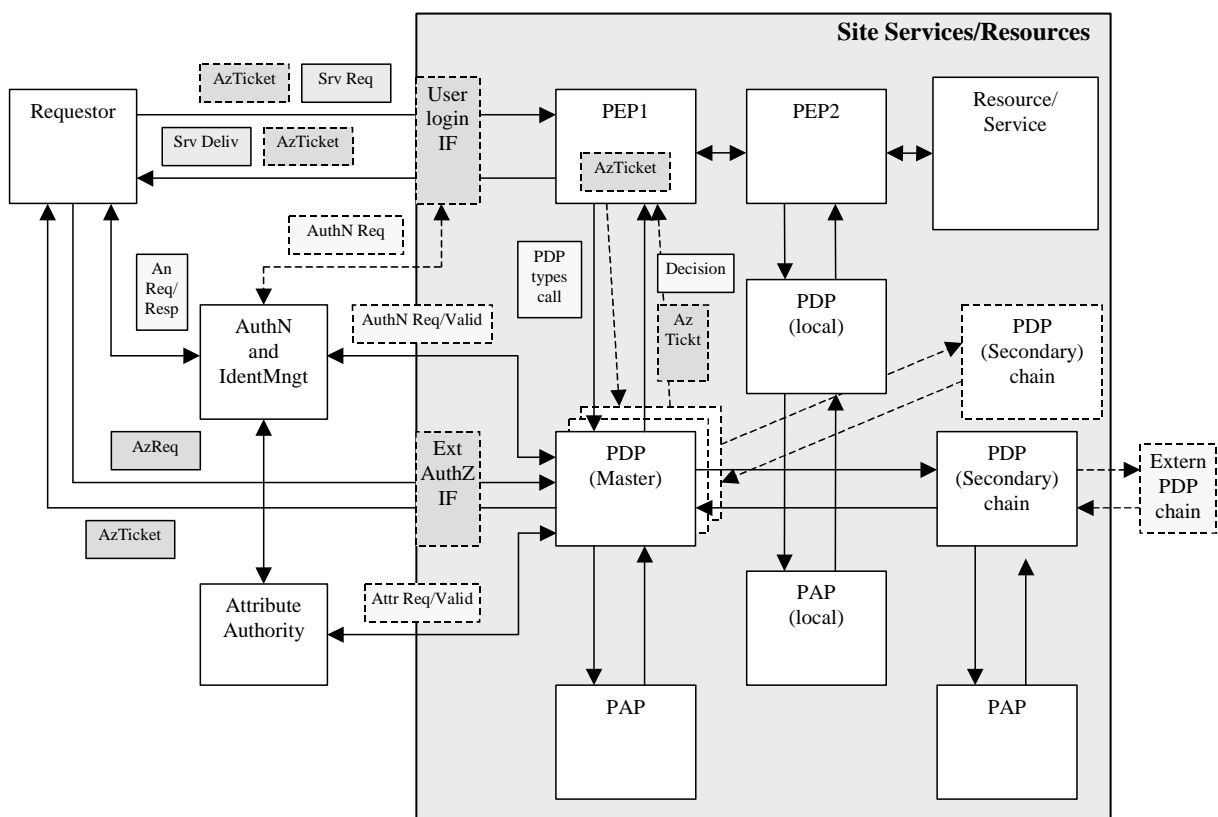


Figure 4. Major components of the site Authorisation service (RBAC and combined pull-push model)

The Requestor requests a service by sending a service request **SrvReq** to the Resource’s PEP providing as much or as little information about Subject/Requestor, Resource, Action, and

additionally Environment as it decides necessary according to used authorisation model and (known to the Requestor) local policies.

In a simple scenario, the PEP sends the decision request to the (designated) PDP and after receiving a permission from PDP relays a service request to the Resource. The PDP identifies the applicable policy instance, retrieves required context information and evaluates the request against the policy. During this process it may need to validate the presented credentials locally based on pre-established/shared trust relations, or call external Authentication and Attribute Authorities.

Described above process represents the basic scenario. However, in more complex and open environment the PEP may receive requests that have different formats and semantics (namespaces) and refer to different policies and respectfully to different policy repositories. In this case PEP should have a possibility to rely a decision request to a proper PDP type capable to handle a whole decision request. It is essential that a request is evaluated in whole and a decision is made by a single PDP, which however can make calls to external PDP's to evaluate some request components and process their decisions as components of the general policy evaluation process. The PDP that makes a final combined decision can be defined as a master PDP and it needs to have mechanisms in place to preserve integrity of the final combined decision.

Existing (open) policy expression formats, e.g. XACML, AAA, provides mechanisms for the particular policy instance to refer to another policy instance. Complex/combined policy can be created by PAP on PDP policy request, or processed by the PDP by requesting required policy components during the request evaluation.

As a trade-off of being open by using separate access control components and open standards, the solution above has known performance concerns. The resolution of this problem is seen in combining pull and push operation models. Since the decision is made by the PDP, the AuthZ ticket can be issued and used in the next similar or repetitive actions requests during the ticket's validity period. AuthZ ticket can be obtained via PEP during the first access request or request directly from the PDP via external AuthZ interface prio to sending a service request.

In the push model the Request first request Authorisation decision and obtains AuthZ ticket, which it will attach to any service request. The PEP will evaluate the validity of the presented tickets and some other security credentials that prove the ticket ownership and trustworthiness. However, no other access decision functions should be given to the PEP as a functional component. If there is a need to enforce other components of the site or resource control, like "blacklist", it should be done via separate (local) PEP-PDP chain.

Figure 5 below illustrates another access control model that combines two other generic AAA Authorisation models: agent model and push model. This model represents a typical Bandwidth-on-Demand (BoD) use case and a general complex service on-demand provisioning use case.

Constructed of the same building blocks, this access control system operates similar to the previously discussed implementation with the difference described below.

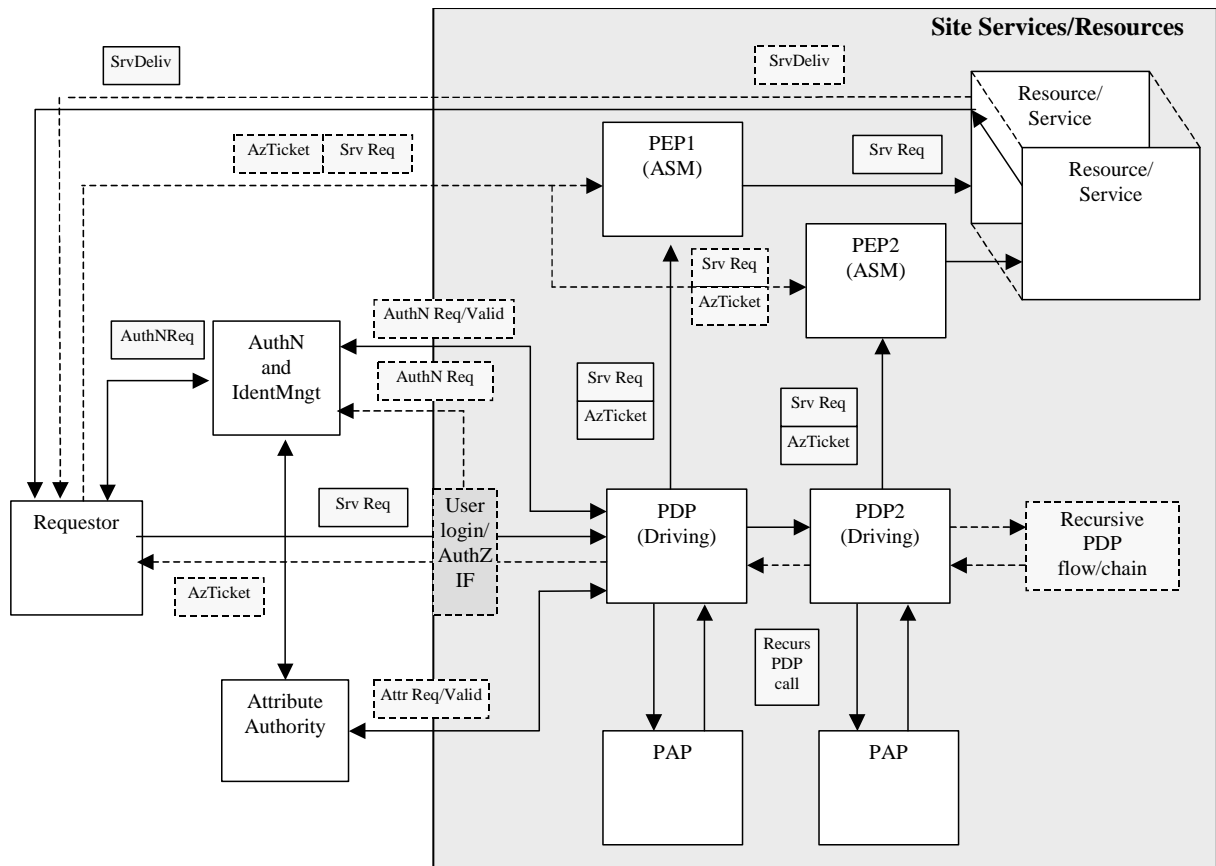


Figure 5. Major components of the site Authorisation service (combined push and agent model, controlling complex/multiple components resource)

In the Agent model the PDP as a component of the AAA system initiates a (complex) service delivery to the Requestor. The policy in this case can be defined as a driving policy and may have elements of the workflow management. In case of complex resource/service request, a sequence of PDP's creates a recursive policy evaluation flow/chain that may use a set of PEP modules to enforce policy for difference resources/services. It is assumed that each PDP can request other PDP's for evaluating some of the policy components for the specific resource.

To be added.

The following implementation suggestions should be considered:

Described above scenarios are simple ones, but they require that both Requestor and the Resource services know explicitly or implicitly the policy, semantics and know or can access the context information. More detailed:

1. PDP and PAP must share common namespace
2. Policy and respectively PAP should be referenced in the request message explicitly or known to PEP and PDP a priori
3. Every PEP in the chain of policy enforcement should take care of the whole request evaluation/enforcement by calling to a single (master) PDP. PEP should not do multiple decision combination.

4. Only one PDP should provide a final decision on the whole request
5. However, PEP may have a possibility to request different PDP types based on request semantics/namespace and referred policy
6. It is suggested that in general and to have a possibility to combine pull and push AuthZ models for the performance optimisation purposes the PEP should understand and have a possibility to validate the AuthZ ticket issued by trusted PDP or AuthZ system in general.
7. For this purpose the Requestor may request, PDP may issue and PEP can relay the AuthZ ticket back to the Requestor. The AuthZ issued by the PDP should have validity and usage restriction and contain information about the decision and the resource.
8. For future validation of the AuthZ tickets, the PEP may cache the ticket locally to speed-up the validation procedure.
9. When using AuthZ tokens that uniquely reference AuthZ tickets but are smaller and simpler, AuthZ tickets should be cached by PEP for future resolution by tokens.

4.3 Mutual authorisation model

Mutual authorisation of the requestor and the resource/service is one of the requirement in the customer centric WSA/SOA applications.

In general, mutual authorisation can be achieved by adding authorisation function to a requestor in respect to a resource or service provider during the negotiation period. This scenario also presumes that a requestor can authorise a resource based on applied policy and issue an Authorisation ticket (AuthzTicket) to a resource. The need for the AuthzTicket in mutual authorisation scenario is motivated by the fact that requested service may require a significant time for processing and may span beyond the requestor service instance lifetime. It is suggested that after submitting a request or task, a requestor may not remain in active state until receiving requested services or result.

Figure 5 below possible operation of the access control system components during mutual authorisation.

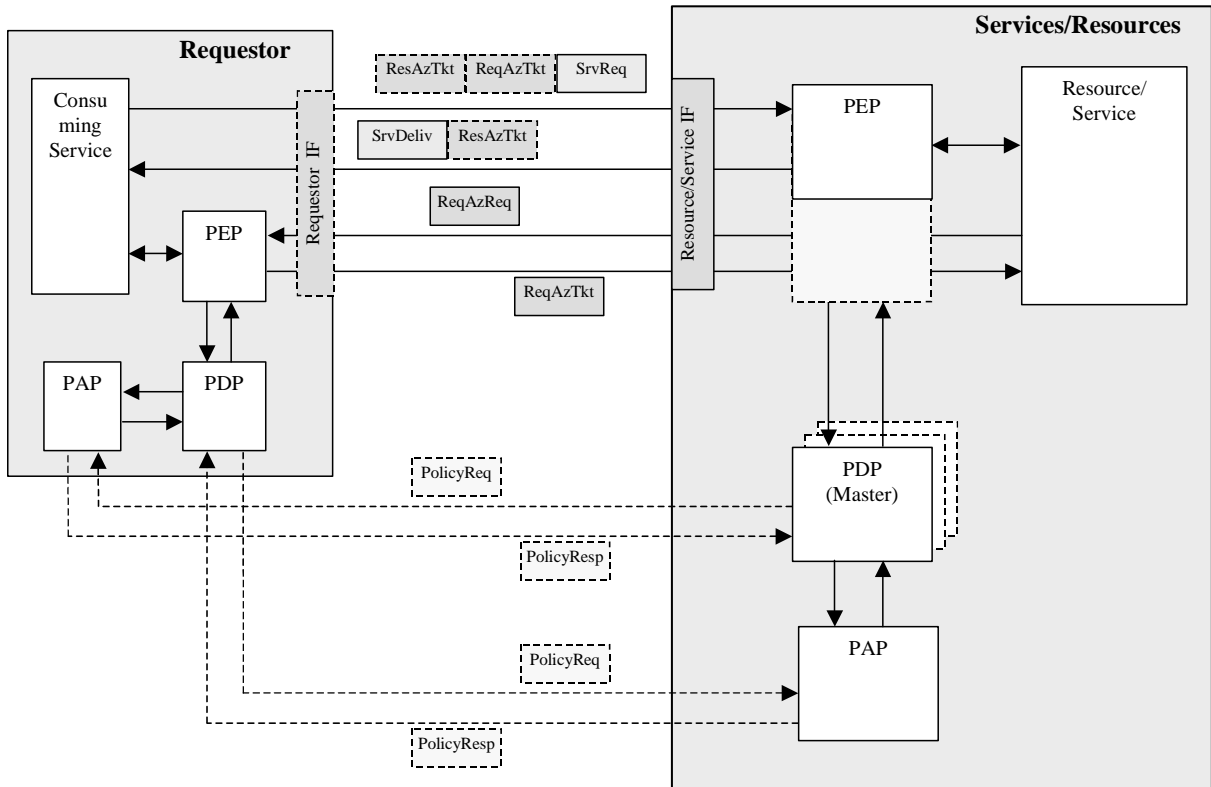


Figure 6. Mutual authorisation operational model.

4.4 Policy enforcement in Service Oriented Architecture using WS-Security/WS-Policy mechanisms

Described above rather traditional (client/server) implementations of the access control system uses a number of presumptions:

- 1) policy is known a priori to all participating parties:
 - Requestor knows what information to present to adhere to a specific policy and in what format, although a Requestor can reference a specific PEP-PDP modules (what may be also required in some PEP-PDP implementations);
 - Policy has been exchanged between all participating parties before sending requests
- 2) PEP knows which PDP it should request to evaluate presented information, and knows a *format* and required *data* to create a request to PDP

Note. Actually in the previously discussed use cases a simple Requestor can send a request in some simple proprietary format and the PEP will take care about representing the request in a standardised format to match a PDP implementation and used policy format. XACML policy format and XACML messaging can be a good choice of the standard PEP-PDP messaging format.
- 3) Resource trusts PDP's decision that can be delivered to a Resource in a form of AuthzTicket or based on default trust between PEP and Resource

- 4) Generally, PDP may discover which AuthN services and which Attribute Authority (AA) it should request to validate Requestor's/Subject's AuthN information and presented attributes by checking credentials' issuer reference, but it may be a need to know it a priori also.

Described above approach requires off-band policy distribution and prior to the security services initiation.

Attributes authority and attribute semantics are another components of the GAAA infrastructure that need to be setup before the service operation. Again, traditional practice presumes that AA and attributes semantics are known, and trust relations between PDP, AA and the resource are established.

Root of policy enforcement hierarchy, like in real life, belongs to the resource owner or organisation "contracted" to protect the resource or resource association.

However, in open WSA/SOA based systems there is a need to provide a mechanism to dynamically link all components of the discussed access control system. This can be done by binding/attaching the policy definition or its reference to the service description. In this case policy context will appear in a following way:

1. the central point of the policy attachment is in the service description in a form of WSDL file, which contains definition of portTypes, available services and messages format. Attaching policy to WSDL means that the policy reference can be added to any of the WSDL element.
2. interacting services will fetch policy document and apply restrictions/rules to the elements, which declared policy compliance requirement; this may apply for both service request and response or service delivery.

WS-PolicyAttachment defines two general-purpose mechanisms for associating Policies with one or more Policy Subjects.

The first one allows XML or WSDL based description of resource (in particular, portType, Operation, Message) to associate Policy as part of their definition [WS-PolicyAttachment] by using global attribute `wsp:PolicyURIs` or child element `wsp:PolicyReference` that reference Policy expression:

```
<xs:schema>  
  <xs:attribute name="PolicyURIs" type="wsp:tPolicyURIs" />  
</xs:schema>
```

or

```
<wsp:PolicyReference URI="URI" />
```

(Note: definition of WS-PolicyAttachment mechanisms semantics has changed in the new published standard version of September 2004.)

The `wsp:PolicyURIs` attribute contains a white space separated list of one or more URIs. There may be few children `wsp:PolicyReference`. If more than one URI is specified, the individual referenced Policies need to be merged together to form a single Element Policy Expression. The resultant Policy is then associated with the element information item's Element Policy property.

The second mechanism allows Policies to be associated with a Policy Subject independently of that subject's definition and/or representation through the use of `<wsp:PolicyAttachment>` element. The following is the pseudo-schema for the `<wsp:PolicyAttachment>` element:

```
<wsp:PolicyAttachment ... >
  <wsp:AppliesTo>
    <x:DomainExpression/> +
  </wsp:AppliesTo>
  ( <wsp:Policy>...</wsp:Policy> |
    <wsp:PolicyReference>...</wsp:PolicyReference> ) +
  <wsse:Security>...</wsse:Security> ?
  ...
</wsp:PolicyAttachment>
```

This element has three components: the Policy Scope of the attachment (`<wsp:AppliesTo>` element), the Policy Expressions being bound (`<wsp:Policy>` or `<wsp:PolicyReference>` elements), and optional security information (`<wsse:Security>`). The Policy Scope of the attachment is defined using one or more extensible domain expressions that identify Policy Subjects, typically using URIs.

Domain expressions identify the domain of the association. That is, the set of Policy Subjects that will be considered for inclusion in the scope using an extensible domain expression model. Domain expressions identify Policy Subjects to be included within the Policy Scope. Domain expressions yield an unordered set of Policy Subjects for consideration.

The RECOMMENDED means of associating a Policy with a Policy Subject that has a WSDL 1.1 description is to attach a reference to the Policy within the WSDL component corresponding to the target Policy Subject. To ensure that consumers of a Policy-annotated WSDL document are capable of processing such Policy Attachments, the specification defines a single extensibility element, `<wsp:UsingPolicy/>`, that MUST appear as an extensibility element in the containing `<wsdl:definitions/>` element of any WSDL that uses the Policy Attachment mechanism. This element can also be marked as mandatory extension with the attribute `wsdl:Required="true"`.

Below is highlight fragment of the WSDL file that refers to the policy instance applied to the whole service as a PortType and to the message:

```
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:wsa="http://schemas.xmlsoap.org/ws/2003/03/addressing"
  xmlns:wsp="http://schemas.xmlsoap.org/ws/2002/12/policy"
  xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext"
  xmlns:wst="http://schemas.xmlsoap.org/ws/2004/04/trust"
  xmlns:cnl="http://cnl.telin.nl/cnl" xmlns:policy="cnl-policy-schema.xsd"
  targetNamespace="http://cnl.telin.nl/cnl">
  <wsp:PolicyReference URI="cnl-policy-02service.xml" />
  <message name="ViewExperimentRequest"
    wsp:PolicyURIs="cnl-policy-02msg.xml">
    <part name="JobID" type="xs:string"/>
    <part name="coordinateX" type="xs:string"/>
    <part name="coordinateY" type="xs:string"/>
    <part name="zoom" type="xs:int"/>
  </message>

  <<< snip >>>>

  <wsp:UsingPolicy wsdl:Required="true"/>
</definitions>
```

Figure 6 below illustrate possible operation of the access control system components in open policy enforcement framework using standard WSA policy attachment mechanisms.

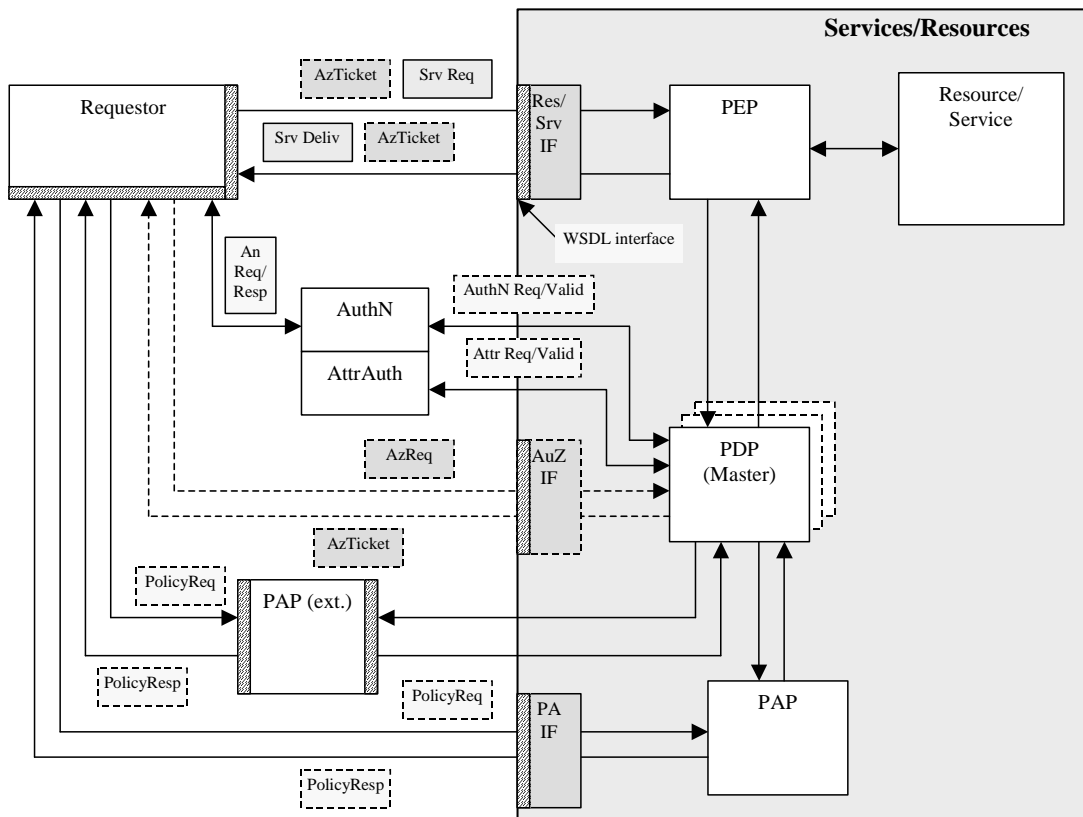


Figure 7. Open policy enforcement model in WSA/SOA using WS-Policy attachment mechanisms.

4.5 Trust relations and Policy management in distributed AAA infrastructure

This section provides an initial analysis and a summary of issues related to the policy and trust management for the basic OCE/CNL Job-centric security model and other typical use cases of the Generic AAA architecture, which have been discussed in previous sections.

The goals of this analysis are the following:

- 1) to analyse existing and define required trust and decision/policy authority relations in typical distributed access control infrastructures, first of all, relation between the Resource and PDP trust domains and the Policy authority;
- 2) provide suggestions for key/credentials distribution for the CNL security architecture
- 3) provide suggestions for formal description of trust relations in the distributed access control infrastructure using one of existing languages for credentials/certificates chain discovery in PKI.

The picture below groups main components of the CNL access control infrastructure into possible groups that are suggested to have the same level of trust and/or authority in respect to the decision making process and its context. This picture represents the Job-centric security model that can work effectively in a dynamic virtualised CNL environment.

It is assumed that in the resource access control model the root of trust belongs to the resource.

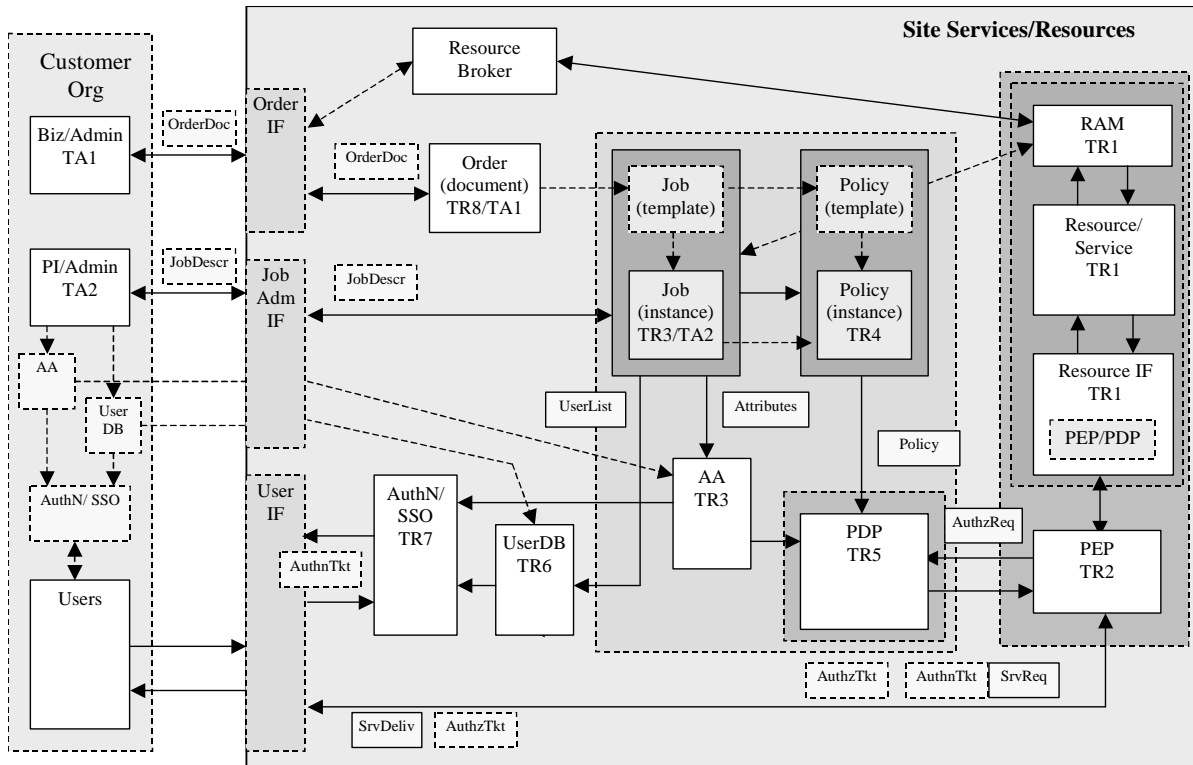


Figure 8. Trust and authority relations in the CNL/OCE access control infrastructure

The CNL/OCE Job-centric model uses one or two shared between the Customer and the Resource sites trust anchors:

- TA1 – contained in the in the signed Order created at the business negotiation stage. TA1 is considered as optional in the discussed Job-centric model but may be also used for generation of the JobDescription that has more specific role in the discussed Job-centric security model.
- TA2 – included into JobDescription that is created by actual job/experiment administrator (via special CNL/experiment Admin interface) and contains all necessary context information for configuring the CNL security services instances. TA2 is suggested as a mandatory in the discussed model.

For convenience, all components of the resource site are assigned credentials, their trust paths to the root of trust (defined by the Resource) are marked as **TR_n**, where **n** is an integer. Using credentials path semantics proposed in [cred-sem], the following trust/credentials chain and delegation are considered between major modules/objects:

User => HomeOrg.staff(TA2) => Job.members => Member.roles =>
Role.permissions

where a credential is identified by the issuer or semantic document as a prefix and an attribute/role as a suffix. This chain contains sequence of semantic meanings/trust delegation by means of cryptographical/trust relations between authorities/issuers and documents/credentials they issue. The expression above should be read as follows:

(1) User will have final permission (to do an action), if he has a credential from the HomeOrg with attribute "staff", he is contained in the "members" list of the Job description, he is assigned a role in the members attribute list (may be a part of the JobDescription or AA database), and finally user's designated role is assigned a "permission".

(2) To obtain a permission, a requestor's role must be confirmed by member roles document for a member of a project defined by the JobDescription that in its own turn provides a list of authorised members of the HomeOrg.

It is suggested that if a chain or delegation/credentials span different trust domains, the trust anchor should be placed in the joint point. The TA2 in our example is bound to the semantic document JobDescription that can be easily shared between the Resource and the customer.

Chain below illustrates trust delegation from the root at the Resource to two entities UserDB and Policy used in the request evaluation by PDP:

```
Resource => Order => JobDescription =====> UserDB(Job.members)
                                     \\
                                     =====> Policy(Role.permissions)
```

Using above semantics the process of obtaining required permissions to perform requested action the user can be described in a form:

```
User => AuthN(HomeOrg.staff, Job.members) =>
      => AuthZ(Member.roles, Policy.permissions) =>
      => Resource.permissions
```

Note. Proposed analysis is a very initial attempt to tackle the problem of formalising trust relations in distributed access control systems. More research will be needed to propose more structured approach and solution. Further analysis will intend to provide recommendations for key management and policy management in the proposed Job-centric security infrastructure.

Further analysis will intend to provide recommendations for key management and policy management in the proposed Job-centric security infrastructure.

Implementation suggestions for CNL/OCE:

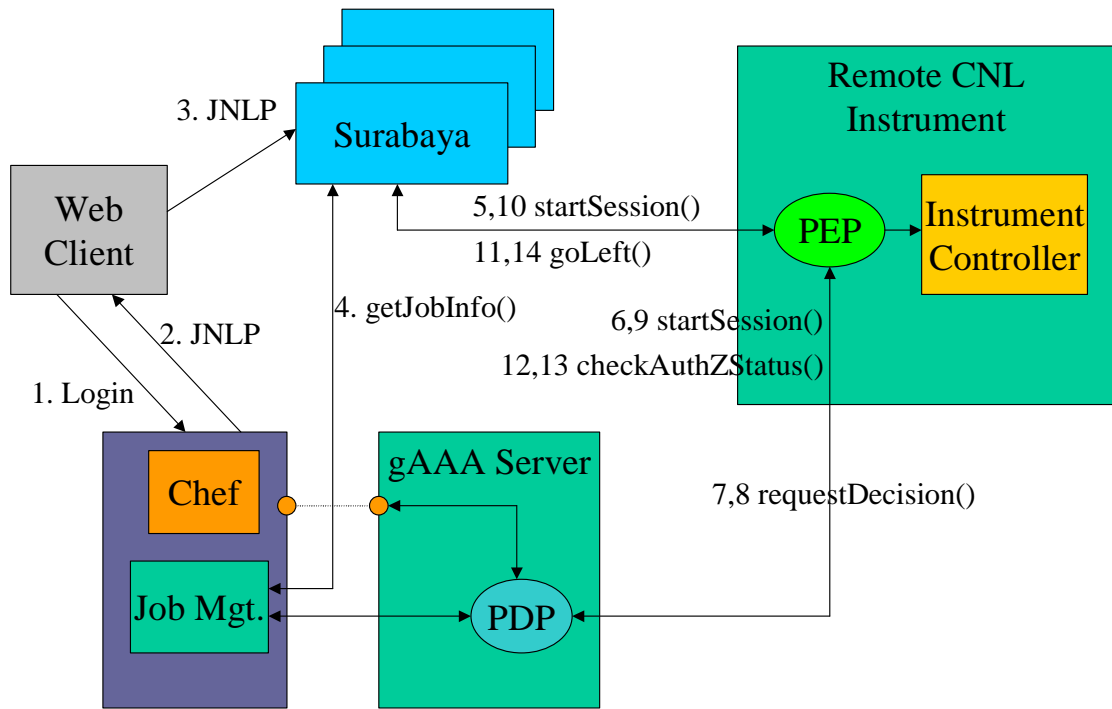
- Root of trust and authority in the proposed model belongs to the Resource
- The trust anchor TA2 embedded into the Job Description is considered as the main trust anchor shared between the resource and the customer; more business oriented model may also use the TA1 defined by the signed order. Both TA2 and TA1 may have the same trust path to the root.

- To become a shared trust anchor for establishing trusted relations between the resource and the customer domains, the Order or JobDescription must contain mutually signed credentials/certificates.
- Although the main PEP operation will assume “pull” authorisation decision request to the trusted PDP, in general it may accept the AuthzTicket from an external PDP belonging to the trusted domain.

To be continued.

5 CNL Authorisation Service operation in a Demo system

This section provides another detailed description of the CNL Authorisation service operation in a CNL2 demo system.



Note: we assume SSL TCP connections all over.

Figure 9: Security interactions in CNL

1. User logs in to Chef using their standard username/password combination
2. When the user clicks on the Launch button, a JNLP file containing:
 - a. AuthN Token
 - b. JobID to use in the default job
 - c. UserID which is a readable string

is sent through to the Web Client.

These parameters are encrypted as a block before being inserted into the JNLP file. Thus, there is only one parameter passed to Surabaya in the JNLP file, which is an aggregation of these encrypted parameters.

3. The web client forwards the JNLP file to its helper app (WebStart). This results in the launching of Surabaya. It contains a “secret” which allows it to decrypt the encrypted parameter block, thereby gaining access to the encrypted parameters. **Note:** the AuthN Token is an important item to keep secure.
4. Surabaya accesses the Job Management System and retrieves the current Job Context. This includes the user’s Role possibilities in the specified Job.
5. The user initiates a CNL Tool and requests to access a remote Instrument. In order to do this, it needs to start a remote Instrument session. It issues a startSession() call that has the following parameters:
 - a. JobID
 - b. AuthNToken
 - c. Role
 - d. UserID (could potentially be embedded in the AuthN Token)

This request is received by the Instrument Controller.

6. The Instrument Controller issues a command on the PEP requesting it to determine whether the sessionStart request should be granted. As part of this call, the above parameters are passed on.
7. The PEP assembles the parameters and requests a decision to be made by the PDP on the requested access. This involves invocation of a requestDecision() method on the PDP.
8. The PDP will:
 - a. Check that the AuthNToken is valid. This can be accomplished by checking that it has been signed by the AuthN Token issuer’s public Key.
 - b. It may also optionally check the validity of the Subject Attributes. That means that the specified UserId is actually valid for the specified Job (at that specified time), and that the specified Role is valid for that User in the Job.
 - c. It will return a Decision which can be one of the following:
 - i. DENY
 - ii. PERMIT +
a set of permitted policy action sets (**Granted Actions**) +
an AuthZ Ticket which is signed by the gAAA server
--- it may also return an obligation, which specifies what to do after the Decision has been received.
Note: An AuthZ Ticket is expected to be a very light-weight token that has a certain lifespan and that is to be used basically as a session management tool. It makes it possible for Surabaya to interact with the Instrument Controller in a session context.

9. The PEP receives the Decision and passes it on to the Instrument Controller.
10. The Instrument Controller passes back the AuthZ Ticket to Surabaya as an indication that the Session has been successfully created.
11. The User decides to steer the Instrument to the left. They interact with their Tool's GUI to do so. It then invokes a goLeft() method on the remote Instrument. This method invocation includes the following parameters:
 - a. AuthZ Ticket,
 - b. AuthN Token
12. The Instrument Controller accepts the method invocation and requests the PEP to check the Status of the AuthN.
13. The PEP checks the following:
 - a. That the AuthZ Ticket is valid.
 - i. If the AuthZ Ticket is not valid, then it needs to obtain a new ticket (return to step 7).
 - b. That the method which is being requested falls within the set of permitted policy action sets. If both conditions are OK, the PEP returns a confirmation of the AuthZ to the Instrument Controller.
14. The Instrument Controller either enacts the requested method or not, based upon the returned advice of the PEP, returning a value to Surabaya (which may be a new ticket) as a return code.

6 Using Authorisation tickets and tokens for performance optimisation in the CNL/OCE Authorisation infrastructure

6.1 CNL AuthzTicket definition and its mapping to XACML messages and SAML assertions formats

The figure below provides the graphical presentation of mapping between conceptual CNL Authorisation ticket CNLAuthzTicket, XACML Request and Response messages, and SAML 2.0 Authorisation Assertion format.

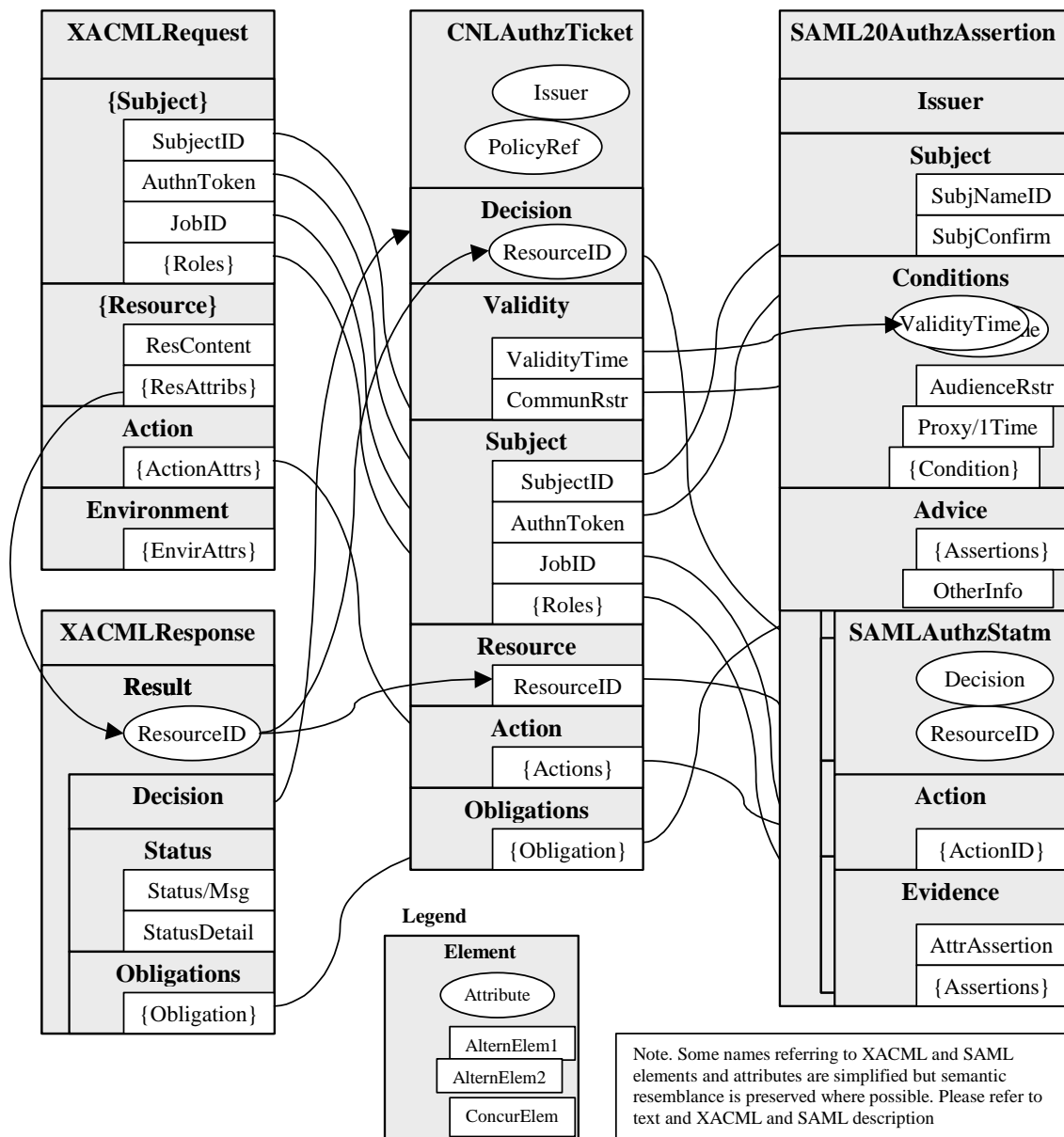


Figure 10. Mapping between XACML Request/Response, CNLAuthzTicket and SAML2.0 Authorization Assertion.

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 means 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 SAML AuthzDecisionStatement 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 SAML Assertion/Advice element or into SAML AuthzDecisionStatement/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 CNLAuthzTicket 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 SAML Assertion/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 be 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 don'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 Fig. 11 below).

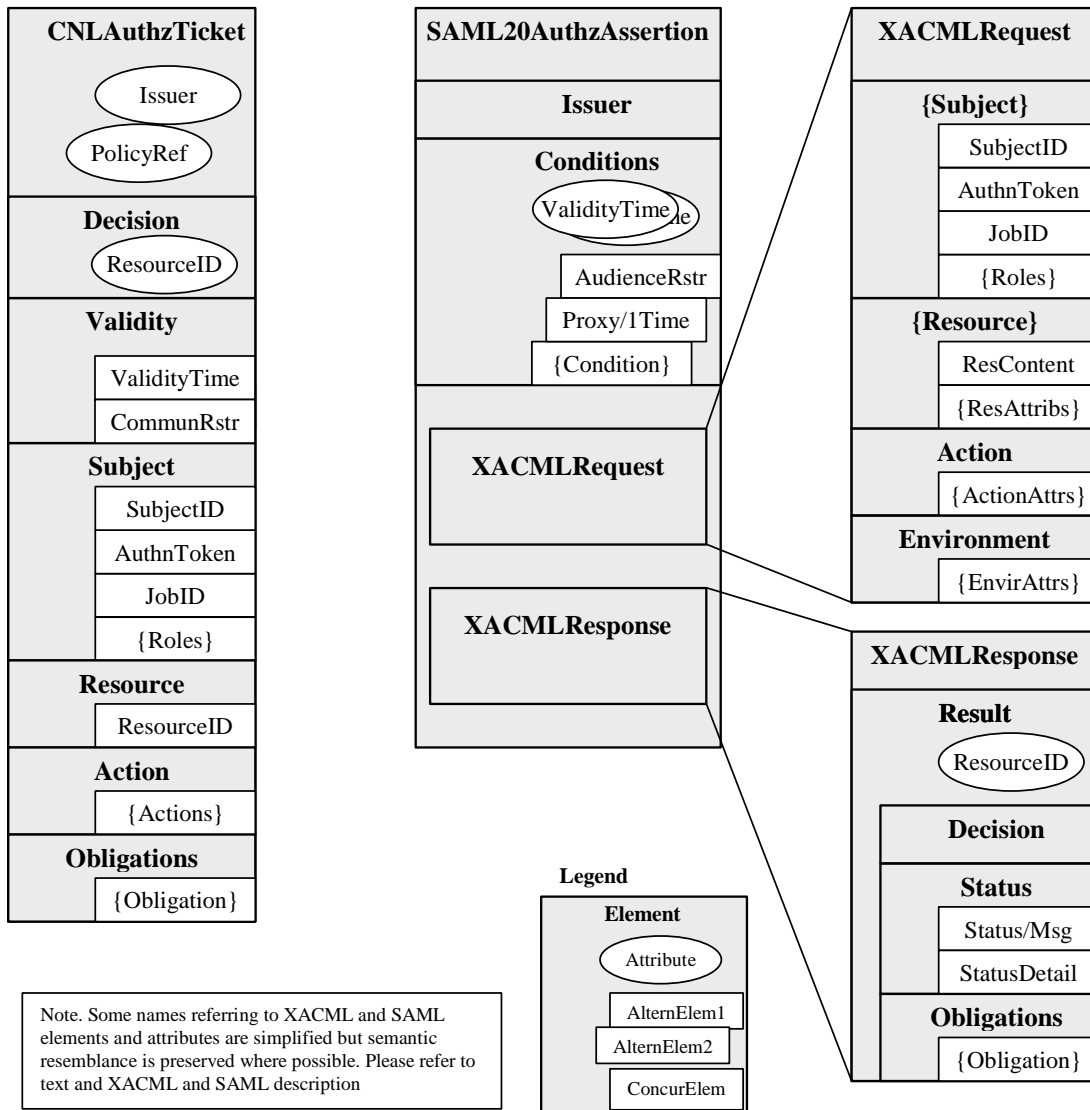


Figure 11 Mapping between CNLAuthzTicket and SAMLAuthzAssertion using XACML profile.

6.2 CNL Authorisation tickets and tokens examples

```
<cnl:CNLAuthzTicket xmlns:AAA="http://www.AAAarch.org/ns/AAA_BoD"
xmlns:cnl="http://www.aaauthreach.org/ns/#CNL"
```

```

Issuer="http://www.AAAarch.org/servers/AAA" PolicyURIs="CNLpolicy01"
SessionIndex="JobXPS1-2005-001" TicketID="ed9d969e1262bald3a7f33dbd670dd94">
  <!-- Mandatory elements -->
  <cnl:Decision
ResourceID="http://resources.collaboratory.nl/Philips_XPS1">Permit</cnl:Decision>
  <cnl:Validity NotBefore="2005-02-15T14:39:54.008Z" NotOnOrAfter="2005-02-
16T14:39:54.008Z"/>
  <!-- Additional elements -->
  <cnl:Subject Id="subject">
    <cnl:SubjectID>WHO740@users.collaboratory.nl</cnl:SubjectID>
    <cnl:AuthnToken>SeDFGVHYTY83ZXxEdsweOP8Iok</cnl:AuthnToken>
    <cnl:JobID>CNL2-XPS1-2005-02-02</cnl:JobID>
    <cnl:Role>analyst@JobID;expert@JobID</cnl:Role>
  </cnl:Subject>
  <cnl:Resource>http://resources.collaboratory.nl/Philips_XPS1</cnl:Resource>
  <cnl:Actions>
    <cnl:Action>cnl:actions:CtrlInstr</cnl:Action>
    <cnl:Action>cnl:actions:CtrlExper</cnl:Action>
  </cnl:Actions>
</cnl:CNLAUTHZTicket>

```

Listing 6.1. Simple ticket format

```

<cnl:CNLAUTHZTicket xmlns:AAA="http://www.AAAarch.org/ns/AAA_BoD"
xmlns:cnl="http://www.aaauthreach.org/ns/#CNL"
Issuer="http://www.AAAarch.org/servers/AAA" PolicyURIs="CNLpolicy01"
SessionIndex="JobXPS1-2005-001" TicketID="ed9d969e1262bald3a7f33dbd670dd94">
  <!-- Mandatory elements -->
  <cnl:Decision
ResourceID="http://resources.collaboratory.nl/Philips_XPS1">Permit</cnl:Decision>
  <cnl:Validity NotBefore="2005-02-15T14:39:54.008Z" NotOnOrAfter="2005-02-
16T14:39:54.008Z"/>
  <!-- Additional elements -->
  <cnl:Subject Id="subject">
    <cnl:SubjectID>WHO740@users.collaboratory.nl</cnl:SubjectID>
    <cnl:AuthnToken>SeDFGVHYTY83ZXxEdsweOP8Iok</cnl:AuthnToken>
    <cnl:JobID>CNL2-XPS1-2005-02-02</cnl:JobID>
    <cnl:Role>analyst@JobID;expert@JobID</cnl:Role>
  </cnl:Subject>
  <cnl:Resource>http://resources.collaboratory.nl/Philips_XPS1</cnl:Resource>
  <cnl:Actions>
    <cnl:Action>cnl:actions:CtrlInstr</cnl:Action>
    <cnl:Action>cnl:actions:CtrlExper</cnl:Action>
  </cnl:Actions>
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
      <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
      <ds:Reference URI="">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/>
          <ds:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315#WithComments"/>
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <ds:DigestValue>nrNrZZDiw/2aDnKXFEHSeoixnsc=</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>
0IZt9WsJT6an+tIxxhTPtiztDpZ+iynx7K7X2Cxd2iBwCUTQ0n61Szv81DKllWsq75IsHfusnm56
zT3fhKU1zEUsob7p6oMLM7hb42+vjfvNeJu2roknhIDzruMrr6hMDsIfaotUREpu7QCT0sADm9If
X89Et55EkSE9oE9qBD8=
</ds:SignatureValue>
  </ds:Signature>
</cnl:CNLAUTHZTicket>

```

Listing 6.2. Signed simple CNLAuthzTicket

```
<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="c236b047d62db5cecec6b240996bcb90" IssueInstant="2005-02-
15T14:53:23.542Z" Issuer="cnl:subject:CNLAAAauthority" MajorVersion="1"
MinorVersion="1">
  <Conditions NotBefore="2005-02-15T14:53:11.745Z" NotOnOrAfter="2005-02-
16T14:53:11.745Z"/>
  <AuthorizationDecisionStatement Decision="Permit"
Resource="http://resources.collaboratory.nl/Philips_XPS1">
    <Action
Namespace="urn:oasis:names:tc:SAML:1.0:action:cnl:action">cnl:actions:CtrlInstr</Ac
tion>
    <Action
Namespace="urn:oasis:names:tc:SAML:1.0:action:cnl:action">cnl:actions:CtrlExper</Ac
tion>
    <Evidence>
      <Assertion AssertionID="f3a7ea74e515ffe776b10a7eef0119d7" IssueInstant="2005-
02-15T14:53:23.542Z" Issuer="cnl:subject:CNLAAAauthority" MajorVersion="1"
MinorVersion="1">
        <Conditions NotBefore="2005-02-15T14:53:11.745Z" NotOnOrAfter="2005-02-
16T14:53:11.745Z"/>
        <AttributeStatement>
          <Subject>
            <NameIdentifier Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress"
NameQualifier="cnl:subject">WHO740@users.collaboratory.nl</NameIdentifier>
            <SubjectConfirmation>
              <ConfirmationMethod>SeDFGVHYTY83ZXxEdsweOP8Iok</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">CNL2-XPS1-2005-02-
02</AttributeValue>
            <AttributeValue
xsi:type="typens:subject">analyst@JobID;expert@JobID</AttributeValue>
          </Attribute>
        </AttributeStatement>
      </Assertion>
    </Evidence>
  </AuthorizationDecisionStatement>
</Assertion>
```

Listing 6.3. CNL AuthzTicket in SAML 1.1 format

```
<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="c236b047d62db5cecec6b240996bcb90" IssueInstant="2005-02-
15T14:53:23.542Z" Issuer="cnl:subject:CNLAAAauthority" MajorVersion="1"
MinorVersion="1">
  <Conditions NotBefore="2005-02-15T14:53:11.745Z" NotOnOrAfter="2005-02-
16T14:53:11.745Z"/>
  <AuthorizationDecisionStatement Decision="Permit"
Resource="http://resources.collaboratory.nl/Philips_XPS1">
    <Action
Namespace="urn:oasis:names:tc:SAML:1.0:action:cnl:action">cnl:actions:CtrlInstr</Ac
tion>
    <Action
Namespace="urn:oasis:names:tc:SAML:1.0:action:cnl:action">cnl:actions:CtrlExper</Ac
tion>
```

```

<Evidence>
  <Assertion AssertionID="f3a7ea74e515ffe776b10a7eef0119d7" IssueInstant="2005-
02-15T14:53:23.542Z" Issuer="cnl:subject:CNLAAAauthority" MajorVersion="1"
MinorVersion="1">
    <Conditions NotBefore="2005-02-15T14:53:11.745Z" NotOnOrAfter="2005-02-
16T14:53:11.745Z"/>
    <AttributeStatement>
      <Subject>
        <NameIdentifier Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress"
NameQualifier="cnl:subject">WHO740@users.collaboratory.nl</NameIdentifier>
        <SubjectConfirmation>
          <ConfirmationMethod>SeDFGVHYTY83ZXxEdsweOP8Iok</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">CNL2-XPS1-2005-02-
02</AttributeValue>
        <AttributeValue
xsi:type="typens:subject">analyst@JobID;expert@JobID</AttributeValue>
      </Attribute>
    </AttributeStatement>
  </Assertion>
</Evidence>
</AuthorizationDecisionStatement>
<ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
  <ds:SignedInfo>
    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
    <ds:Reference URI="">
      <ds:Transforms>
        <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/>
        <ds:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315#WithComments"/>
      </ds:Transforms>
      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <ds:DigestValue>YHUVqrSI/y+EDzdVTKQ45cKKzS0=</ds:DigestValue>
    </ds:Reference>
  </ds:SignedInfo>
  <ds:SignatureValue>
Yy1pKNhPXoGPLrZ806UEFZe+bYacwoBajhXsM1/nGzMe/597xca6hJw/z8mdVs83iRCYiVnzqLdl
Aht5ID0c0YSrMyJlGkBeynliqNroIp//sFuz4T4aLXsektuBgM2E6hDCvlnSqc+VxmRdiGnPJ11W
VM5noQ5cxyGf2OXQTHo=
</ds:SignatureValue>
</ds:Signature>
</Assertion>

```

Listing 6.4 Signed CNLSAMLAUTHZTicket

```

<cnl:CNLAUTHZToken TokenID="ed9d969e1262ba1d3a7f33dbd670dd94">
  <cnl:TokenValue>
0IZt9W9sJT6an+tIxhhTptiztDpZ+iynx7K7X2Cxd2iBwCUTQ0n61Szv81DK1lWsq75IsHfusnm56
zT3fhKU1zEUSob7p6oMLM7hb42+vJfvNeJu2roknhIDzruMrr6hMDsIfaotUREpu7QCT0sADM9If
X89Et55EkSE9oE9qBD8=
</cnl:TokenValue>
</cnl:CNLAUTHZToken>

```

```

<cnl:CNLAUTHZToken ID="ticketID">
  <cnl:TokenValue>
o9uZ9XVKY27fYeGBM/oPFxgAzyjI6Qadk1V5EA7R6jJlv6szK3g88yXdrFxFxK/mTKzet+siJtUN
xq9O4AhFYubjXZwqe7TJpMXdkObWNe3dCEnYde/OHsLon/5mBDWvbgXcVksXVnN30UBRr4gTCmMK
tLntTWqlJ7iHSg9abVc=

```

```
</cnl:TokenValue>
</cnl:CNLAAuthzToken>
```

Listing 6.5. Signed CNLAAuthzToken in full format (a) and minimum format (b)

```
<cnl:CNLAAuthnTicket xmlns:AAA="http://www.AAAarch.org/ns/AAA_BoD"
xmlns:cnl="http://www.aaauthreach.org/ns/#CNL"
Issuer="http://www.AAAarch.org/servers/AAA"
TicketID="f35585dfb51edec48de0c7eadb11c17e">
  <!-- Mandatory elements -->
  <cnl:Validity NotBefore="2005-02-15T14:33:10.548Z" NotOnOrAfter="2005-02-
16T14:33:10.548Z"/>
  <cnl:Subject Id="subject">
    <cnl:SubjectID>WHO740@users.collaboratory.nl</cnl:SubjectID>

<cnl:SubjectConfirmationData>SeDFGVHYTY83ZXxEdsweOP8Iok</cnl:SubjectConfirmationDat
a>
  <!--Optional elements -->
  <cnl:SubjectAttribute attrname="urn:cnl:subject:attribute:job-id">CNL2-XPS1-
2005-02-02</cnl:SubjectAttribute>
  <cnl:SubjectAttribute
attrname="urn:cnl:subject:attribute:role">analyst@JobID;expert@JobID</cnl:SubjectAt
tribute>
  </cnl:Subject>
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
      <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
      <ds:Reference URI="">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-
signature"/>
          <ds:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315#WithComments"/>
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <ds:DigestValue>TVHE9Ovf7r/St/zbzi2ZrhWUblI=</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>
aCRW7MSeNDiaOmXUrSRcJe/8BPCcgWmlGBHhQwE633VlfmOdvACYVOFbfC6Q05Za4eNivucX1y7s
twb2a6sIRIQtre9hTsfHnKFdmKOFVrfjStMFkFGvkhFKnUuRfKP3+tT68oj/rNO32VtpV8m5Ik8e
jGfQ106WvAVsqE2+AEA=
</ds:SignatureValue>
  </ds:Signature>
</cnl:CNLAAuthnTicket>
```

Listing 6.5. Signed CNLAAuthzToken in full format (a) and minimum format (b)

```
<cnl:CNLAAuthnToken xmlns:cnl="http://www.aaauthreach.org/ns/#CNL"
TokenID="f35585dfb51edec48de0c7eadb11c17e">
  <cnl:SubjectID>WHO740@users.collaboratory.nl</cnl:SubjectID>
  <cnl:TokenValue>SeDFGVHYTY83ZXxEdsweOP8Iok</cnl:TokenValue>
</cnl:CNLAAuthnToken>
```

Listing 6.5. Signed CNLAuthzToken in full format (a) and minimum format (b)

7 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].

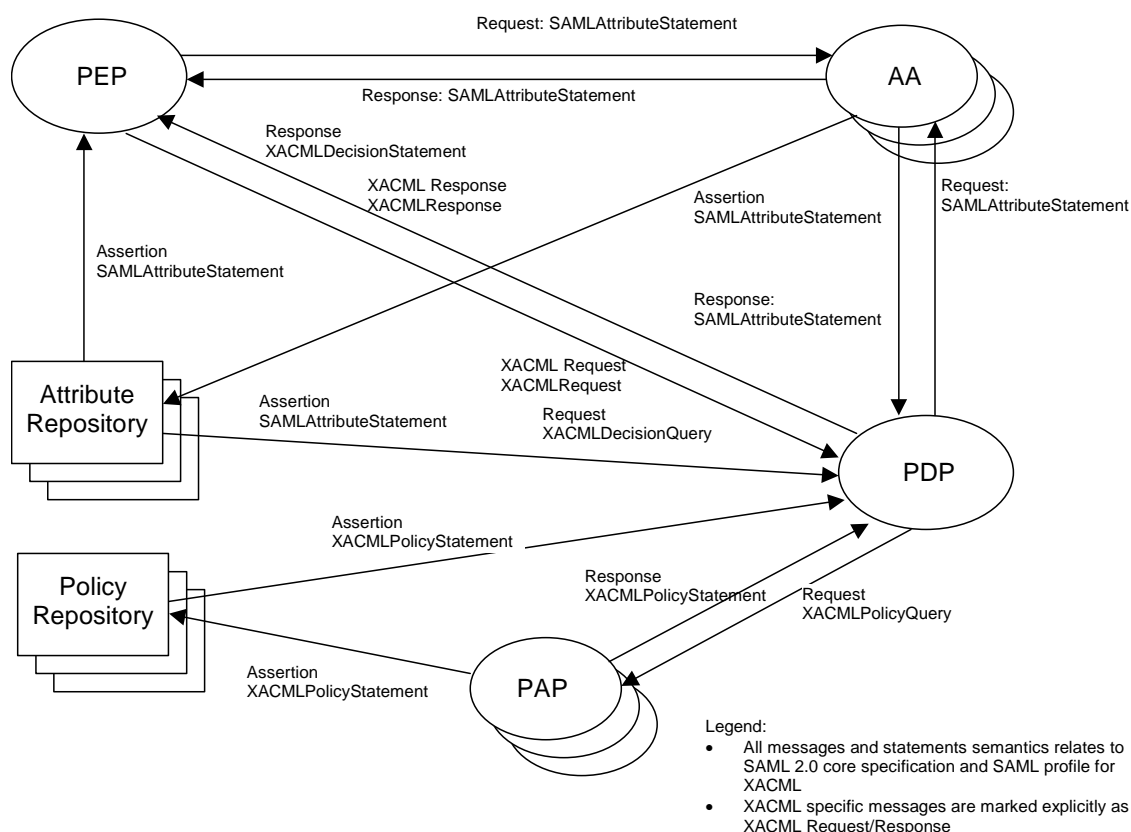


Figure 12 Using SAML and XACML for messaging and assertions

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.

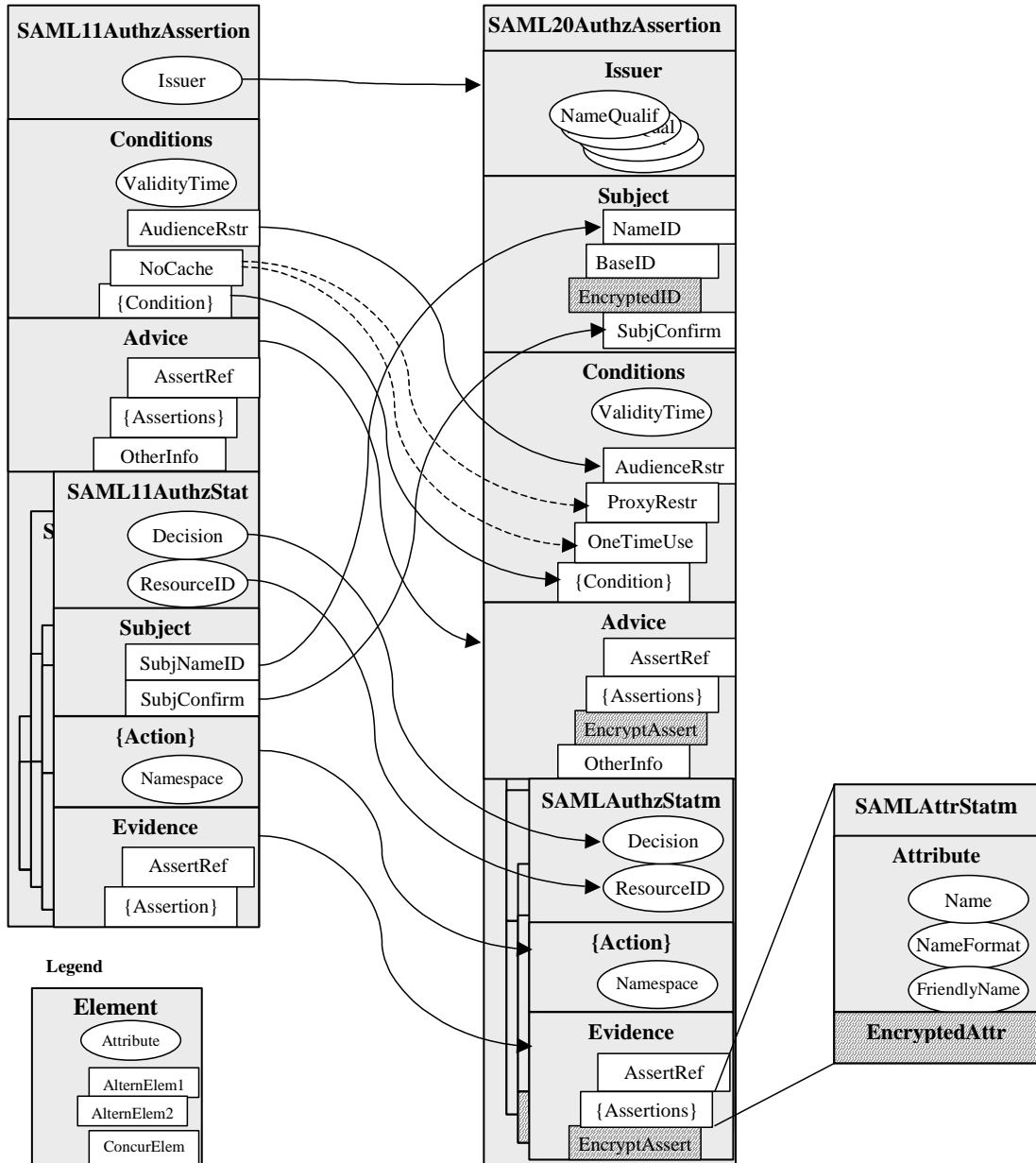


Figure 13. Comparison between SAML 1.1 and SAML 2.0 (for the Authorization Assertion as an example)

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.

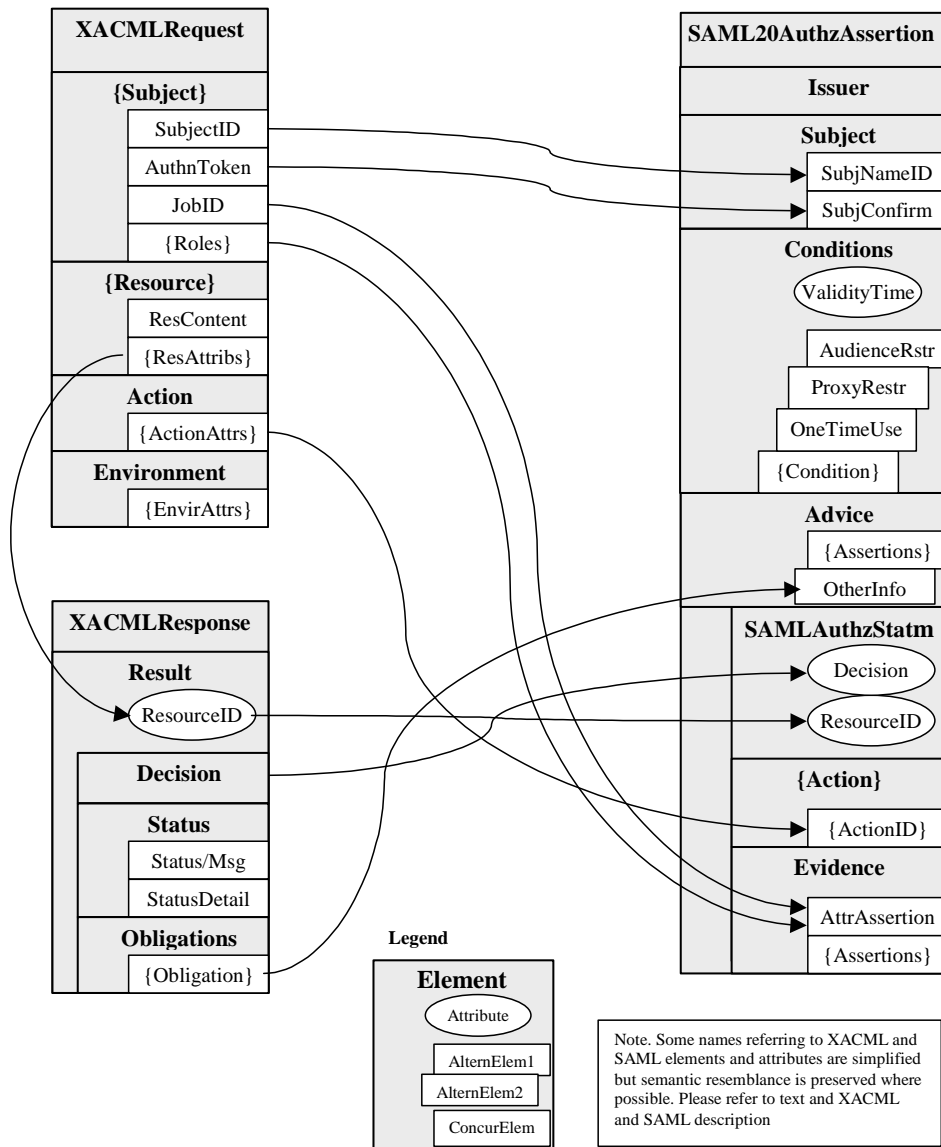


Figure 14. Mapping between XACML Request/Response messages and SAML 2.0 Authorisation assertion.

8 Providing Integrity and Confidentiality with XML Digital Signature and XML Encryption

XML Signature and XML Encryption are two standard mechanisms used to provided integrity and confidentiality of XML documents. Both of these mechanisms are integrated into SAML. Generally, XML Signature and XML Encryption can be added to any other XML format at the level of the XML schema extension or to any valid XML document instance by using numerous available XML Security packages.

8.1 XML Digital Signature format and processing

This section provides general information about XML Signature format and some suggestions how it can be used together with SAML, XACML and XML Web Services in CNL.

XML Signature has the following structure:

```

<Signature ID?>
  <SignedInfo>
    <CanonicalizationMethod/>
    <SignatureMethod/>
    (<Reference URI? >
      (<Transforms>)?
      <DigestMethod>
      <DigestValue>
    </Reference>)+
  </SignedInfo>
  <SignatureValue>
  (<KeyInfo>)?
  (<Object ID?>)*
</Signature>

```

There are three types of the XML Signature algorithms:

- **Enveloped XML Signature.**
An enveloped signature is a signature of either an entire document or a document fragment, where the XML signature will itself be embedded within the signed document. An enveloped signature transform is always used to remove the signature structure from the signing process.
- **Enveloping XML Signature**
An enveloping signature is a signature where the signed data is actually embedded within the XML signature structure: The XML signature specification provides the ability for arbitrary XML structures to be embedded within a signature, for this express purpose.
- **Detached XML Signature**
A detached signature is a signature where the signed entities are separate from the actual signature fragment. The signed entities can be remote XML documents or remote non-XML documents. They can also be XML fragments located elsewhere in the same document as the XML signature.

It is important to mention that WS-I Basic Security Profile for Web Services recommends using detached signature and strongly discourages enveloping signature use, enveloped signature can be used but it provides limited functionality for signing the document and managing security components.

More details about XML Signature syntax and processing together with examples can be found in the Appendix D.

XML Signature Implementation suggestions for CNL/OCE;

For the purpose of CNL security assertions/tickets exchange and managing security context in the Job-centric security model, the enveloped XML Signature provides necessary functionality:

- Enveloped signature is always placed under the root of document, however multiple Signatures are allowed in one document.
- Enveloped Signature can sign document in total and multiple elements defined by multiple Reference elements

- To protect XML document integrity, it is recommended to sign the whole document with URI = "" and other critical security components such as security tokens, subject attributes and conditions that can be identified by element's ID/Id, xpointer or XPath expressions.
- Such voluminous component of the XML Signature element as KeyInfo can be reduced to KeyName if exchanging systems use pre-installed shared keys or public key certificates.
- The input document (in the form of octet stream or XML object) MUST be well-formed XML document, but the input needs not to be validated.

8.2 XML Encryption

XML Encryption datamodel:

```
<EncryptedData Id? Type? MimeType? Encoding?>
  <EncryptionMethod/>?
  <ds:KeyInfo>
    <EncryptedKey>?      # extension to XMLSig KeyInfo
    <AgreementMethod>?
    <ds:KeyName>?
    <ds:RetrievalMethod>?
    <ds:*>?              #
  </ds:KeyInfo>?
  <CipherData>          # envelopes or references the raw encrypted data
    <CipherValue>?
    <CipherReference URI?>? # points to the location of the raw encrypted
data
  </CipherData>
  <EncryptionProperties>? # e.g., timestamp
```

</EncryptedData> CipherData element contains the encrypted octet sequence as base64 encoded text of the CipherValue element, or provides a reference to an external location containing the encrypted octet sequence via the CipherReference element.

XML Encryption implementation suggestions for CNL/OCE;

XML Encryption can be used to encrypt sensitive parts of CNL Job description that provides both BA and TA for CNL/OCE secure environment. It can also provide a method to exchange shared secrets like in case of establishing trusted relations between participating in CNL parties or VO members. Actually, XML Encryption is a part of WS-SecureConversation mechanisms.

- XML Encryption uses a security paradigm slightly different from XML Signature. XML Signature uses public key or symmetric crypto algorithm for signing quite straightforward. Major suggested/recommended XML Encryption procedure uses symmetric data encryption key (dek), often generated instantly, for encrypting data and next supplies this (instantly generated) data encryption key in the encrypted form using another symmetric key (kek) or public key of the recipient. The reason for this is that the symmetric key and data encrypted with the symmetric key have smaller size.
- Recommended in XML Encryption specification and by WS-I are the following algorithms: AES as a data encryption algorithm and key; TripleDES key wrap or RSA version 1.5 key transport for dek encryption.
- To encrypt XML document or element with Apache XML Security package for Java, it is required to provide the context document and an element to be encrypted. It is important that the DOM methods used to get the element is namespace aware as both XML Encryption and XML Signature are namespace aware. This is in particular related to the methods getElementByTagNameNS(URI-NS, ElementName) which is namespace aware and

getElementById(IdString) which is not namespace aware. However, the element can be identified by Id by applying ds:Reference URI transformation.

9 References

- [1] Security Architecture for Open Collaborative Environment, by Yuri Demchenko, Leon Gommans, Cees de Laat, Bas Oudenaarde, Andrew Tokmakoff, Martin Snijders, Rene van Buuren. - Accepted paper for EGC2005 Conference February 14-16, 2005.
- [2] Job-centric Security model for Open Collaborative Environment, by Yuri Demchenko, Leon Gommans, Cees de Laat, Bas Oudenaarde, Andrew Tokmakoff, Martin Snijders. - Accepted paper for CTS2005 Conference May 2005.
- [3] Collaboration and security in CNL's virtual laboratory, by Andrew Tokmakoff, Yuri Demchenko and Martin Snijders. WACE 2004, 23 September 2004. -<http://www-unix.mcs.anl.gov/fl/events/wace/wace2004/talks/tokmakoff.pdf>
- [4] Demchenko Yu. Virtual Organisations in Computer Grids and Identity Management. – Elsevier Information Security Technical Report - Volume 9, Issue 1, January-March 2004, Pages 59-76.
- [5] Web Services Architecture, W3C Working Draft 8 August 2003 - <http://www.w3.org/TR/ws-arch/>
- [6] The Open Grid Services Architecture, Version 1.0 – 12 July 2004, 2004. - <http://www.gridforum.org/Meetings/GGF12/Documents/draft-ggf-ogsa-specv1.pdf>
- [7] Security in a Web Services World: A Proposed Architecture and Roadmap, Version 1.0, A joint security whitepaper from IBM Corporation and Microsoft Corporation. April 7, 2002, <http://www-106.ibm.com/developerworks/library/ws-secmap/>
- [8] 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>
- [9] 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>
- [10] Role Based Access Control (RBAC) – NIST, April 2003. - <http://csrc.nist.gov/rbac/>
- [11] RFC 2903 , Experimental, "Generic AAA Architecture",. de Laat, G. Gross, L. Gommans, J. Vollbrecht, D. Spence, August 2000 - <ftp://ftp.isi.edu/in-notes/rfc2903.txt>
- [12] RFC 2904 , Informational, "AAA Authorization Framework" J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege, D. Spence, August 2000 - <ftp://ftp.isi.edu/in-notes/rfc2904.txt>
- [13] 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>
- [14] 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

- [15] Web Services Architecture, W3C Working Draft 8 August 2003 - <http://www.w3.org/TR/ws-arch/>
- [16] Web Services Security Framework by OASIS - http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
- [17] 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>
- [18] 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>
- [19] A grammar for Policies in a Generic AAA Environment - <http://www.ietf.org/internet-drafts/draft-irtf-aaaarch-generic-policy-05.txt>
- [20] Web Services Policy Framework (WS-Policy). Version 1.1. - <http://msdn.microsoft.com/ws/2002/12/Policy/>
- [21] Web Services Policy Attachment (WS-PolicyAttachment). - <http://msdn.microsoft.com/ws/2004/09/PolicyAttachment/>
- [22] XACML profile for Web-services (WSPL): - <http://www.oasis-open.org/committees/download.php/3661/draft-xacml-wspl-04.pdf>
- [23] Web-services policy language use-cases and requirements: <http://www.oasis-open.org/committees/download.php/1608/wd-xacml-wspl-use-cases-04.pdf>
- [24] 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
- [25] 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
- [26] 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
- [27] 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
- [28] Web Services Federation Language (WS-Federation) Version 1.0 - July 8 2003 – <http://msdn.microsoft.com/ws/2003/07/ws-federation/>
- [29] Liberty Alliance Phase 2 Final Specifications - <http://www.projectliberty.org/specs/>
- [30] Web Services Secure Conversation Language (WS-SecureConversation) - <http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-secureconversation.asp>
- [31] Grid User/Site Security Requirements – MJRA3.1 - <https://edms.cern.ch/document/485295/1>

- [32] EGEE Global Security Architecture (GSA) rev. 1 - DJRA3.1 - <https://edms.cern.ch/document/487004/1.1>
- [33] EGEE Site Access Control Architecture. – DJRA3.2 - <https://edms.cern.ch/file/523948/0.20/DJRA3.2-0.20.pdf>
- [34] Virtual Organization Membership Service - <http://edg-wp2.web.cern.ch/edg-wp2/security/voms/>
- [35] 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.
- [36] WS-I Basic Security Profile Version 1.0. Working Group Draft. – May 12, 2004. - <http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0-2004-05-12.html>
- [37] Device Profile for Web Services. Microsoft Corporation. - August 2004. - <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnglobspec/html/devprof.asp>
- [38] XML-Signature Syntax and Processing. – W3C REC: <http://www.w3.org/TR/xmlsig-core/> Draft IETF Standard: <http://www.ietf.org/rfc/rfc3275.txt>
- [39] XML Encryption XML Encryption Syntax and Processing - <http://www.w3.org/TR/xmlenc-core/>
- [40] Use of SAML for OGSA Authorisation Use of SAML for OGSA Authorization. By Von Welch, Frank Siebenlist, Sam Meder, Laura Pearlman. – February 15, 2004. - <http://www.ggf.org/Meetings/ggf7/drafts/OGSA%20SAML%20Authorization%20Assertions-Feb15.pdf>
- [41] GFD.38 Conceptual Grid Authorization Framework and Classification. M. Lorch, B. Cowles, R. Baker, L. Gommans, P. Madsen, A. McNab, L. Ramakrishnan, K. Sankar, D. Skow, M. Thompson - <http://www.ggf.org/documents/GWD-I-E/GFD-I.038.pdf>
- [42] Java Cryptography Extension (JCE) Reference Guide for the Java 2 SDK, Standard Edition, v 1.4. - <http://java.sun.com/j2se/1.4.2/docs/guide/security/jce/JCERefGuide.html>
- [43] Java Cryptography Architecture (JCA): API Specification & Reference - <http://java.sun.com/j2se/1.4.2/docs/guide/security/CryptoSpec.html>
- [44] Java Secure Socket Extension (JSSE) Reference Guide for the Java 2 SDK, Standard Edition, v 1.4. - <http://java.sun.com/j2se/1.4.2/docs/guide/security/jsse/JSSERefGuide.html>
- [45] N. Li, W. Winsborough, J.C. Mitchell, Distributed Credential Chain Discovery in Trust Management, J. Computer Security, vol 11, no 1, 2003, pages 35-86. - <http://theory.stanford.edu/people/jcm/papers/disc.pdf>
- [46] RFC 2693 SPKI Certificate Theory. - <http://www.ietf.org/rfc/rfc2693.txt>
- [47] Internet X.509 Public Key Infrastructure: Certification Path Building. Work in progress. - <http://www.ietf.org/internet-drafts/draft-ietf-pkix-certpathbuild-05.txt>