GENERALISED ARCHITECTURE FOR DYNAMIC INFRASTRUCTURE SERVICES

Large Scale Integrated Project
Co-funded by the European Commission within the Seventh Framework Programme
Grant Agreement no. 248657
Strategic objective: The Network of the Future (ICT-2009.1.1)

Start date of project: January 1st, 2010 (36 months duration)

Security Architecture for On-Demand Infrastructure Services Provisioning

Version 0.2

Due date:	Not applicable
Submission date:	Not applicable
Deliverable leader:	University of Amsterdam
Author list:	Yuri Demchenko
	Canh Ngo

Dissemination Level

☒	**PU:**	Public
☐	**PP:**	Restricted to other programme participants (including the Commission Services)
☐	**RE:**	Restricted to a group specified by the consortium (including the Commission Services)

☐ **CO:** Confidential, only for members of the consortium (including the Commission Services)

Abstract

This document provides the general description of the security architecture for GEYSERS on-demand infrastructure services provisioning, its major components and implementation suggestions.

Table of Contents

0 Objectives and scope of document

This document provides the general description of the security architecture for GEYSERS on-demand infrastructure services provisioning, its major components and implementation suggestions.

The major objectives of the document is to provide necessary information to developers of other components of the GEYSERS architecture how to integrate and use security services to achieve secure operation of the whole GEYSERS infrastructure.

1 Introduction

The main objective of the GEYSERS project is to address some of the key technical challenges to enable on-demand Network and IT resources and infrastructure services provisioning. The Authentication, Authorization Infrastructure (AAI) is as an important component of the supporting infrastructure for on-demand Infrastructure Services Provisioning (ISOD). Consistent AAI operation requires interaction of the related AAI components at all ISOD layers and during all provisioning stages.

This document describes the result of the development of the AAI architecture for ISOD. The proposed architecture attempts to address key access control problems when integrating heterogeneous virtualisation platforms and Control and Management planes. The proposed architecture also targets to ensure future compatibility with the emerging Cloud platforms and physical resources access control solutions and infrastructures.

The report is organised as follows. Section 2 provides short description of the GEYSERS architecture including GEYSERS Service Delivery Framework (SDF) followed by the description of the basic use cases and abstraction model used for security services and AAI definition and development. Section 3 defines the general requirements to ISOD security infrastructure and services.

Section 4 provides an overview of the generic access control models such as Role Based Access Control (RBAC), Generic AAA Authorization Framework (GAAA-AuthZ) and its extension for dynamically provisioned services Dynamic Access Control Infrastructure (DACI). The section describes also solutions used in existing Cloud IaaS platform such as Amazon Web Services (AWS) and Microsoft Azure.

Section 5 describes the proposed AAI architecture for on-demand Infrastructure Services Provisioning (ISOD) that address both tasks – secure operation or the provisioning infrastructure and provisioning of the Dynamic Access Control Infrastructure (DACI) as a part of the on-demand provisioned infrastructure. The proposed architecture framework includes also such components as Security Services Lifecycle Management (SSLM) model and security context management framework. It identifies key functionalities to support multidomain network+IT infrastructure services and introduces a number of mechanisms and solutions to support them, in particular: AuthZ ticket format for extended AuthZ session management, Token Validation Service (TVS) to enable token based policy enforcement, policy Obligation Handling Reference Model (OHRM), and XACML policy profile for ISOD. The proposed architecture will allow smooth integration with other authorization frameworks as currently used and developed by Cloud and networking community.

Section 6 describes how the proposed GAAA-ISOD architecture is implemented in the current version of the GAAA Toolkit. It provides general description of the GAAA Toolkit structure and functionalities to support network resource provisioning and more detailed description of such components as TVS and GAAAPI that can be used as a pluggable component to add AAA/AuthZ services to different NRPS frameworks.

Section 7 provides detailed description of the Common Security Services Interface (CSSI) that is used as a common generalised interface for accessing AAI/GAAA-ISOD services and for their simple integration with other components of the GEYSERS architecture.

Finally, section 7 provides summary of the current results and suggests further developments.

2 GEYSERS Architecture Overview

2.1 GEYSERS Architecture

The GEYSERS architecture re-qualifies the interworking of legacy planes by means of a virtual infrastructure representation layer for network and IT resources and its advanced resource provisioning mechanisms. The GEYSERS architecture presents an innovative structure by adopting the concepts of Infrastructure as a Service (IaaS) and service-oriented networking to enable infrastructure operators to offer new network and IT converged services. On one hand, the service-oriented and IaaS paradigm enable flexibility of infrastructure provisioning in terms of configuration, accessibility and availability for the user. On the other hand, the layer-based structure of the architecture enables separation of functional aspects of each of the entities involved in the converged service provisioning, from the service consumer to the physical infrastructure. Figure 2.1 shows the layering structure of the GEYSERS architecture reference model comprised of four layers: the Service Middleware Layer (SML), the enhanced Network Control Plane (NCP), the novel Logical Infrastructure Composition Layer (LICL) and the physical infrastructure.

The Logical Infrastructure Composition Layer (LICL) [3] is a middleware aiming at decoupling infrastructure resource management from the actual service provisioning. This is performed by adopting an Infrastructure as a Service (IaaS) management model for both optical network and IT resources. Although IaaS is a well-known model in IT environment, it is not so common for networking, in favour of Network as a Service (NaaS)..

In addition to IaaS, LICL is based in infrastructure resource virtualisation paradigms for granting flexible infrastructure service provisioning. A number of projects have successfully dealt with virtualisation for leveraging infrastructure resources utilisation. At the same time, virtualisation allows reducing capitalisation costs, which is especially critical for scientific communities where the equipment acquisition and network deployment costs considerably diminish project budgets.

The LICL is located between the physical infrastructure resources and the upper layers in GEYSERS architecture, such as the Network Control Plane and the Service Middleware Layer. In GEYSERS architecture, the LICL is responsible for the creation and maintenance of virtual resources as well as virtual infrastructures. In the context of GEYSERS, infrastructure virtualisation is the creation of a virtual representation of a physical resource (e.g., optical network node or computing device), based on an abstract model that is often achieved by partitioning or aggregation. A virtual infrastructure is a set of virtual resources interconnected together that share a common administration framework. Within a virtual infrastructure, virtual connectivity (virtual link) is defined as a connection between one port of a virtual network element to a port of another virtual network element.

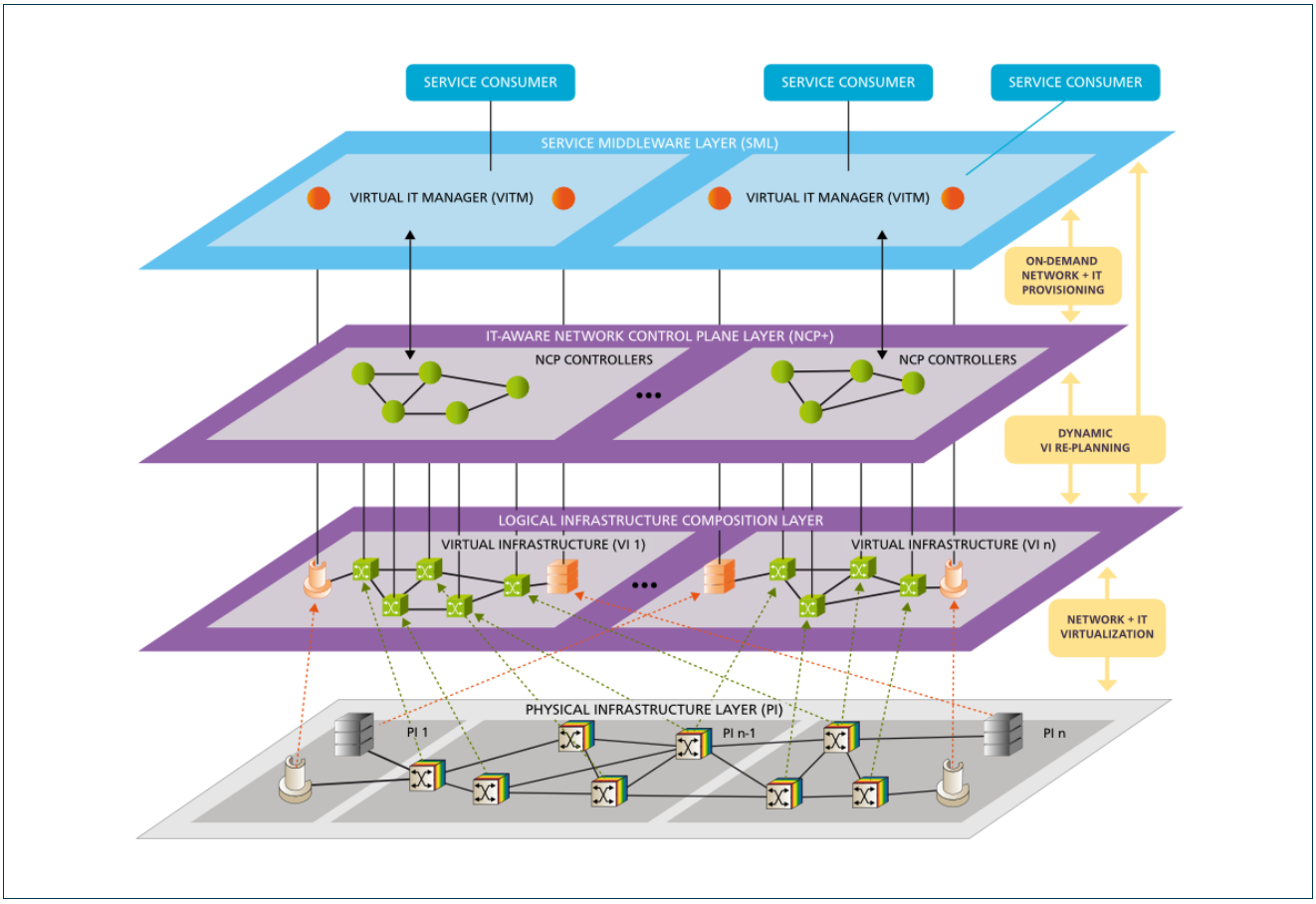


Figure 2.2-1: GEYSERS Architecture

2.2 Logical Infrastructure Composition Layer (LICL)

LICL is the key element in the GEYSERS architecture in order to provision infrastructure services. This section provides a short description of the functional architecture of the LICL that provides a practical implementation of the abstract IaaS provisioning model described in section VIII (refer to this section for PIP, VIP and VIO defintion).

The LICL is divided into two main sub-systems depending on the functionalities implemented in each sub-system and also depending on the role that uses such functionalities. On the one hand, there is the upper-LICL, which is responsible mainly for the virtual infrastructure management and satisfies the needs and requirements of the virtual infrastructure provider. On the other hand we have, the lower-LICL, which is responsible for physical resource virtualisation and management and which satisfies the requirements of the physical infrastructure provider.

The upper-LICL is composed of different modules. The functionalities covered at this level are the virtual infrastructure creation, management and re-planning, and the SLA enforcement. The virtual infrastructure creation is done as a composition of different virtual resources available from one or multiple PIPs. Such a virtual infrastructure is provisioned towards the virtual infrastructure operator as a unit. Furthermore, the upper-LICL offers dynamic re-planning functionalities as a response to the changing requirements of the VIO. Such dynamic re-planning may involve the inclusion of new resources to the virtual infrastructure, the release of un-used resources, or even the resizing of some of them (e.g., increase or decrease the total bandwidth capability of a virtual link). As a part of the system oriented to provide dynamic infrastructure services, the upper-LICL provides capabilities to ensure SLA levels are met during the whole service lifecycle.

The lower-LICL covers the functionalities regarding physical resource abstraction and resource virtualisation. The tools offered by the lower-LICL are used by the PIP in order to manage its own infrastructure. The lower-LICL is responsible for the physical resource abstraction that basically comprehends all the necessary steps to create a logical resource representing the physical resource. It also is in charge of the virtual resource creation and management, as well as the resource monitoring and configuration. The lower-LICL also offers an information service, which is used by the PIP to send information about its domain capabilities towards the different VIPs.

Figure 2.3 depicts the functional architecture of the LICL, split into the two aforementioned components. It also shows the different interfaces in each component in order to communicate with the outer world. In the case of the upper-LICL, it has the Management-to-LICL (MLI) interface, which offers all the virtual infrastructure management operations (e.g., request, re-planning, decommission) and then the SML-to-LICL (SLI) interface and the Call Controller Interface (CCI), used to offer operation capabilities over the virtual infrastructure. In detail, the SLI offers operations over the virtual IT resources and the CCI over the virtual network resources. However, it is remarkable that this is a logical differentiation, since the implementation of the system offers one interface and handles the virtual resources in a converged manner independently of its nature. Finally, the lower-LICL offers the VR request service, used to request for single virtual resources, the Resource Operation Service, that represents the operation interfaces for the virtual resources, and the information service, which is used to exchange information with the different physical infrastructure providers.

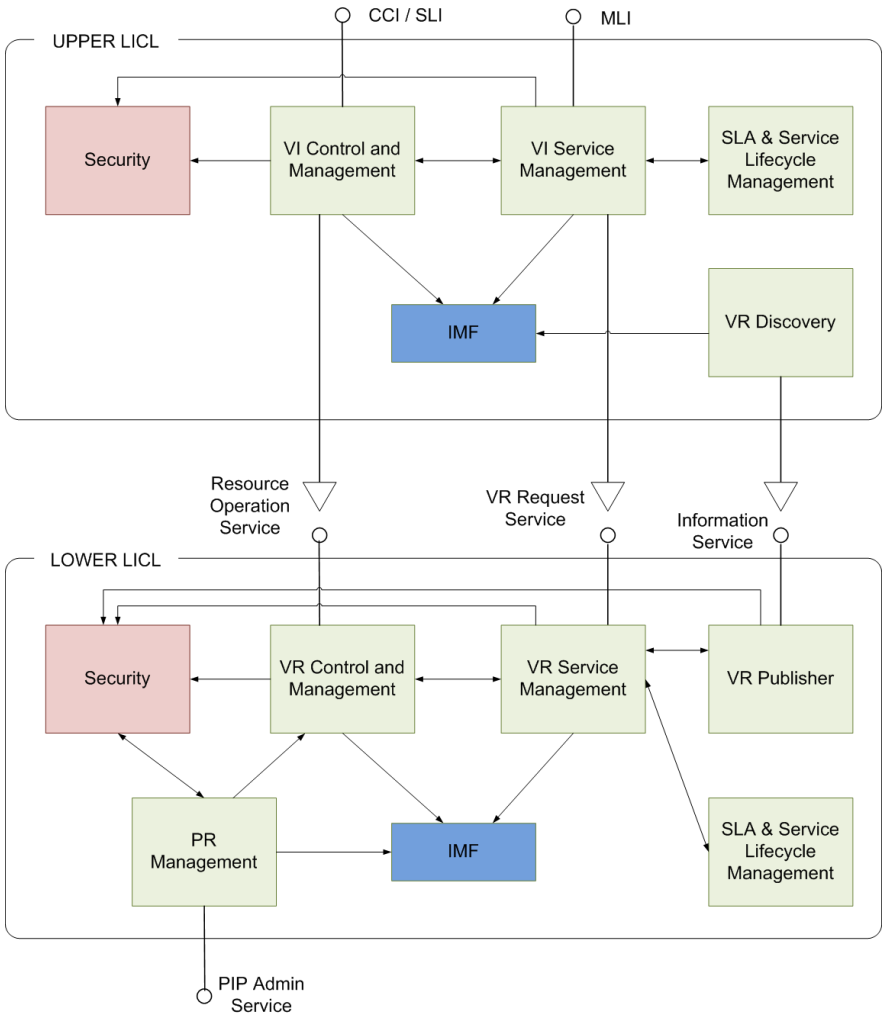


Figure 2.3?: LICL functional architecture overview

2.3 General use-cases in On-demand Infrastructure Services Provisioning

The two basic use-cases for on-demand infrastructure service provisioning can be considered: large scientific infrastructures and network infrastructure provisioning [4, 5, 6]. These use-cases represent the two different perspectives in developing infrastructure services – the user and application developer perspective on one side, and the provider perspective on the other side. Users are interested in uniform and simple access to the resource and the services that are exposed as Cloud or Grid resources and can be easily integrated into the scientific or business workflows. Infrastructure providers are interested in infrastructure resource pooling and virtualisation to simplify their on-demand provisioning and extend their service offering and business model to Virtual Infrastructure provisioning.

Figure 2.2 illustrates the typical e-Science infrastructure that includes Grid and Cloud based computing and storage resources, instruments, control and monitoring system, visualization system, and users represented by user clients. The diagram also reflects that there may be different types of connecting network links: high-speed and low-speed which both can be permanent for the project or provisioned on-demand.

The figure also illustrates a typical use-case of a high-performance infrastructure, which is used by two or more cooperative research groups in different locations. In order to complete their task (e.g. cooperative image processing and analysis) they require a number of resources and services to process raw data on distributed Grid, Cloud or proprietary data centers, analyse intermediate data using specialised applications and finally deliver the resulting data to the scientists. This use-case includes all basic components of the typical e-Science research process: data collection, initial data mining and filtering, analysis with specialised scientific applications, and finally presentation and visualisation to the users.

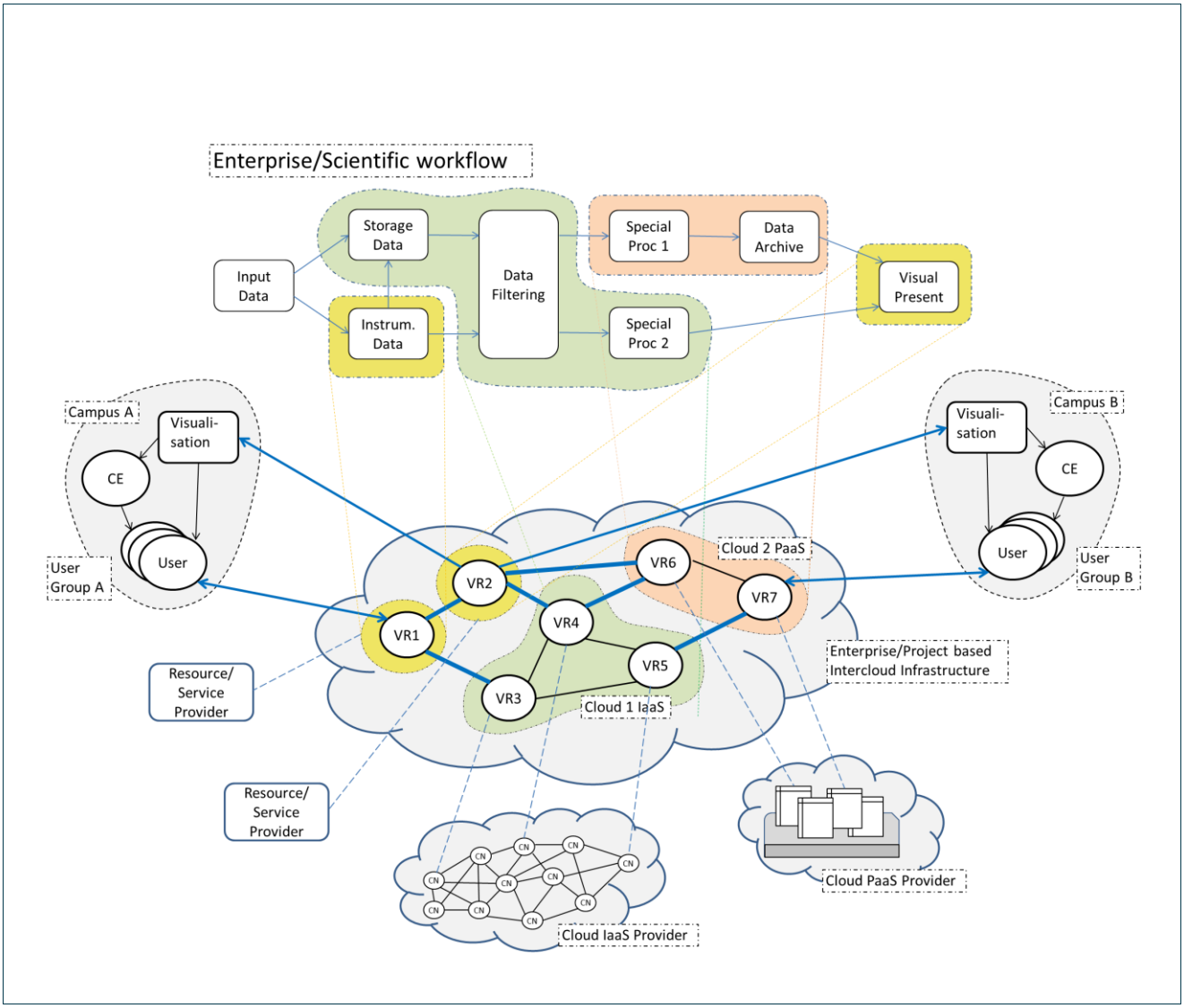


Figure 2.3: **Typical usecase for cloud based heterogeneous e-Science or enterprise infrastructure provisioning.**

2.4 ISOD Abstract Provisioning model

Figure 2.3 below illustrates the abstraction of the typical project or group oriented Virtual Infrastructure (VI) provisioning process that includes both computing resources and supporting network that commonly referred as infrastructure services [4, 5]. The VI is provisioned for two collaborative user groups in different locations that in order to fulfill their task (e.g. cooperative image processing and analysis) require a number of resources and services to process raw data on distributed Grid or Cloud data centers, analyse intermediate data on specialist applications and finally deliver the result data to the users/scientists. The discussed use case contains all basic components of the typical e-Science research process: data production with scientific instrument (labeled as VIR4 node), initial data mining and filtering (VIR3, VIR5), analysis with special scientific applications (VIR1, VIR6), and finally presentation and visualisation (VIR1, VIR6) to the users.

The figure also shows the main actors involved into this process, such as Physical Infrastructure Provider (PIP), Virtual Infrastructure Provider (VIP), Virtual Infrastructure Operator (VIO). The required supporting infrastructure services are depictured on the left side of the picture and includes functional components and services used to support normal operation of all mentioned actors.

The LICL (or Virtual Infrastructure Composition and Management (VICM)) layer includes the Logical Abstraction Layer and the VI/VR Adaptation Layer facing correspondingly lower PIP and upper Application layer. These layers represent information used by VIO/user applications to access VRI and support necessary logical transformation of the resources during composition and operation stages. VICM middleware is one of the key functionalities that enables all component services to interact, includes message processing functionality, middleware security, composition and orchestration services.

The proposed architectural framework for On-Demand Infrastructure Services provisioning (ISOD) comprises of the following main components [3, 4]: the Logical Infrastructure Composition Layer (also defined in [3, 4] as Composable Services Architecture (CSA)) that intends to provide a conceptual and methodological framework for developing dynamically configurable virtualised infrastructure services; the Infrastructure Services Modeling Framework (ISMF) that provides a basis for the infrastructure resources virtualisation and management, including description, discovery, modeling, composition and monitoring; the Service Delivery Framework (SDF) that provides a basis for defining the whole composable services life cycle management and supporting infrastructure services. Two cross-layer functionalities include Service Control and Management Plane (CMP) and Security Infrastructure described in this document.

The proposed architecture is a SOA (Service Oriented Architecture) based [7] and uses the same basic operation principle as known and widely used SOA frameworks, what also provides a direct mapping to the possible VICM implementation platforms such as Enterprise Services Bus (ESB) or OSGi framework [8,9].

The SDF introduced as a part of the proposed GEYSERS architectural framework (as being developed in [2, 3]) extends the proposed by the TeleManagement Forum the Service Delivery Framework as a part of the Software Enabled Services Management Solution [10, 11]. It includes the following main stages: (1) infrastructure creation request sent to VIO or VIP that may include both required resources and network infrastructure to support distributed user groups and/or applications; (2) infrastructure planning and advance reservation; (3) infrastructure deployment including services synchronization and initiation; (4) operation stage, and (5) infrastructure decommissioning. It combines/consolidates in one provisioning workflow all processes that are run by different supporting systems and executed by different actors.

The main infrastructure component to support SDF is the Service Lifecycle Meatadata Service (MD-SL) that provides necessary information to store/identify composed services identifiers, stages, versions and also bind this information to the SLA and provisioning sessions IDs.

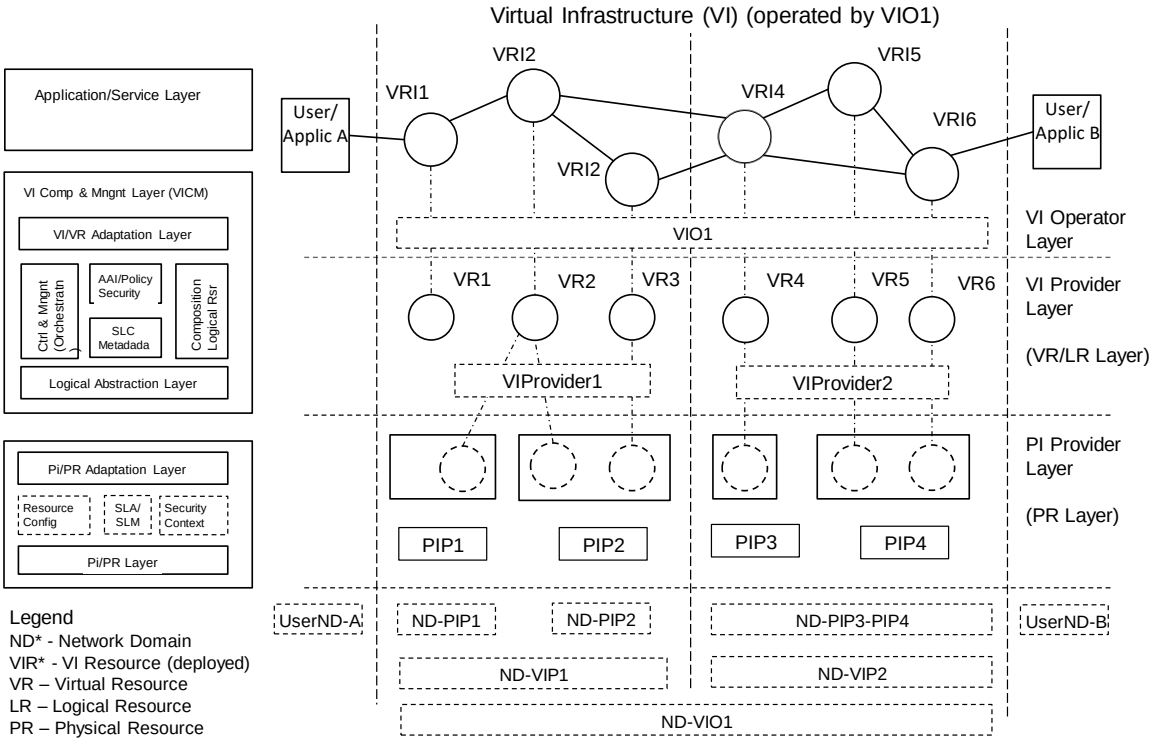


Figure 2.3. Main actors, functional layers and processes in on-demand infrastructure services provisioning

Physical Resources (PR), including IT resources and network, are provided by Physical Infrastructure Providers (PIP). In order to be included into VI composition and provisioning by the VIP they need to be abstracted to the Logical Resource (LR) that will undergo a number of abstract transformations, including possibly also interactive negotiation with the PIP. The composed VI need to be deployed to the PIP which will create virtualised physical resources (VPR) that may be a part or a pool of the resources provided by PIP. The deployment process includes distribution of common VI context, configuration of VPR at PIP, advance reservation and scheduling, and virtualised infrastructure services synchronization and initiation, to make them available to Application layer consumers.

The proposed abstract model allows outsourcing the provisioned VI operation to the VI Operator (VIO) who is from the user point of view provides valuable services of the required resources consolidation - both IT and networks, and takes a burden of managing the provisioned services.

The described model is being developed in the GEYSERS project [10] that targets to provide a generic architecture for Cloud Infrastructure as a Service (IaaS) provisioning model, allowing also to use and integrate other Clouds provisioning models for individual resources virtualisation.

The proposed architecture provides a basis and motivates development of the generalised framework for provisioning dynamic security infrastructure that includes Security Services Lifecycle Management model (SSLM), common security services interface (CSSI), and related security mechanisms to allow the consistency of the dynamically provisioned security services operation. The required security infrastructure should provide a common framework for operating security services at VIP and VIO layer and be integrated with PIP's legacy security services.

It is important to mention that discussed here physical and virtual resources are in fact complex software enabled systems with their own operational systems and security services. The VI provisioning process should support their smooth integration into the common federated VI security infrastructure allowing to define a common access control policies.

Access decision made at the VI level should be trusted and validated at the PIP level, what can be achieved by creating dynamic security associations during the provisioning process.

2.5 GEYSERS Service Delivery Framework (SDF)

The LICL operation relies on the well-defined services lifecycle management (SLM) model that is defined based on the TeleManagement Forum Service Delivery Framework (SDF) [10] that includes both the service delivery stages and required supporting infrastructure services.

Figure 2.4 illustrates the main service provisioning or delivery stages that address specific requirements of the provisioned on-demand virtualised infrastructure services:

Service Request Stage (including SLA negotiation). The SLA can describe QoS and security requirements of the negotiated infrastructure service along with information that facilitates authentication of service requests from users. This stage also includes generation of the Global Reservation ID (GRI) that will serve as a provisioning session identifier and will bind all other stages and related security context.

Composition/Reservation Stage that also includes **Reservation Session Binding** with the GRI, which provides support for complex reservation processes in multi-domain multi-provider environments. This stage may require access control and SLA/policy enforcement.

Deployment Stage, including services **Registration and Synchronisation**. The deployment stage begins after all component resources have been reserved and includes distribution of the common composed service context (including security context) and binding the reserved resources or services to the GRI as a common provisioning session ID. The Registration and Synchronisation stage (which can be considered as optional) specifically targets scenarios with provisioned service migration or re-planning. In a simple case the Registration stage binds the local resource or hosting platform run-time process ID to the GRI as a provisioning session ID.

Operation Stage (including Monitoring). This is the main operational stage of the provisioned on-demand composable services. Monitoring is an important functionality of this stage to ensure service availability and secure operation, including SLA enforcement.

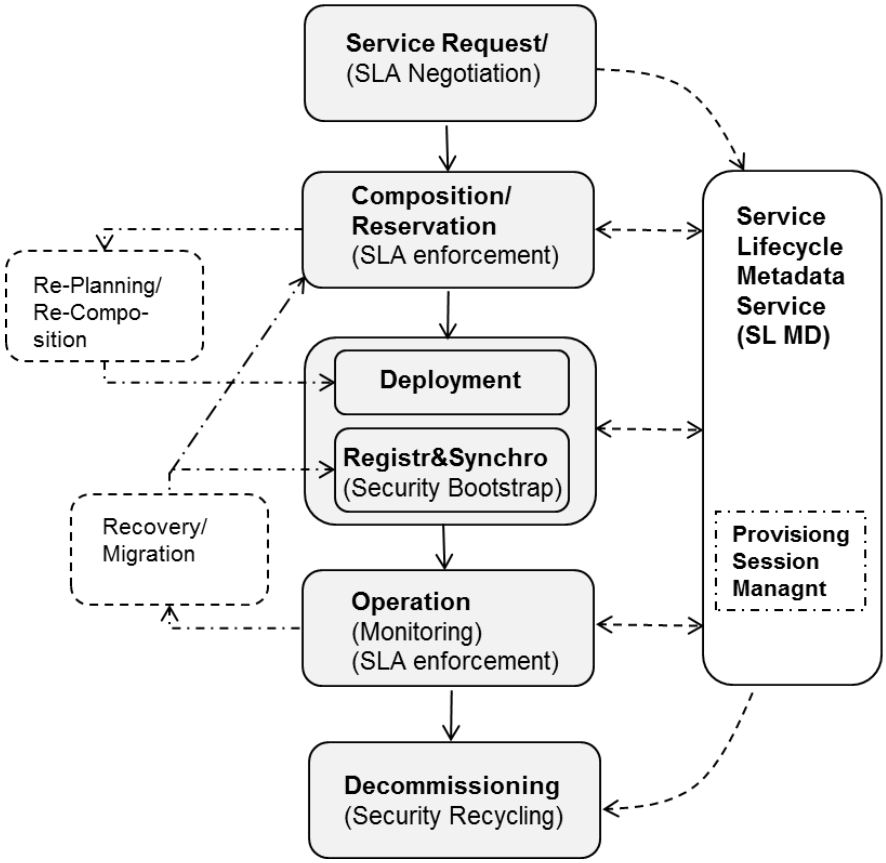
Decommissioning Stage ensures that all sessions are terminated, data is cleaned up, and session security context is recycled. The decommissioning stage can also provide information to or initiate service usage accounting.

Two additional (sub-)stages can be initiated from the Operation stage, based on the running composed service or component services state:

Re-composition or **Re-planning Stage** should allow incremental infrastructure changes.

Recovery/Migration Stage can be initiated by the user or the provider. This process can use MD-SLC to initiate a full or partial resource re-synchronisation, it may also require re-composition.

Implementation of the proposed SDF requires a special Service Lifecycle Metadata Repository (MD SLC as shown on Figure 2.3) to support consistent services lifecycle management. MD SLC keeps the services metadata that include at least service state, service properties, and services configuration information.



2.4?. GEYSERS Service Delivery Framework

3 General Security requirements for On-demand Infrastructure Services Provisioning

3.1 General Requirements to Security Services and Authorization Authentication Infrastructure

Providing consistent security services in GEYSERS architecture is of primary importance due to potentially multi-provider and multi-tenant nature of virtual infrastructures provisioned on-demand. The GEYSERS security architecture should address two aspects of the VI operation and dynamic security services provisioning:

- Provide security infrastructure for secure VI operation, including access control and SLA and policy enforcement for all interacting roles and components in VI and VIP/VIO, secure messaging and transport services.
- Provisioning dynamic security services, including creation and management of the dynamic security associations, as a part of the provisioned complex/composite services or virtual infrastructures.

The first task is a traditional task in security engineering, while dynamic provisioning of managed security services remains a problem and requires additional research.

The Security Services Lifecycle Management (SSLM) as an important issue on building consistent security services for dynamically provisioned virtual infrastructures is discussed below [sslm]. The SSLM extends the described above Geysers SDF service lifecycle management model and workflow with additional sub-stages and functions to bind dynamic security context to the general provisioning session and Cloud virtualisation and hosting platform in such a way that to ensure all operations on the virtual infrastructure and user data to be secured during their whole lifecycle.

The Geysers-Security Infrastructure (GSI) should provide the following basic infrastructure security services to ensure normal operation of the virtual infrastructure:

- Access control (e.g. Authentication, Authorization, Identity Management)
- Policy and SLA enforcement
- Trust management (including interdomain and inter-provider and dynamic security associations)
- Data, messaging and communication security
- Additionally, auditing/logging and accounting.

As a part of provisioned VI, the security solutions and supporting infrastructure should address the following problems, mostly related to data integrity and data processing security:

- Secure data transfer that should be enforced with data activation mechanism
- Protection of data stored on the virtualisation platform
- Restore from the process failure that entails problems related to secure job/application session and data restoration.

Initial suggestions to address those problems are based on the secure provisioning and application/job session management:

- Special session for data transfer that should also support data partitioning and run-time activation and synchronization.
- Secure job/session fail-over that should rely on the session synchronization mechanism when restoring the session.
- Session synchronization mechanisms that should protect the integrity of the remote run-time environment.

The following problems/challenges arise from the GEYSERS provisioning environment analysis for security services/infrastructure design:

- Data protection both stored and “on-wire” that include beside necessary confidentiality, integrity, access control services, also data lifecycle management and synchronization.
- Access control infrastructure virtualisation and dynamic provisioning, including dynamic/automated policy composition or generation.
- Security services lifecycle management, in particular related services metadata and properties, binding to main services.
- Security sessions and related security context management during the whole security services lifecycle, including binding security context to the provisioning session and virtualisation platform.
- Dynamic security associations (DSA) and trust/key management, including trust anchor bootstrapping during deployment stage, what should provide fully verifiable chain of trust from the user client/platform to the service/data runtime environment.
- SLA management, including initial SLA negotiation and further SLA enforcement at the planning and operation stages.

Initial suggestions to address those problems require the consistent secure provisioning and application sessions management, in particular:

- Special session for data transfer that should also support data partitioning and run-time activation and synchronization.
- Session synchronization mechanisms that should protect the integrity of the remote run-time environment.
- Secure session fail-over that should rely on the session synchronization mechanism when restoring the session.
- Standardized interfaces that will answer some of user concerns on cloud security.

Successful GEYSERS architecture adoption by industry and its integration with advanced infrastructure services will require implementing manageable security services and mechanisms for the remote control of the provisioned infrastructure operational environment integrity by users.

GYESERS-Security should implement multi-layer security services including transport, messaging and application/data security, and additionally network layer security for distributed VPN based enterprise domains. Security and security services in the GEYSERS architecture design are applied at different layers and can be called from different functional components using standard/common security services interface. Security services are governed by related security policies.

Security services can be designed as: pluggable services operating at the messaging layer; OSGi bundles that can be dynamically added as composable services to other composable services to form an instant virtual infrastructure; or exposed as Web services and be integrated with other services by means of higher level workflow management systems.

4 Existing Security Frameworks and Platforms

4.1 Role Based Access Control

Although RBAC is technically a form of non-discretionary access control, it is often considered as one of the three primary access control policies (the others are DAC and MAC). In RBAC, access decisions are based on the roles that individual users have as part of an organization. Users take on assigned roles (such as professor, student, operator, or manager). Access rights are grouped by role name, and the use of resources is restricted to individuals authorized to assume the associated role. The use of roles to control access can be an effective means for developing and enforcing enterprise-specific security policies and for streamlining the security management process.

Under RBAC, users are granted membership into roles based on their competencies and responsibilities in the organization. The operations that a user is permitted to perform are based on the user's role. User membership into roles can be revoked easily and new memberships established as job assignments dictate. Role associations can be established when new operations are instituted, and old operations can be deleted as organizational functions change and evolve. This simplifies the administration and management of privileges; roles can be updated without updating the privileges for every user on an individual basis.

Generic RBAC model [15, 16, 17] provides an industry recognised solution for effective user roles/privileges management and policy based access control. It extends Discretional Access Control (DAC) and Mandatory Access Control (MAC) models with more flexible access control policy management adoptable for typical hierarchical roles and responsibilities management in organisations, but at the same time it suggest a full user access control management from user assignment to granting permissions. This can be suitable for internal organisational environment and particularly for human access rights management but reveals problems when applied to distributed service-oriented environment.

Sandhu in his two research papers [15, 16] describes 4 basic RBAC models:

- Core RBAC (RBAC0) that associates Users with Roles (U-R) and Roles with Permissions (R-P);
- Hierarchical RBAC (RBA1) that adds hierarchy to roles definition;
- Constrained RBAC (RBAC2) that extends RBAC0 with the constrains applied to U-R and R-P assignment;
- Consolidated RBAC (RBAC3) that adds role hierarchy to RBAC2.

RBAC is described in the ANSI INCITS 359-2004 standard [9] that partly re-defined the first three basic RBAC models in the context of static or dynamic separation of duties (SSD vs DSD). In both models, initial Sandhu's and ANSI RBAC, there is a notion of the user session which is invoked by a user and provides instant session-based U-R association. Final result/stage of the RBAC functionality are permissions assigned to the user based on static or dynamic U-R and R-P assignment. RBAC doesn't consider (user) permissions enforcement on the resource or access object. This functionality can be attributed to other more service-oriented frameworks such as ISO/ITU X.811/X.812 Authentication/Authorization framework [18, 19] or generic AAA Authorization framework [20, 21].

4.2 Generic AAA Authorization Framework

Authentication, authorization, and accounting (AAA) is a term used to refer to a framework for intelligently controlling access to computer resources, enforcing policies, auditing usage, and providing the information necessary to bill for services. These combined functions are considered important for effective network management and security.

The generic Authentication, Authorization, Accounting (AAA) architecture was proposed in RFC2903 [20] and generic AAA Authorization framework (GAAA-AuthZ) is described in RFC2904 [21] as a development of the ITU-T X.812 Authorization framework [19] for distributed multidomain systems.

Authentication (AuthN) and Authorization (AuthZ) are the components of the access control function to ensure that access to a resource or service is granted to the access subject (human, service or process) that has right to use the resource and perform those operation on the resource that it is allowed.

Authentication is the process of identifying a user or an access subject, based on identity credentials which examples are username and password, digital certificates, one-time-tokens, etc. Authentication refers to the confirmation that a user/subject who is requesting services is a valid user of the resources or services requested. Typically AuthN involves comparing a user's authentication credentials with the user credentials stored in a user database (UserDB) or the AuthN/AAA service, or checking validity of the user credentials obtained from the trusted AuthN service or trusted Identity Provider.

Based on positive AuthN, a user must obtain authorization for doing certain tasks. Authorization is the process of granting or denying a user access to network resources once the user has been authenticated. The amount of information and the amount of services the user will be granted depends on the user's authorization level which is defined by the user attribute credentials. In other words, Authorization is the process of enforcing policies: determining what types or qualities of activities, resources, or services a user is permitted. Usually, authorization occurs within the context of authentication. Authenticated user is provided with the attributes that are required for authorization decision.

Accounting is the process of keeping track of a user's activity while accessing the resources or services. Accounting is carried out by logging of session statistics and usage information and used for trend analysis, capacity planning, billing, auditing and cost allocation.

In modern Service Oriented Architecture (SOA) applications a Resource or a Service are protected by the site access control system that relies on both AuthN of the user and/or request message and AuthZ that applies access control policies against the service request. It is essential in a service-oriented model that AuthN credentials are presented as a security context in the AuthZ request and that they can be evaluated by calling back to the AuthN service and/or Attribute Authority (AttrAuth). This also allows for loose coupling of services in distributed hierarchical access control infrastructure.

The GAAA-AuthZ model is illustrated on Figure 4.1 and includes such major functional components as: Policy Enforcement Point (PEP), Policy Decision Point (PDP), Policy Authority Point (PAP). It is naturally integrated with the RBAC separated User-Role and Role-Privilege management model that can be defined and supported by separate policies.

The Requestor requests a service by sending a service request ServReq to the Resource’s PEP providing as much (or as little) information about the Subject/Requestor, Resource, Action as it decides necessary according to the implemented authorization model and (should be known) service access control policies.

In a simple scenario, the PEP sends the decision request to the (designated) PDP and after receiving a positive PDP decision relays a service request to the Resource. The PDP identifies the applicable policy or policy set and retrieves them from the Policy Authority, collects the required context information and evaluates the request against the policy.

In order to optimise performance of the distributed access control infrastructure, the Authorization service may also issue AuthZ assertion in the form of AuthzTicket that confirm access rights. They are based on a positive decision from the Authorization system and can be used to grant access to subsequent similar requests that match an AuthzTicket. To be consistent, AuthzTicket must preserve the full context of the authorization decision, including the AuthN context/assertion and policy reference.

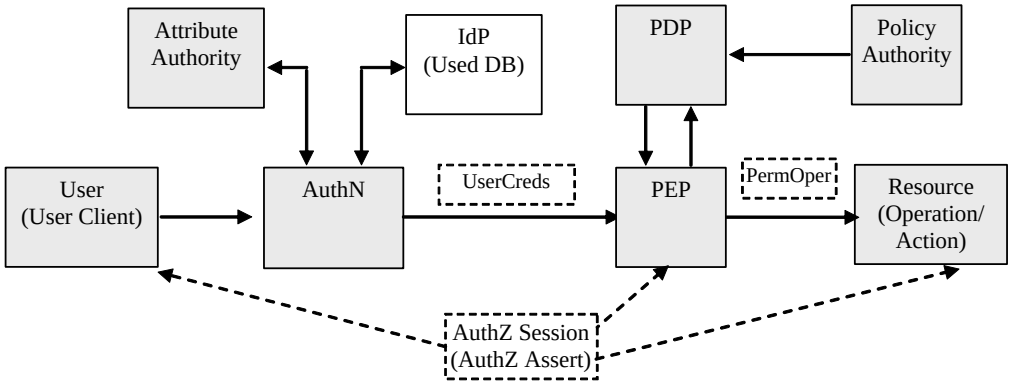


Figure 4.1. Generic Authentication and Authorization services interaction.

Generic AAA Authorization Framework defines three basic operational models that describe interaction (in sense of request/response sequences) between a user, a service or resource provider and AAA Authorization service acting as an Authority:

The push authorization sequence. Within the push (or token-) sequence, the User first requests an authorization from a trusted Authorization service that may or may not honor the User's request. It then may issue and return some kind of Authorization assertion (a secured ticket or token) that acts as a proof of right or as asserted list of requestor capabilities. Typically such an assertion has an associated validity time window. The assertion may subsequently be used by the User to request a specific service by contacting the Resource. The Resource will accept or reject the authorization assertion and will report this back to the requesting Subject. The Resource must have been provisioned with the appropriate key material to recognize the appropriate assertions.

The pull authorization sequence. Within the pull (or outsource-) sequence, the User will contact the Resource with a request. Before admitting the service request, the Resource must contact its Authorization service. The Authorization service will evaluate the request against a specific authorization policy and will return an authorization decision. The Resource will subsequently grant or deny the service to the User by returning a result message. The Resource, which enforces a policy, effectively out-sources a policy decision.

The agent authorization sequence. Using the agent (or provision-) sequence, the User will contact an Agent, which will handle the User's request for the particular Resource. The Agent is trusted both by the User and the Resource. The Agent will make an authorization decision and, using its own or User-delegated credentials, it will contact the Resource to provision the requested service. The Agent will provide the User with details on how to contact and use the Service.

The three basic authorization sequences described above are elementary abstractions of more complex real world examples that normally combine the basic sequences. It may use various protocols and message formats to handle and secure user credentials and requests.

Although more functions can be found in both an Authority and a Resource, an Authority typically acts as a Policy Decision Point (PDP) and a resource incorporates a Policy Enforcement Point (PEP) which used to call for the policy decision to the Authority and enforce already made decision. In the subsequent discussion we may use the term PDP and PEP to represent functions inside the corresponding entities.

4.3 Dynamic Access Control Services in existing Cloud IaaS platforms

Clouds technologies [6] are emerging as infrastructure services for provisioning computing and storage resources on-demand in a simple and uniform way. However there is no well-defined architectural model for the Cloud Infrastructure a Service (IaaS) provisioning model despite its wide use among big Cloud providers such as Amazon, RackSpace, Google, and others. Recent research based on the first wave of Cloud Computing implementation have revealed a number of security issues both in actual services organisation and operational and business models [27, 28]. Current Clouds security model is based on the assumption that the user/customer should trust the provider. This is governed by the general Service Level

Agreement (SLA) that defines mutual provider and user expectations and obligations for the whole provisioned services but doesn't allow dynamic Quality of Services (QoS) management in potentially changing resources availability due to changing resources demand and utilisation in typically multi-user Cloud environment.

Although Cloud provider invested a lot into making their own infrastructure secure and complying existing security management standards (e.g. Amazon Cloud recently achieved PCI compliance certification [29]), still the overall security of the Cloud based applications and services will depend on two other factors: security services implementation in user applications and binding between virtualised services and Cloud based virtualisation platform, that should also ensure protection against malicious users and risks related to possible Denial of Service (DoS) attacks.

Practical Cloud usage within one provider infrastructure brings illusion about unlimited availability, "elasticity" and "perfect" security, but in practice this is related only to limited range of services and with limited manageability. Currently implemented and provided security are based on VPN and provide only simple access control services based on users access over SSH channel. More advanced security services and fine grained access control cannot be achieved without deeper integration with the Cloud virtualisation platform and incumbent security services, what in its own turn can be achieved with open and well defined Cloud IaaS platform architecture.

More complex and community oriented use of Cloud infrastructure services will require developing new service provisioning and security models that could allow creating complex project and group oriented infrastructures provisioned on-demand and across multiple providers.

4.3.1 Amazon AWS Security

Regarding access control services for on-demand infrastructure, there are several existing works such as Amazon AWS Identity and Access Management (IAM) for Amazon Cloud products [30], the Access Control Service in the Windows Azure AppFabric [31].

The Amazon AWS IAM is the integration of an Identity Management System and an Access Control System. On reserving an Amazon AWS product, each customer is assigned an AWS account. Operations on AWS products are binded to this account. Amazon IAM provides a mechanism to create and manage multiple users binding to the customer's AWS account. Using rules and policies at IAM side and at AWS product side, the IAM could control users' activities on AWS resources. To guarantee security requirements on confidentiality and integrity, users have their own security credentials for accessing AWS resources.

Although Amazon AWS IAM is suitable for Amazon AWS products in identity and access management, it's still rigid in trust establishment and not flexible for multi-security domains and multi-tenancies while there is only one provider role for Amazon AWS products. Amazon plays as a PIP to provide individual virtualized resources such as EC2 or S3 and also a VIP to integrate such virtualized resources together. The access control model in Amazon AWS IAM is not well supported for complex organizations because it only manages users in groups and performs authorization based on assigned permissions to groups. Many other features of Role-based Access Control model [17] are not present in Amazon AWS IAM.

4.3.2 Access Control Service for Windows Azure Cloud platform

Access Control Service for Windows Azure AppFabric [Azure] is one of middleware services for applications in Microsoft Azure Platform as in Figure 4.3:

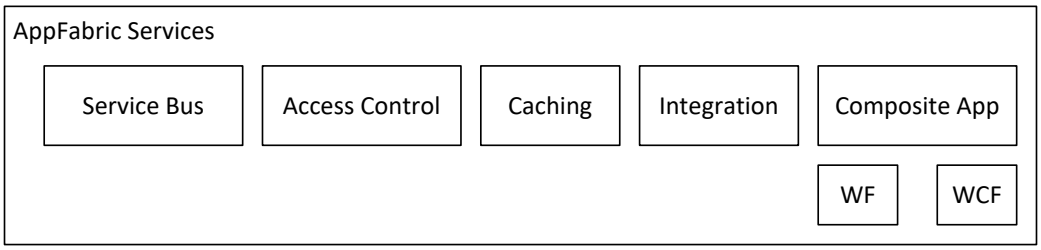


Figure 4.3 – Microsoft Azure AppFabric Services

This service enables authorization decisions are separated from regular applications and their clients to delegate to an external access control engine. It has many notice features such as federation identity in access control, supports multiple credentials, flexible and light-weight developer friendly programming model. AppFrabric Access Control plays the role in Windows Azure Platform as the intermediate trust-party between user side and service side as below:

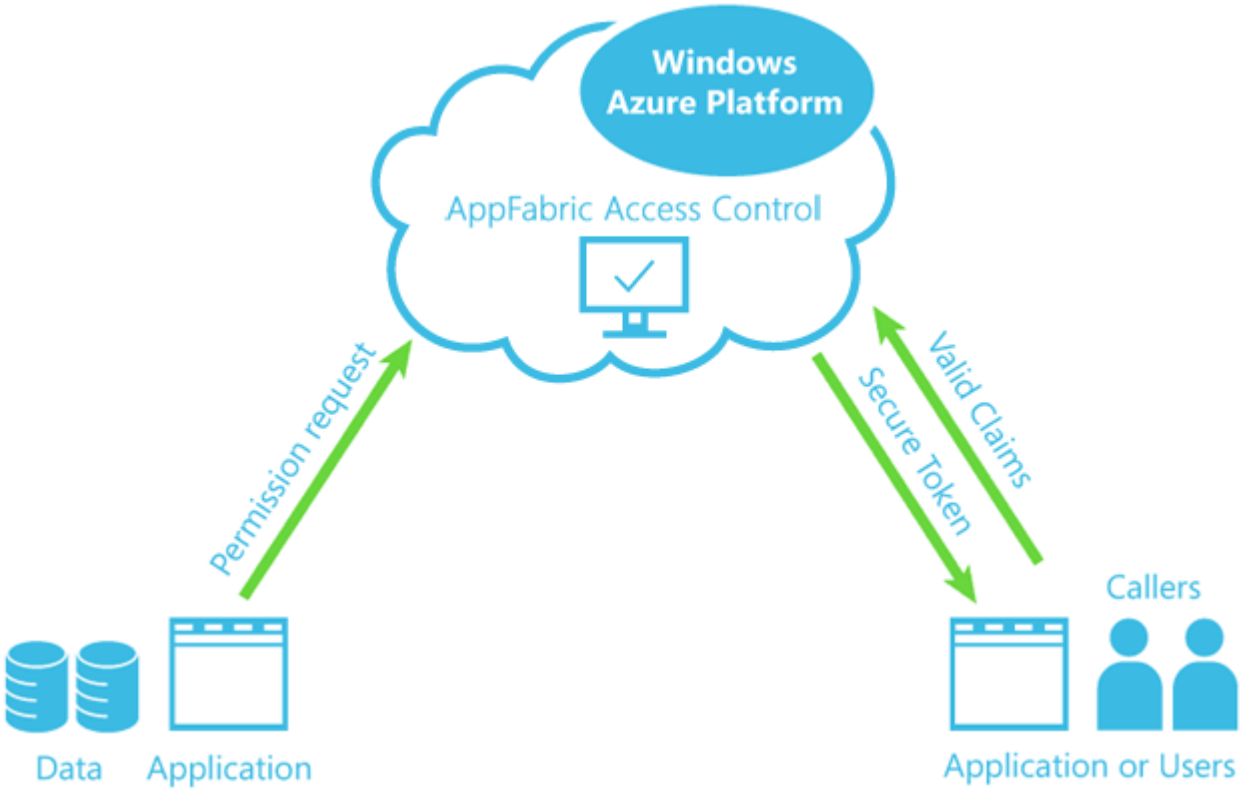


Figure 4.3 – AppFabric Access Control for Microsoft Azure Platform (the figure from Microsoft Azure)

However, AppFabric Access Control (AC) has some limitations. Although it's flexible to operate and provide access control service in federated identity environment, AC is not support for complex on-demand provisioning services, in which the composite service could be assembled parts from legacy services. And because of not supporting Service Lifecycle Management, AC couldnot dynamically establish trust relationships between user-side and a provisioned resource at service side. Hence, this solution also does not adapt access control requirements in GEYSERS.

5 Security Architecture

5.1 Multi-Layer Security Services

There are four main aspects what concern to security that the LICL must handle. First, there must be an access control over the resources, both virtual and physical ones, and also at VI level; access control will be obtained via authentication and authorization mechanisms. Secondly, the data has to be protected, implying that data traffic remains isolated between VI, as well as, stored data is not accessible from others VIs, independently they are allocated over the same physical resource. Third, security has to facilitate policy enforcement, assuring that VI usage does not affect on the performance of other VIs. These two last aspects relate to the isolation capability between VRs. Finally, LICL has to provide security on the service provisioning process as well.

Security is considered a cross-layer functionality as it affects components from different layers, like virtual infrastructures, or physical resources.

5.2 Authentication and Authorization Infrastructure

Developing a consistent framework for dynamically provisioned security services requires deep analysis of all underlying processes and interactions. Many processes typically used in traditional security services infrastructures need to be abstracted, decomposed and formalized. First of all, it is related to the security services setup, configuration and security context management that in many present solutions/frameworks is provided manually, during the service installation or configured out-of-band.

The general security framework for on-demand provisioned infrastructure services should address two general aspects [32]: (1) supporting an access control architecture for multi-providers to provide on-demand provisioning services, and (2) provisioning a Dynamic Access Control Infrastructure (DACI) as part of the provisioning on-demand virtual infrastructure. The first task primarily focuses on the access control solution supporting on-demand provisioning resources with security contexts synchronizationand management over multi-domains. The DACI must be bootstrapped to the provisioned on-demand VI and VIP/VIO trust domains as entities participating in the handling initial request for VI and legally and securely bound to the VI users. Such security bootstrapping can be done at the deployment stage.

Virtual access control infrastructure setup and operation is based on the mentioned DSA that links the VI dynamic trust anchor(s) with the main actors and/or entities participating in the VI provisioning – VIP and the requestor or target user organisation (if they are different). As discussed above, the creation of such DSA for the given VI can be done during the reservation and deployment stages. Reservation stage allows to distribute the initial provisioning session context and collects the security context (e.g. public key certificates) from all participating infrastructure components. The deployment stage can securely distribute either shared cryptographic keys or another type of security credentials that will allow validating information exchange and apply access control to VI users, actors, services.

Figure 5.1 illustrates in details interactions between main actors and access control services during the reservation stage and also includes other stages of provisioned infrastructure lifecycles. The request to create VI (RequestVI) initiates a request to VIP that will be authorized by VIP-AAI against its regular access control policies, what next will be followed by VIP requests to PIPs for required or selected physical resources PR's, which in its own turn will be authorized by PIP-AAIs. The SDF and SSLM requirements show that the initial RequestVI all as well as communication and access control evaluations should be bound to the provisioning session identifier GRI. The chain of requests from the User to VIO, VIP and PIP can also carry corresponding trust anchors TA0…TA2, e.g. in a form of public key certificate (PKC) [33] or WS-Trust security tokens [34].

DACI is initialized at the deployment stage to controls accesses and activities on the VI resources. The DACI bootstrapping can be done either by fully pre-configuring trust relations between VIP-AAI and DACI or by using special bootstrapping

registration procedure similar to those used in TCPA [35], or use the dynamic trust establishment protocols for multi-providers scenarios [67]

To ensure unambiguous session context and all involved entities and resources identification the following types of identifiers are used:

- Global Reservation ID (GRI) – generated at the beginning of the VI provisioning, stored at VIO and returned to User as identification of the provisioning session and the provisioned VI.
- VI-GRI – generated by VIP as an internal reservation sessions ID, which can be also re-folded GRI, depending on VIP provisioning model.
- PR-LRI and VR-LRI – provide identification of the committed or created PR@PIP and VR@VIP.

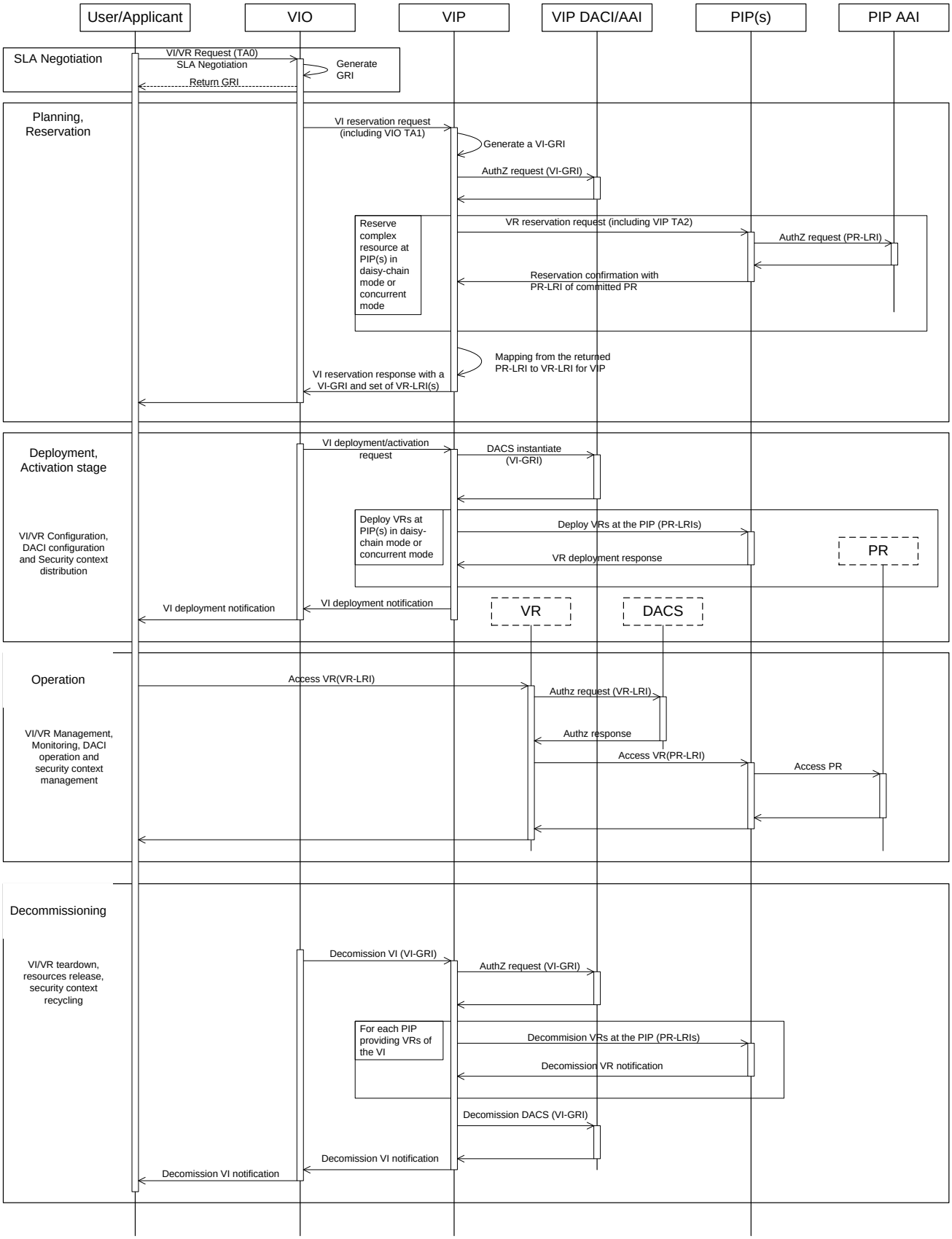


Figure 5.1: Dynamic Access Control Infrastructure during VI Provisioning and Operation

5.3 Security Services Lifecycle Management Model

Most of the existing security lifecycle management frameworks, such as defined in the NIST Special Publication 800-14 "Generally Accepted Principles and Practices in Systems Security" [36], provide a good basis for security services development and management, but they still reflect the traditional approach to services and systems design driven by engineers force. The defined security services lifecycle includes the following typical phases: Initiation, Development/Acquisition, Implementation, Operation/Maintenance, and Disposal.

Figure 5.2 (b) illustrates the proposed Security Services Lifecycle Management (SSLM) model [37] that reflects security services operation in generically distributed multidomain environment and their binding to the provisioned services and/or infrastructure. The SSLM includes the following stages:

- **Service Request** and generation of the GRI that will serve as a provisioning session identifier (SessionID) and will bind all other stages and related security context. The Request stage may also include SLA negotiation which will become a part of the binding agreement to start on-demand service provisioning.
- **Reservation stage** and Reservation session binding that provides support for complex reservation process including required access control and policy enforcement.
- **Deployment stage** begins after all component resources have been reserved and includes distribution of the security context and binding the reserved resources or services to the Global Reservation ID (GRI) as a common provisioning session ID.
- **Registration&Synchronisation stage** (that however can be considered as optional) that specifically targets possible scenarios with the provisioned services migration or failover. In a simple case, the Registration stage binds the local resource or hosting platform run-time process ID to the GRI as a provisioning session ID.
- During **Operation stage** the security services provide access control to the provisioned services and maintain the service access or usage session.
- **Decommissioning stage** ensures that all sessions are terminated, data are cleaned up and session security context is recycled.

The proposed SSLM model extends the existing SLM frameworks and earlier proposed by authors the GAAA-NRP model [26] with the new stage "Registration & Synchronisation" that specifically targets such security issues as the provisioned services/resources restoration (in the framework of the active provisioning session) and provide a mechanism for remote data protection by binding them to the session context.

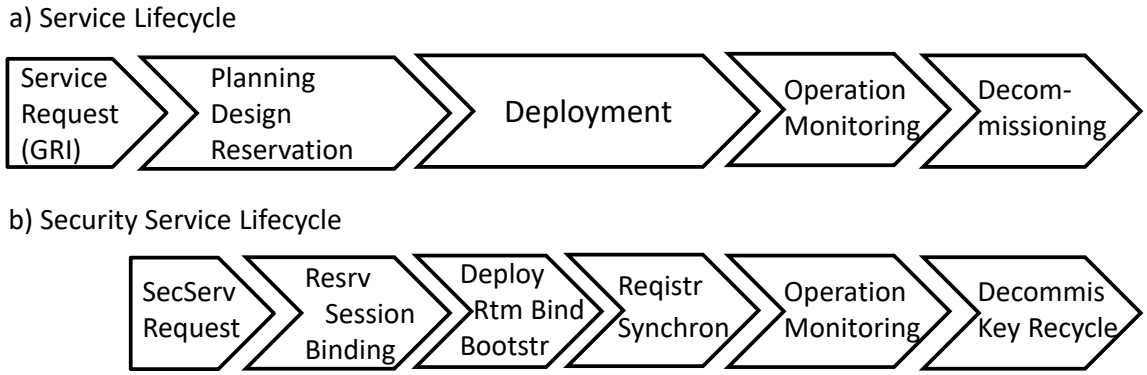


Figure 5.2: The proposed Security Services Lifecycle Management model.

Table A explains what main processes/actions take place during the different SLM/SSLM stages and what general and security mechanisms are used:

- SLA – used at the stage of the service Request placing and can also include SLA negotiation process.
- Workflow is typically used at the Operation stage as service Orchestration mechanism and can be originated from the design/reservation stage.
- Metadata are created and used during the whole service lifecycle and together with security services actually ensure the integrity of the SLM/SSLM.
- Dynamic security associations support the integrity of the provisioned resources and are bound to the security sessions.
- Authorization session context supports integrity of the authorization sessions during Reservation, Deployment and Operation stages.
- Logging can be actually used at each stage and essentially important during the last 2 stages – Operation and Decommissioning.

Table 5.1. Relation between SSLM/SLM stages and supporting general and security mechanisms

SLM stages	Request	Design/Reserv. Development	Deployment	Operation	Decomissioning
Process/ Activity	SLA Negotiation	Service/ Resource Composition Reservation	Composition Configuration	Orchestration/ Session Management	Logoff Accounting
Mechanisms/Methods					
SLA	**M**				**M**
Workflow		O		**M**	
Service Lifecycle Metadata	**M**	**M**	**M**	**M**	
Dynamic Security Associatn		O	**M**	**M**	
AuthZ Session Context		**M**	O	**M**	
Logging		O	O	**M**	**M**

Legend:

M – Mandatory; O - Optional

5.4 Security context management in VI resources provisioning

5.4.1 Session types and security context

VI authorization session in LICL is based on general SDF model that includes stages such as reservation, deployment, access/operation, and decommission [38] as in Figure 5.. It is necessary to enforce access control policies at the beginning of each step.

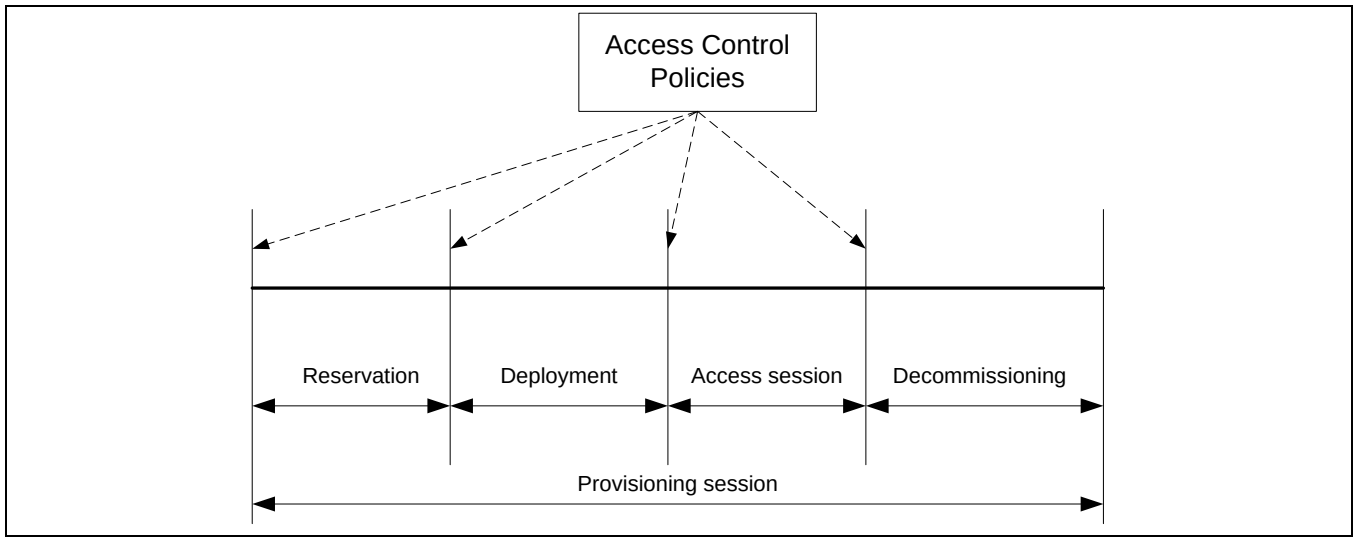


Figure 5.3: Virtual Infrastructure life cycle session stages

To achieve consistent security services in dynamically created virtualised resources and infrastructure in general, it is required that resources lifecycle information/data should have sufficient security context information as described below.

From reservation stage each VI instance has a unique identifier value to distinguish among VIs through its life cycles. This identifier value is called Global Reservation Identifier for VI (VI-GRI). VI-GRI should be generated at the beginning of provisioning session at the VI request side (SML) or VI management side (upper-LICL layer). To correctly apply security services, upper-LICL keeps VI related security information as the metadata in the "Service Lifecycle Metadata Repository". This metadata is called "VI security context".

Lower-LICL layers are implemented at PIPs for VR abstraction and management. Each VR object is identified by a unique Local Reservation Identifier (VR-LRI) which is generated at the VR reservation stage. Similar to upper-LICL, the lower-LICL also need to keep VR related security information in the "Service Lifecycle Metadata Repository" as the "VR security context".

The general security context must contain following information:

- Session identifier: this is the unique value for identification. It could be get the value or derived from VI-GRI when the context for VI or LRI when the context for VR.
- Session condition: set of conditions and obligations for the resource object (e.g.: validity time, conditions implied by the previous policy decisions).
- Resource information or reference: contains a set of the resource attributes required for enforcing security policy. For the VI, it could be set of VRs and their related attributes, including resource lifecycle stage. For the VR, it could be VR attributes using to access a concrete PR at the PIP. Resource attributes included into the security context object must be unambiguously linked to the full resource description, e.g. via GRI or LRI.

- KeyInfo: contains related information on cryptographic keys used for security operations. When the context for VI, they could be sharing keys between VIO and VI. When the context for VR, it could contain cryptographic keys for trust relations between VIO and the VR or between the VR with others.

5.4.2 Using Authorization tokens for security context management

Although DACI operates at the Operation stage, its security context is bound to the overall provisioning process starting from SLA negotiation that will provide a trust anchor TA0 to User/application security domain with which the DACI will interact during the Operation stage. The RequestVI initiates the provisioning session inside of which we can also distinguish two other types of sessions: reservation session and access session (the deployment session is used only for control and management purposes in the services provisioning), which however can use that same access control policy and security context management model and consequently can use the same format and type of the session credentials. In the discussed DACI we re-use the AuthZ tokens (AuthzToken) mechanism initially proposed in the GAAA-NRP and used for authorization session context management in multi-domain network resource provisioning [25, 26]. Tokens as session credentials are abstract constructs that refer to the related session context stored in the provisioned resources or services. The token should carry session identifier, in our case GRI or VI-GRI.

When requesting VI services or resources at the operation stage, the requestor need to include the reservation session credentials together with the requested resource or service description which in its own turn should include or be bound to the provisioned VI identifier in a form of GRI or VI-GRI. The DACI context handling service should provide resolution and mapping between the provided identifiers and those maintained by the VIP and PIP, in our case VR-LRI or PR-LRI. If session credentials are not sufficient, e.g. in case when delegation or conditional policy decision is required, all session context information must be extracted from AuthzToken and the normalised policy decision request will be sent to the DACI Policy Decision Point (PDP) which will evaluate the request against the applied access control policy.

In the discussed DACI architecture the tokens are used both for access control and signalling at different SSLM/SDF stages as a flexible mechanism for communicating and signalling security context between administrative and security domains (that may represent PIP or individual physical resources). Inherited from GAAA-NRP the DACI uses two types of tokens:

- Access tokens that are used as AuthZ/access session credentials and refer to the stored reservation context.
- Pilot tokens that provide flexible functionality for managing the AuthZ session during the Reservation stage and the whole provisioning process.

Figure 5.4 illustrates the common data model of both access token and pilot token. Although the tokens share a common data-model, they are different in the operational model and in the way they are generated and processed. When processed by the AuthZ service components they can be distinguished by the token type attribute which is optional for access token and mandatory for pilot token.

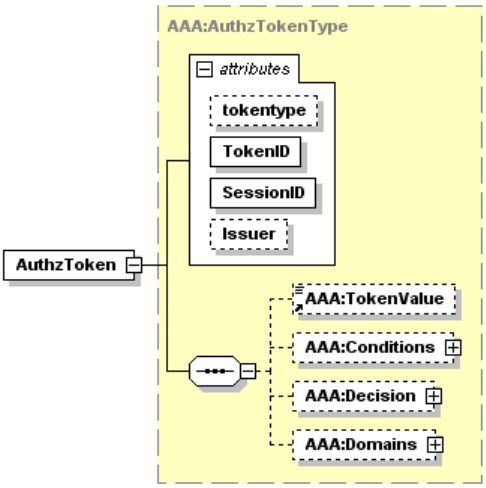


Figure 5.4. Common access and pilot token datamodel.

Access tokens contain three mandatory elements: the SessionId attribute that holds the GRI; the TokenId attribute that holds a unique token ID attribute and is used for token identification and authentication; and the TokenValue element. The optional elements include: the Condition element that may contain two time validity attributes notBefore and notOnOrAfter; the Decision element that holds two attributes ResourceId and Result; and optional element Obligations that may hold policy obligations returned by the PDP. Pilot token may contain another optional Domains element that serves as a container for collecting and distributing domain related security context.

For the purpose of authenticating token origin, the pilot token value is calculated of the concatenated strings “DomainId, GRI, TokenId”. This approach provides a simple protection mechanism against pilot token duplication or replay during the same reservation/authorization session. The following expressions are used to calculate the TokenValue for the access token and pilot token:

```
TokenValue = HMAC(concat(DomainId, GRI, TokenId), TokenKey)
```

6 AAI Components

The GEYSERS AAI Authentication and Authorization Servers have following components:

- **Authentication and Identity Management Service:** this server provides authentication service,issues and verifies attribute statements binding to authenticated subjects using SAML profile [39]
- **Authorization Service:** provides the authorization service compliant with SAML-XACML profile [40]
- **Token Validation Service:** performs token verifications on AuthZ tokens

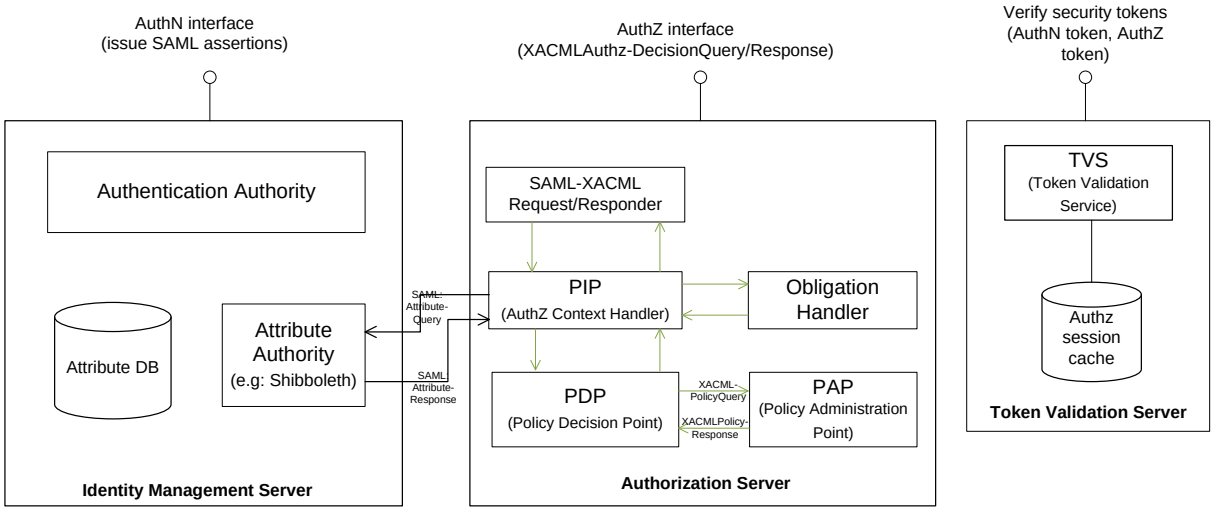


Figure 6-1 Authentication and Authorization Infrastructure components

6.1 Identity Management Service

The Identity Management Service has two tasks:

- Authentication: authenticates subjects based on their submitted credentials. There are several credential types, such as: username/password, X.509 certificates.

- Issue authentication tokens (authn-token): the Identity Management Server may issue an authn-token to the authenticated subject. The authn-token could be a standard token: SAML authentication assertions [SAML2] or Keberos tickets. The issuer identifier of these token is the Identity Management Server.

The authn-token could also be verified at for their lifetime and content validities

The Identity Management Server could be utilized from existing Authentication Authority and Attribute Authority such as Shibboleth [41].

6.2 Authorization Service

Authorization Service is built upon the pluggable GAAA-TK library [25] which follows the generic Authentication, Authorization and Accounting (AAA) framework (GAAA-AuthZ) [21]. The purpose of Authorization Service is to grant or deny actions under an authenticated subject . The authorization policies are composed using XACML standard [39].

The authorization interface is in compliance with SAML profile of XACML [40] in which authorization requests and responses are XACMLAuthzDecisionQuery and XACMLAuthzDecisionResponse.

The Authorization Server may issue "XACMLAuthzDecision Assertion" as the authorization token for a request from PEP. The content and usage recommendations of XACMLAuthzDecision Assertion are specified in [40].

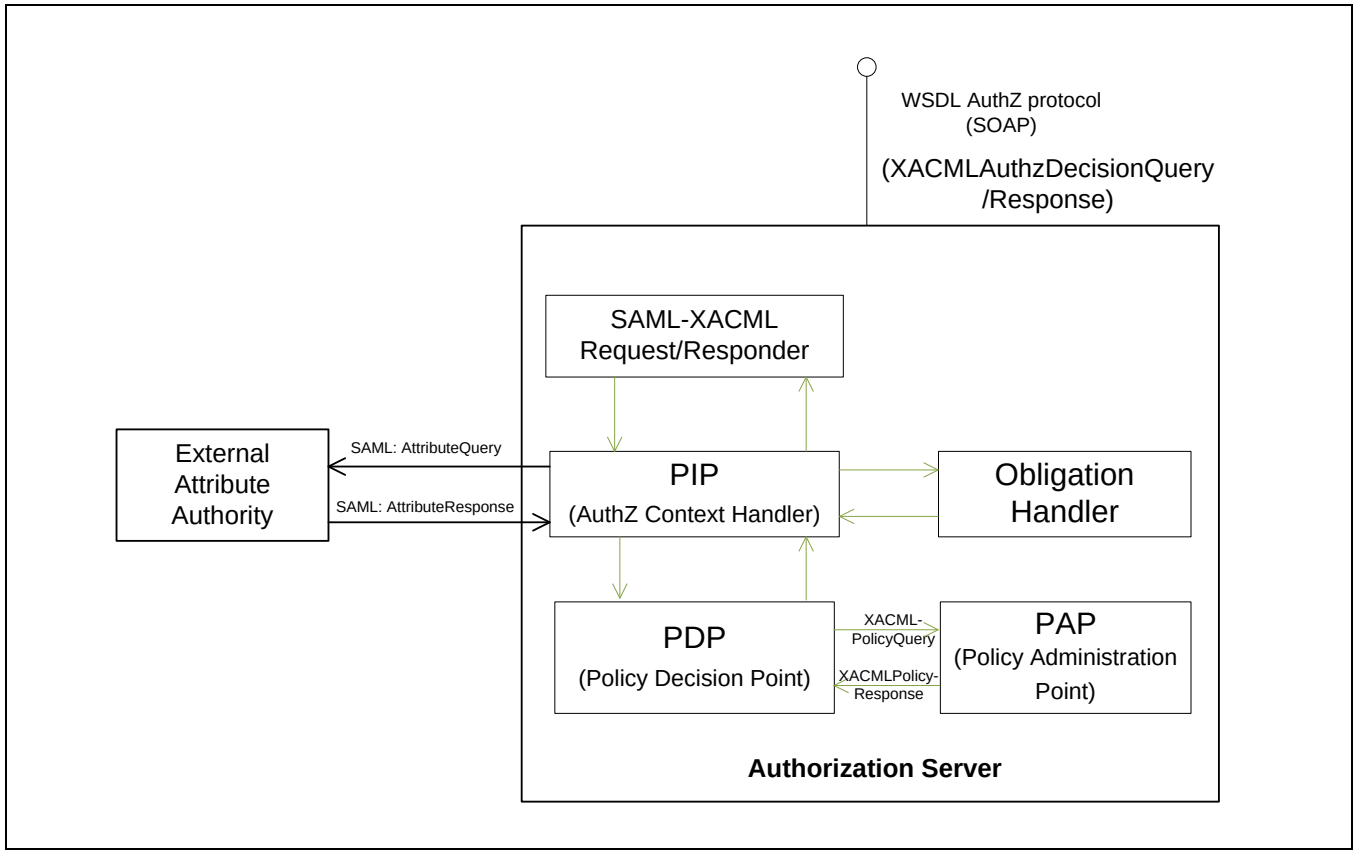


Figure 6-2: Authorization server

Components in the Authorization Servers have the following functionalities:

- SAML-XACML Request/Responder: handles SAML messages carrying XAML authorization requests and responses.

- Policy Information Point (PIP): collect necessary attributes that provides to PDP for authorization policy evaluations
- Policy Decision Point (PDP): evaluate authorization requests against set of XACML policies provided by PAP.
- Policy Administration Point (PAP): provide policies to the PDP using SAML protocol in carrying policy requests and policy responses.

6.3 Token Validation Service

The aim of Token Validation Server is to issue and validate authorization tokens to improve decision performance of the authorization service.

6.4 The Security Gateway library

The security gateway is the auxiliary library facilitating the usages of Authentication, Authorization and Token servicesservices.

- CSSI/GAAAPI: The client server application utilizes Security Gateway through the CSSI interface for invoking authentication service
- PEP is in charge of communicating with “Authorization Server” to get authorization decisions by using “SAM-XACML Request/Responder”.
- SAM-XACML Request/Responder: the component to handle XACML authorization requests from PEP to the SAML protocol [SAM- XACML2] before sending them to the “Authorization Server”.

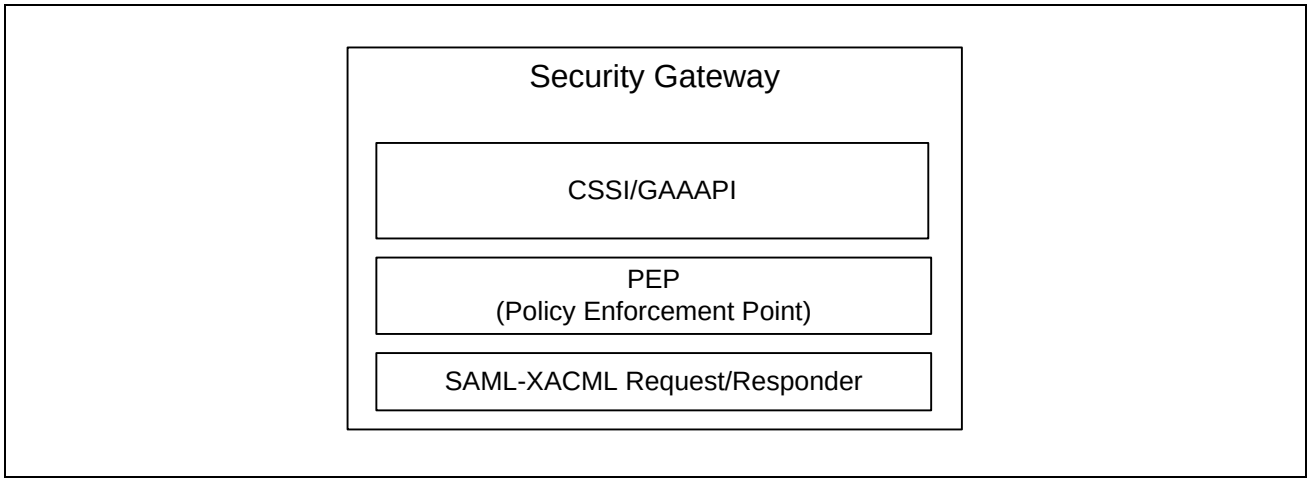


Figure 6-3: Security Gateway library for AAI

6.5 AAI Interfaces

This part describes external interfaces of AAI components used to interact with other applications.

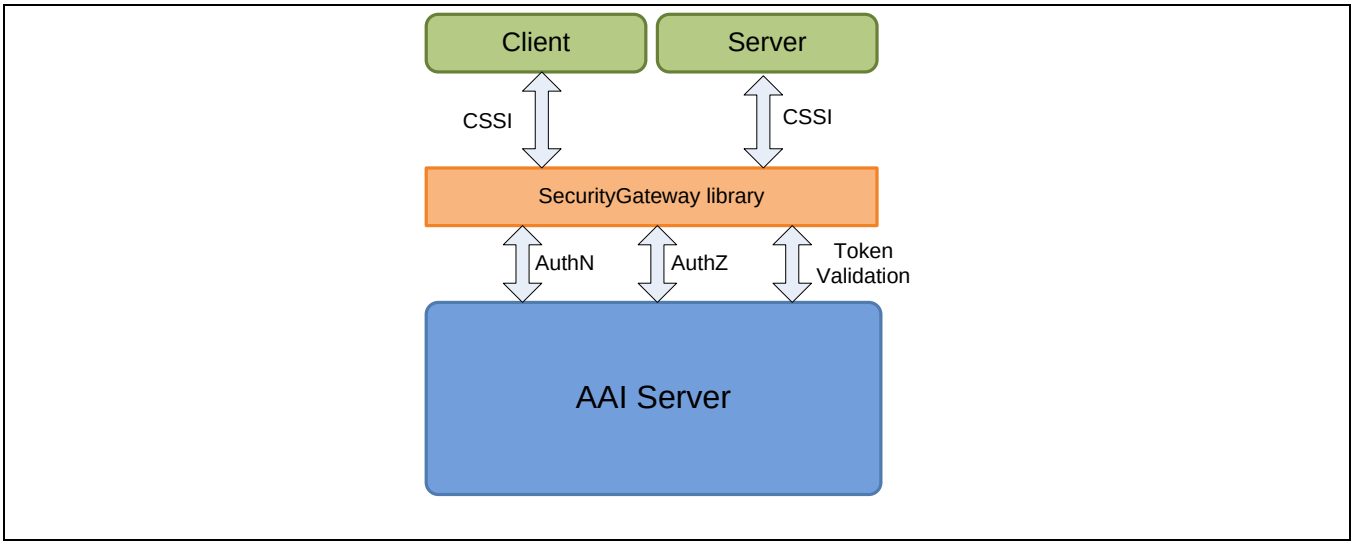


Figure 6-4: Interaction with AAI components through interfaces

Following table summarises interfaces and their messages

Table 6.1 – AAI interfaces and messages

Interface	Peer	Protocol/API	Direction	Functionalities
AuthN	SecurityGateway ←→ AAI Server	SAML protocol over SOAP	In/out	Authenticate a subject based on submitted credential and return authn-token.
AuthZ	SecurityGateway ←→ AAI Server	SAML-XACML protocol over SOAP	In/out	Provide decisions for authorization requests.
Token Validation	SecurityGateway ←→ AAI Server	SOAP	In/out	Validate authentication tokens and authorization tokens.
CSSI	Client/ Server ←→ SecurityGateway	Authenticate(authn-credentials)	In/out	Authenticate with a Identity Management Server and return authn-token.
		AuthorizeAction(request)	In/out	Authorize a request with the AuthZ/AAI server.

7 Common Security Services Interface (CSSI)

7.1 General CSSI functional structure

WS-Security standard, as native to SOA and ESB [7, 8], provides necessary security mechanisms and interface for virtualised resources interconnection, but their practical use in multi-domain/inter-domain virtualised environment will complicated with necessary trust relations and namespaces configuration at each communicated entity. To simplify this problem for the dynamically provisioned virtualised security services, at the level security related interfaces configuration and information management, the CSSI has been proposed. Technically CSSI combines the core functionality of the GSS-API

[42] for authentication service, GAAA-NRP authorization and session/token management [25]. The CSSI can be used together with WS-Security but introduces a simplified CSSI request format and SOAP security header structure that used a common SecurityContext container with the following structure:

```
SecurityContext (AuthenticationData, AuthorizationData, SessionData, SecurityData)
```

This will allow more flexibility in defining actual security data format and semantic that will exchanged between the virtualised services and the provider services, which due to their dynamicity will have high variation of the structure and semantics. CSSI and DACI will be configured together with provisioned VI at the deployment stage.

GEYSERS security services can be called from all other services to implement/add basic security services such as (1) data protection, (2) access control (authentication, authorization, delegation or identity mapping), and (3) policy enforcement. It should be noted that the security services discussed here are related to securing GEYSERS services and applications interaction and may be positioned as application layer security services according to X.800/ISO7498 Open Systems Security Architecture [18, 19]. GEYSERS security infrastructure may use other layer security services and mechanisms to protect communication channels such as VPN/IPSec, HTTPS, but these services can be implemented using existing standard libraries and are not the scope of the GEYSERS security design.

GEYSERS CSSI implements the following interface components:

- Standard Generic Security Services Application Programming Interface (GSS-API) [42] that supports data and/or message encryption/decryption, signature, authentication and delegation.
- Generic Authentication and Authorization API (GAAAPI) that supports basic authentication and extended authorization functionality for complex multi-domain resource provisioning [GAAA-NRP, IETF-RFC2904] that requires inter-domain provisioning and authorization sessions management and supports the whole provisioned services lifecycle. The basic GAAAPI functionality is implemented in the GAAA Toolkit (GAAA-TK) pluggable Java Library that will be extended with additional functions for combined network and IT resources provisioning.
- Simple Policy Based Management interface that supports policy based processes and objects management. These types of functions are called out from the Control and Management System that executes an object or runs a process during its execution.

The messages to request CSSI functions are described in the following table:

Table 7.1: CSSI functions and messages

Functionality	Message	Direction	Description
Data Encryption	Encrypt (data) Decrypt (cipher data)	Service or Application ←→ Encryptor	Enrypts and decrypts data in a form of binary data or XML document. Protects data confidentiality.
	Encrypt and Wrap (data)	Service or Application ←→ Encryptor	Encrypts data and enclose them in a standard container/envelop, e.g., XMLEncryption.
Data Signing	Sign (data) Validate (signed data, signature)	Application ←→ Signer	Signs data and validates signature where data can be in a form of binary data or XML document. Protects data integrity.

	Wrap and Sign (data) Validate (container with signed data)	Application ←→ Signer	Wraps data into standard container, signs and attach signature. Protects data integrity.
Authentication and Delegation	Authenticate (ID, credentials)	Application ←→ AuthN Service	Request to retrieve monitoring information about the status of a physical resource. Issues AuthN token that confirms positive authentication.
	Delegate (AuthenticatedEntityID, AuthN assertion, newID)	Application ←→ Identity Manager	Allows delegation or mapping of the authenticated entity. Allows mapping between entities and roles in different domains.
Authorization	AuthorizeAction (subject, resource, action)	Application ←→ AuthZ Service	Performs authorization of the request to do action or the resource for the subject. May issue AuthZ ticket issued as an authorization credential/assertion.
	AuthorizeActionSession (subject, resource, action, SessionID)	Application ←→ AuthZ Service	Performs authorization of the request to do action or the resource for the subject and binds AuthZ context to the SessionID. May return AuthZ ticket issued as an authorization credential/assertion. Pilot token is issued as a session credential.
	AuthoriseActionObligated (subject, resource, action)	Application ←→ AuthZ Service	Allows conditional AuthZ decision. Additionally may return a set of conditions or Obligations that shall be enforced either by the resource or next domain in case of multiple or multidomain resources access.
Policy Based Management or Tasks Execution	AuthoriseObject (object, resource, policy)	Object/Process ←→ Ctrl&Mngnt Service	Allows policy based process/object management or tasks execution.
	AuthoriseObjectObligated (object, resource, policy)	Object/Process ←→ Ctrl&Mngnt Service	Extends policy based process/object management with obligated/conditional decision.

Detail descriptions of Authentication and Delegation interface, Authorization interface are provided in subsequent sections.

7.2 Authentication and Delegation Interface

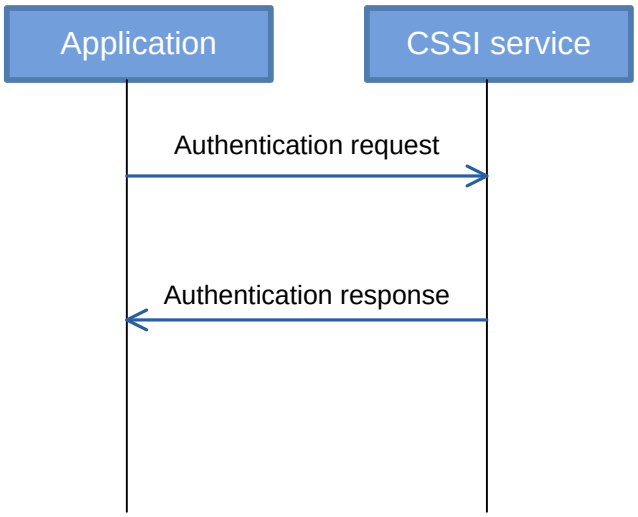


Figure 7-1 – Authentication sequence diagram

Msg no.	Abstract message	Message direction
1	Authentication Request	Application → CSSI service
2	Authentication Response	CSSI service →Application
3	Delegation Request	Application →CSSI service
4	Delegation Response	CSSI service → Application

Message – Authentication request			
Elements	**Multiplicity**	**Description**	**Element Type**
Message_Type	1	Type of the message	Integer
Credential_type	1	Type of credential to authenticate	Integer
Credential	1	Credential data, it could be UsernameCredential element for usename/password authentication or the existing authentication token in the AuthenticationTokenCredential element.	Credential

Element – UsernameCredential			
Sub-elements	**Multiplicity**	**Description**	**Element Type**
Username	1	Username of the subject to authenticate	String
Password_Type	1	Type of password. It could be the default PasswordText(0) or PasswordDigest(1) which using nonce value in digest.	Integer
Password_Value	1	Password information which related to the password_type, e.g: hash of the password	String

Nonce	0..1	The cryptographic random nonce using for password. The encoding type is Base64	String

Element – AuthenticationTokenCredential			
Sub-elements	**Multiplicity**	**Description**	**Element Type**
Base64Encoding	1	The base64 encoding of XML credential	String

Message – Authentication response			
Elements	**Multiplicity**	**Description**	**Element Type**
Authentication_status	1	Authentication status: AUTHENTICATED (0) or UNAUTHENTICATED (1)	Integer
Authentication_Token	0..1	If the Authentication_Status is AUTHENTICATED, this field contains authentication token in Base64 encoding.	String

7.3 Authorization Interface

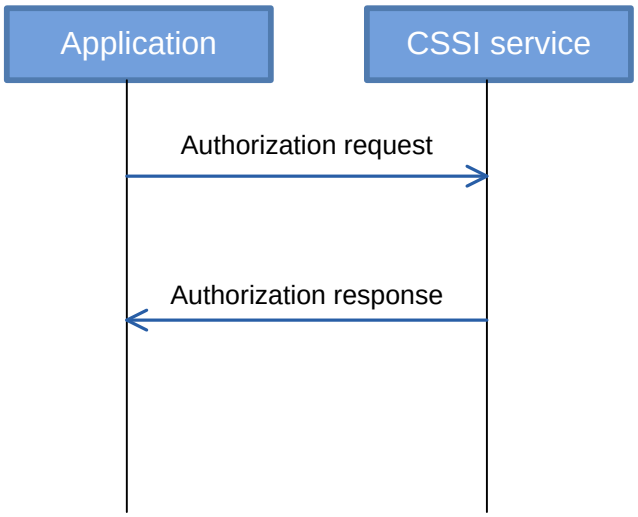


Figure 7-2 – Authorization sequence diagram

Msg no.	Abstract message	Message direction
1	Authorization Request	Application → CSSI service
2	Authorization Response	CSSI service →Application

Message – Authorization request			
Elements	**Multiplicity**	**Description**	**Element Type**
Message_Type	1	Type of the message	Integer
SessionID	1	The authorization session Id to specify which authorization service to request	String

Subject	1	The subject to authorize	AttributeList
Resource	1	The resource to authorize	AttributeList
Action	1	The action perform on the resource	AttributeList
Environment	1	The environment information for authorization	AttributeList

Element – AttributeList			
Sub-elements	**Multiplicity**	**Description**	**Element Type**
NumberOfAttribute	1	The number of attribute in the list	Integer
Attribute	1..n	The attribute in the list	Attribute

Element – Attribute			
Sub-elements	**Multiplicity**	**Description**	**Element Type**
ID	1	The identifier of the attribute value	String
Value	1	The value of the attribute	String

Message – Authorization response			
Elements	**Multiplicity**	**Description**	**Element Type**
Status	1	Contains one of following value: AUTHORIZED (0) or UNAUTHORIZED (1)	Integer
Token	0..1	Contains returned authentication token if the result is AUTHORIZED	String

7.4 Authentication and Authorization for NIPS client-server

Workflows to support authentication and authorization for NIPS server could be in Pull model (Figure 7-3) or in Push model (Figure 7-4).

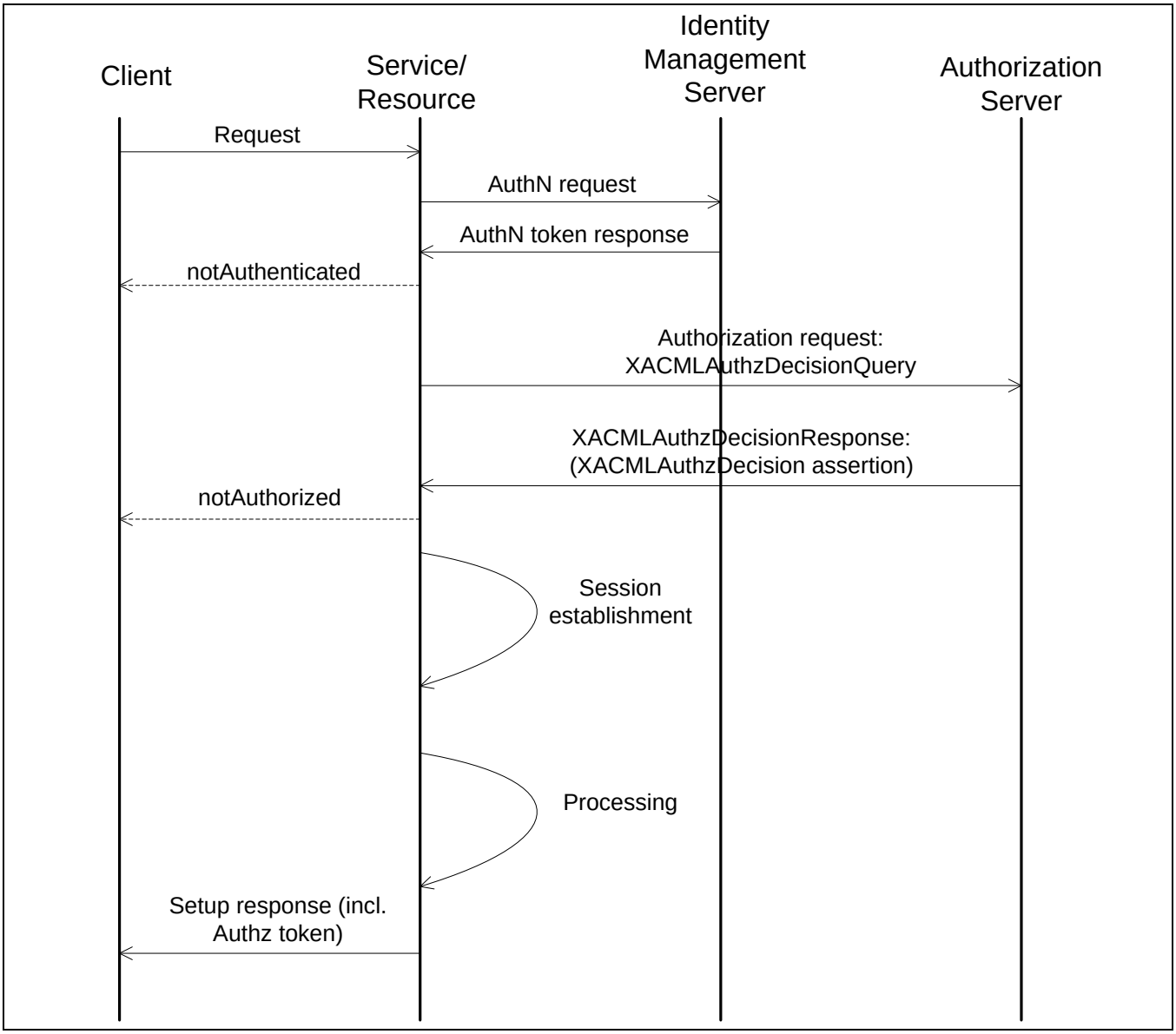


Figure 7-3: Sequence diagram of Authentication and Authorization in Pull model

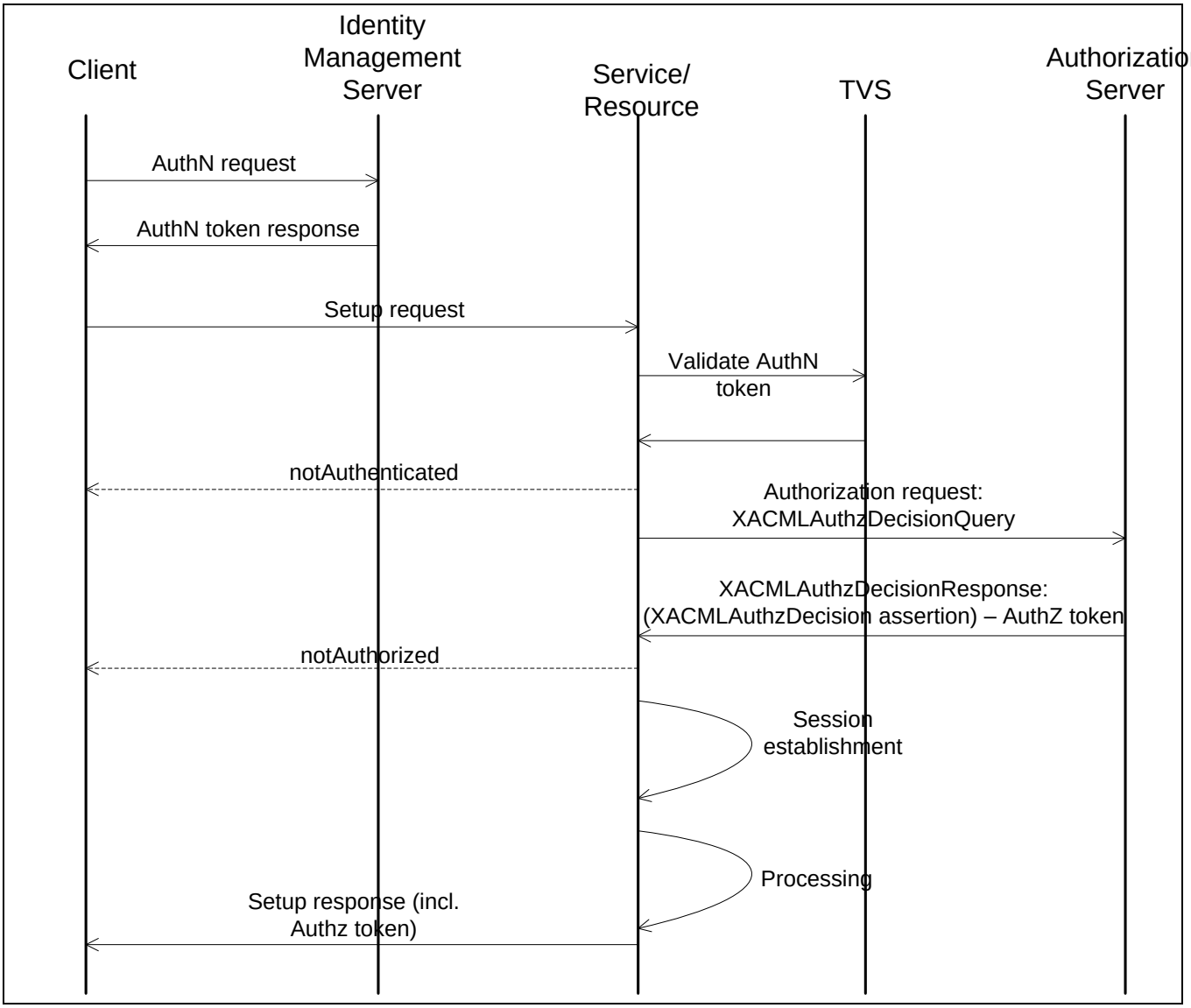


Figure 7-4: Sequence diagram of Authentication and Authorization for NIPS-UNI in Push model

After receiving the NIPS response including the AuthZ token, in subsequent messages, the NIPS client could utilize this token to get advantage of performance (Figure 7-5).

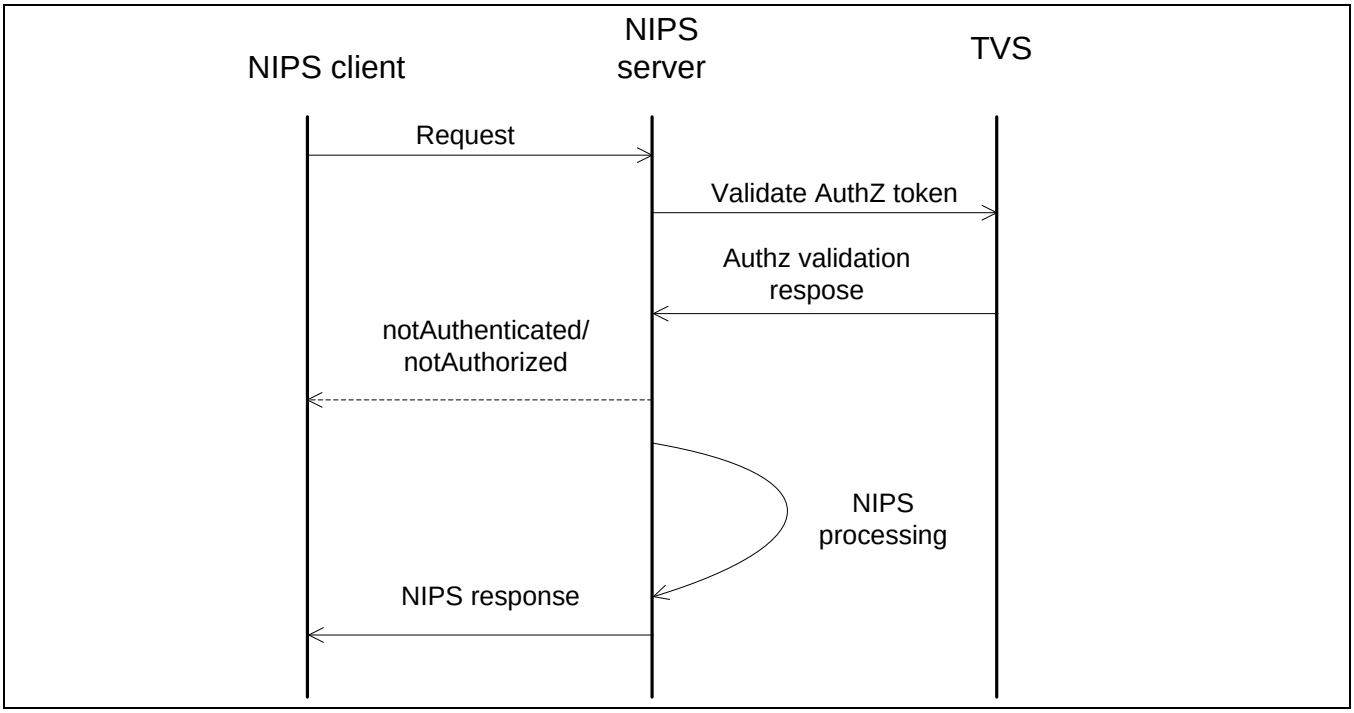


Figure 7-5: Sequence diagram for NIPS client-server using AuthZ token

7.5 AAI Request and Response Formats

The AAI Authorization and Authentication protocol will use SAML protocol as a basic and in particular SAML2-XACML2 protocol that incapsulate XACML Request/Response messages. However it will be extended with the possibility to carry on authorization tokens. Details will be worked out at the design and implementation stage.

8 GEYSERS Access Control Use Cases

According to use-cases at different layers SML, NCP+ and LICL, the AAI needs to fulfil following access control use-cases:

- Access Control Use Cases at NCP+ : which is the interaction between the VIO-IT at SML and the VIO-N at NCP+
- Access Control Use Cases at Upper-LICL: which are the interactions between the VIO (at SML and NCP+) and the VIP at upper-LICL.
- Access Control Use Cases at Lower-LICL (PIP): These use-cases illustrate interactions between the VIP and the PIP.

8.1.1 Access Control Use Cases at NCP+ (VIO-N)

The SML, under the context of the VIO-IT, could interact with NCP+ (the VIO-N) through the NIPS-UNI interface. Access control use-cases at this interface are as follow:

Table 8.1: Permissions for NIPS-UNI interface

Permission	Role	Resource	Action	Description
1	VIO-IT	IT-Advertisement	NIPS:Validate-IT-Advertisement	Validation of IT advertisements provided by the VIO-IT
2	VIO-IT	Network-Service	NIPS:Setup	VIO-IT is allowed to request the setup of a new network service between two end-points (they could be declared in terms of IT capabilities)
3	VIO-IT	Network-Service	NIPS:Modify	VIO-IT is allowed to request the modification of an existing network service.
4	VIO-IT	Network-Service	NIPS:Request-Quotation	VIO-IT is allowed to request the quotations for network connectivity services between different end-points.
5	VIO-IT	Network-Service	NIPS:Delete	VIO-IT is allowed to delete the network service between to end-points
6	VIO-IT	Monitoring-Info	NIPS:Request-Monitoring-Info	VIO-IT is allowed to request/receive monitoring info

8.1.2 Access Control Use Cases at Upper-LICL (VIP)

The Upper-LICL, which is under context of the VIP, has external interfaces for other layers of other roles to communicate with. They are SLI, CCI and NLI.

8.1.2.1 *SLI interface*

This interface is used by SML (VIO-IT) to invoke upper-LICL layer (VIP). Access control use-cases at this interface are as summarized in the following table:

Table 8.2 : Permissions for SLI interface

Permission	Role	Resource	Action	Description
1	VIO	VR-RP	SLI:Instantiate-VR-IT	A VIO is allowed to request VR instantiation:
2	VIO	VR	SLI:Decommission-VR-IT	A VIO is allowed to decommission a VR:
3	VIO	VR-RP-Info	SLI:Get-Available-VR-Pool-IT	A VIO is allowed to get available resource for a resource pool
4	VIO	VR	SLI:Operate-VR	A VIO is allowed to operate/control on a particular VR instance
5	VIO	VR-State-Info	SLI:Monitor-VR-Info	A VIO is allowed to request the state of a device/node

6	VIO	VR-Power-Info	SLI:Monitor-VR-Info	A VIO is allowed to get device power consumption of a device/node
7	VIO	VR-Status-Info	SLI:Monitor-VR-Info	A VIO is allowed to get status of a device/node
8	VIO	VR-Info	SLI:Subscribe-VR-Monitoring	A VIO is allowed to subscribe monitoring information of a device/node from VIP
9	VIO	VR-Info	SLI:Unsubscribe-VR-Monitoring	A VIO is allowed to remove monitoring subscription of a device/node from VIP
10	VIO	VR	SLI:Add-VirtualNetworkIf	A VIO is allowed to add new virtual network interface.
11	VIO	VR	SLI:Remove-VirtualNetworkIf	A VIO is allowed to remove virtual network interface
12	VIO	VR	SLI:Create-StorageImage	A VIO is allowed to create a new storage image
13	VIO	VR	SLI:Remove-StorageImage	A VIO is allowed to remove a storage image from a node/VR
14	VIP	VR-RP-Info	SLI:Advertise-VR-Pool	A VIP is allowed to advertise available resource pool
15	VIP	VR-RP-Instantiation-Status	SLI:Notify-VR-Info	VIP is allowed to notify instantiation request status to VIO
16	VIP	VR-RP-Decommission-Status	SLI:Notify-VR-Info	VIP is allowed to notify a decommission request status to VIO
17	VIP	VR-Operation-Status	SLI:Notify-VR-Info	VIP is allowed to notify a operation request status to VIO
18	VIP	VR-Info	SLI:Notify-VR-Info	VIP is allowed to notify a subscription update to VIO

8.1.2.2 *CCI interface*

These are access control use-cases between the NCP+ and the upper-LICL:

Table 8.3: Permissions for CCI interface

Permission	Role	Resource	Action	Description
1	VIO-N	VNode-Info	CCI:Synch-Request	VIO-N is allowed to request synchronize information of virtual node at VIP (LICL)
2	VIO-N	VNode	CCI:Configure	VIO-N is allowed to configure a cross-connection in the virtual node at VIP (LICL)

3	VIO-N	VNode-Monitor-Info	CCI:Monitor	VIO-N is allowed to get monitoring information from the virtual node at VIP (LICL)
4	VIP	VNode-Info	CCI:Synch-Update	VIP (LICL) is allowed to update information about node and its interfaces to VIO-N (NCP+)
5	VIP	VNode-Operation-Info	CCI:Notify	VIP is allowed to notify about cross-connection operation progress to VIO-N (NCP+)
6	VIP	VNode-Status-Info	CCI:Notify	VIP is allowed to notify about virtual node status to VIO-N (NCP+)

8.1.2.3 *MLI interface*

These are access control use-cases between the SML and NCP+ to the upper-LICL:

Table 8.4: Permissions for MLI interface

Permission	Role	Resource	Action	Description
1	VIO	VI	MLI:Request-VI	A VIO is allowed to request a VI
2	VIO	VI-Request	MLI:Query-VI-Request-Status	A VIO is allowed to query VI request status
3	VIO	VI-Request	MLI:Get-SLA-Offer	A VIO is allowed to get SLA offer of sent VI request
4	VIO	VI-Request	MLI:Sign-SLA-Offer	A VIO is allowed to sign SLA Offer of sent VI request
5	VIO	VI	MLI:Instantiate-VI	A VIO is allowed to request the instantiation of its VI
6	VIO	VI	MLI:Decomission-VI	A VIO is allowed to request the decommissioning of its VI.
7	VIO	VR-IT	MLI:ReplanningVI:Add-VR-IT	Replanning: Add IT node: The VIO asks to include a new device on the VI
8	VIO	VR-IT	MLI:ReplanningVI:Modify-VR-IT	Replanning: Modify IT node: The VIO requests to modify some of the characteristics of an IT node (+/- storage, +/- computing power)
9	VIO	VR-IT	MLI:ReplanningVI:Delete-VR-IT	Replanning: Delete node: The VIO requests to delete a device from the VI.
10	VIO	VLink	MLI:ReplanningVI:Add-VLink	Replanning: Add a network link: The VIO requests to add a new link between two devices on the VI
11	VIO	VLink	MLI:ReplanningVI:Modify-VLink	Replanning: Modify link: The VIO requests to modify the capacity of a link
12	VIO	VLink	MLI:ReplanningVI:Delete-VLink	Replanning: Delete link: The VIO requests to delete a link from the VI.

13	VIO	VI	MLI:ReplanningVI:Modify-Time	Replanning: Modify VI: The VIO requests to modify the timeline of a VI (+/- time reserved).

8.1.3 Access Control Use Cases at Lower-LICL (PIP)

Lower-LICL at PIP provides two interfaces for VIP running Upper-LICL: ROS interface and VR Management interface. It also has the PR Management Interface which the PR-Admin can use to manage physical resources at PIP.

8.1.3.1 *ROS Interface*

Table 8.5: Permissions for ROS interface

Permission	Role	Resource	Action	Description
1	PIP	VR-Mon-Info	ROS:Notify-VR-Info	PIP is allowed to send a VR monitoring (status change) notification information to VIP through ROS interface at Upper-LICL
2	PIP	VR-Operation-Info	ROS:Notify-VR-Operation	PIP is allowed to send a VR operation execution status notification information to VIP through ROS interface at Upper-LICL
3	PIP	RP-Operation-Info	ROS:Notify-RP-Operation	PIP is allowed to send a Resource Pool operation execution status notification information to VIP through ROS interface at Upper-LICL
4	PIP	VR-Sync-Info	ROS:Notify-VR-Info	PIP is allowed to send a VR information update (configuration change) to VIP through ROS interface at Upper-LICL
5	VIP	VR-RP	ROS:Instantiate-VR-IT	A VIP is allowed to request the instantiation of its VR-ITs: from VR resource pool to VR
6	VIP	VR	ROS:Decommission-VR-IT	A VIP is allowed to request the decommissioning of its VR-IT (from VR to VR IT resource pool)
7	VIP	VR	ROS:Configure-VR	A VIP is allowed to send configuration commands to its VRs.
8	VIP	VR-RP	ROS:Get-Available-VR-Pool-IT	A VIP is allowed to get available IT resources for the VR IT resource pool
9	VIP	VR	ROS:Monitor-VR-Info	A VIP is allowed to request/receive monitoring information from its VRs.

8.1.3.2 *VR Management interface*

Table 8.6: Permissions for VR Management interface

Permission	Role	Resource	Action	Description
1	VIP	Resource-Kinds-Info	Request-Resource-Kinds	A VIP is allowed to request the Resource Kinds information of a PIP + the PIP inter-domain connections' information.
2	VIP	LR	Request-VR	A VIP is allowed to request a set of VR to the PIP
3	VIP	LR	Instantiate-VR	A VIP is allowed to request the instantiation of its VR: from LR to VR (with VR network), or from LR to VR resource pool (with VR IT)
4	VIP	VR	Decommission-VR	A VIP is allowed to request the decommissioning of its VR. With VR-IT, from VR-resource pool to LR

8.1.3.3 *PR Management Interface*

Table 8.7: Permissions for PR Management interface

Permission	Role	Resource	Action	Description
1	PIP-Admin	PR	Add-PR	PIP's Admin is allowed to add new PR to the PR Management at Lower-LICL
2	PIP-Admin	PR	Delete-PR	PIP's Admin is allowed to remove an existing PR to the PR Management at Lower-LICL
3	PIP-Admin	Link	Add-Link	PIP's Admin is allowed to add new link to the PR Management at Lower-LICL
4	PIP-Admin	Link	Delete-Link	PIP's Admin is allowed to remove an existing link to the PR Management at Lower-LICL
5	PIP-Admin	Domain	Add-Domain	PIP's Admin is allowed to add new domain to the PR Management at Lower-LICL
6	PIP-Admin	Domain	Delete-Domain	PIP's Admin is allowed to remove an existing domain to the PR Management at Lower-LICL
7	PIP-Admin	SLATemplate	Add-SLA-Template	PIP's Admin is allowed to add an SLA template Management at Lower-LICL

8.2 XACML Attribute Profile for GEYSERS

8.2.1 Resource profile

Attribute name	Attribute ID	Full XACML attributeId semantics (e.g: ns-prefix = http://geysers.eu/)	Notes
Resource Identifier	resource-id	{ns-prefix}/{domain}/ resource/resource-id	Unique identifier of a resource. This is the value of VI-GRI, VR-LRI or PR-LRI.
Resource Type	resource-type	{ns-prefix}/{domain}/resource/resource-type	Specify type of resource.
VI Identifier	VI-id	{ns-prefix}/{domain}/resource/vi-id	This attribute specifies the identifier of the VI in which the resource belongs to.
Domain	resource-domain	{ns-prefix}/{domain}/resource/resource-domain	Specify security domain in which the resource belongs to.

8.2.2 Subject profile

Subject related attributes allow building policy depending on the properties of the request Subject or user. Subject related attributes are considered as a part of the XACML Subject definition.

Attribute name	Attribute ID	Full XACML attributeId semantics (e.g: ns-prefix = http://geysers.eu/)	Notes
Subject Identifier	subject-id	{ns-prefix/subject/subject-id	Indicate the identifier entity of a specific role.
Subject Role	subject-role	{ns-prefix}/subject/subject-role	E.g: VIO, VIO-N, VIP, PIP

Subject Confirmation Data	subject-confdata	{ns-prefix}/subject/subject-confdata	This attribute specifies the material using to confirm subject. It could be an authentication token (e.g: SAML assertion, Keberos ticket)

8.2.3 Action profile

Attribute name	Attribute ID	Full XACML attributeId semantics (e.g: ns-prefix = http://geysers.eu/)	Notes
Action ID	action-id	{ns-prefix}/action/action-id	Could use standard XACML attribute: urn:oasis:names:tc:xacml:1.0:action:action-id

9 AAI Implementation with GAAA Toolkit

The AAI implementation are implemented as Java OSGi service bundles that can be deployed in Karaf/Servicemix enviroments, which includes authnsvc, authzsvc, tokensvc and the securitygateway bundles.

9.1 Authentication bundle

9.1.1 Service configuration

The authentication service uses a configuration file to store its parameters, including its public-private keypair, passphrase, the list of trusted certificates and the session life-time for the authentication token.

The global configuration file contains following parameters:

Parameter	Default value	Description
BaseDir	authnsvc/etc/upper-licl	Path to authnsvc configuration directory
KeyStore	upperlicl-authnsvc.jks	Keystore (.jks) of the authnsvc authority, using for signing SAML assertions
KeyStorePassword	cloudsecurity	Password to access keystore
KeyAlias	upperlicl-authnsvc	Key alias of the private key used for signing SAML token
KeyPassword	authnsvc-cloud	Password to access private key

CredentialFileName	Credentials	Credentials (usernames, hashed passwords) of users.
CertificateTrustList	ctl.properties	File containing certificate trust list.
MaxSessionTimeOut	30	The maximal session timeout for the SAML token, in minutes

9.1.2 Certificate and public-private keypair generation

The authentication bundle needs a public/private keypair for SAML assertion issuing and verification. This keypair and its equivalent X.509 certificate are stored in a .jks file specified by the "KeyStore" parameter. The script to generate this keypair and its X.509 certificate needs the 'keytool' inside JRE Java Runtime Environment package.

```
#!/bin/bash
KEYSTORE="upperlicl-authnsvc.jks"
STORETYPE="JKS"
STOREPASS="cloudsecurity"
KEYPASS="authnsvc-cloud"
ALIAS="upperlicl-authnsvc"
VALIDITY=180
KEYSIZE=2048

keytool -genkey -alias $ALIAS -dname "CN=UpperLICL-AuthnSvc, OU=SNE Group, O=UvA, C=NL" -validity
$VALIDITY -keypass $KEYPASS "RSA" -keysize $KEYSIZE -keystore $KEYSTORE -storepass $STOREPASS -
storetype $STORETYPE

keytool -exportcert -file "$ALIAS.crt" -keystore $KEYSTORE -storepass $SOTREPASS -alias $ALIAS -rfc
```

The content of the X.509 certificate file (.crt) needs to be directly copied to the certificate trust list file specified within the global configuration.

9.1.3 User Management

The credential file is used to store users' authentication passwords. Each line in the file contains a user's credentials with following format:

```
$username:$base64_password_hashed:$base64_salt:$user_attribute_file
```

The password hash is computed from the hash operations of the plaintext password and a random generated string called 'salt'.

```
hash_password = SHA1(SHA1(salt | password))
```

Note that an external administration tool 'authnsvc-admin' is provided in order to generate credentials from usernames and passwords.

9.2 Authorization bundle

9.2.1 Service configuration

The authzsvc configuration folder contains the policyconfig.xml file and the policies folder as follows, first for the Upper-LICL and secondly for the lower-LICL.

```
\policyconfig.xml
\policies\
       permission-cci-operations.xml
       permission-mli-replanning-vlink-operations.xml
       permission-mli-replanning-vr-it-operations.xml
```

```
        permission-mli-vi-operations.xml
        permission-mli-vi-request-operations.xml
        permission-ros-notifications.xml
        permission-sli-operations.xml
        PPS-PIP-Role.xml
        PPS-VIO-N-Role.xml
        PPS-VIO-Role.xml
        RPS-PIP-Role.xml
        RPS-VIO-N-Role.xml
        RPS-VIO-Role.xml
```

```
\policyconfig.xml
\policies\
        permission-prmi.xml
        permission-ros-operations.xml
        permission-vrmi.xml
        PPS-PIP-Admin-Role.xml
        PPS-VIP-Role.xml
        RPS-PIP-Admin-Role.xml
        RPS-VIP-Role.xml
```

List of policies are specified in the policyconfig.xml, including two types of policies: context and reference policies. The context policies are policies identified based on the attribute of request, in this case are roles policies. Other policies are identified by references.

9.2.2 Policy management

XACML policies are organized using RBAC profile as follows:

9.2.2.1 *Permission policies*

Permissions of an interface are defined in one or several xml files having file name syntax:

```
Permission-$interfaceName-$permissionGroup.xml
```

Permissions policies are:

- permission-cci-operations.xml
- permission-mli-replanning-vlink-operations.xml
- permission-mli-replanning-vr-it-operations.xml
- permission-mli-vi-operations.xml
- permission-mli-vi-request-operations.xml
- permission-prmi.xml
- permission-ros-notifications.xml
- permission-ros-operations.xml
- permission-sli-operations.xml
- permission-vrmi.xml

9.2.2.2 *Permissions assigned to role policies*

Permissions that are mapped to the role policies have the filename:

```
PPS-$RoleName-Role.xml
```

Each policy of a role contains references to permission policies assigned to this role. For example with permissions assigned to VIP role policy, it has permissions of VRMI interface, ROS-operations, SLI notifications and CCI notifications.

```
<?xml version="1.0"?>
<PolicySet PolicySetId="PPS:VIP:role"
PolicyCombiningAlgId="urn:oasis:names:tc:xacml:1.0:policy-combining-algorithm:permit-overrides">
    <Target/>
    <!-- Rules for VIP role at L-LICL -->
    <PolicySetIdReference>permission:vrmi</PolicySetIdReference>
    <PolicySetIdReference>permission:ros-operations</PolicySetIdReference>

    <!-- Notification policy for VIP at SML -->
    <PolicySetIdReference>permission:sli:notifications</PolicySetIdReference>

    <!-- Notification policy for VIP to NCP+ -->
    <PolicySetIdReference>permission:cci:notifications</PolicySetIdReference>
</PolicySet>
```

Figure 9-1: Sample permissions assigned to the VIP role policy

Policy filenames are:

- PPS-PIP-Admin-Role.xml
- PPS-PIP-Role.xml
- PPS-VIO-N-Role.xml
- PPS-VIO-Role.xml
- PPS-VIP-Role.xml

When the administrator needs to change determined permissions of a given role, he only needs to add or remove the necessary references in the above files.

9.2.2.3 ***Role policies***

These policies contain role attribute matching to link with permission assigned to role policies. Policy filenames are:

- RPS-PIP-Admin-Role.xml
- RPS-PIP-Role.xml
- RPS-VIO-N-Role.xml
- RPS-VIO-Role.xml
- RPS-VIP-Role.xml

10 Conclusion

The document described the proposed AAI for on-demand provisioned virtualised infrastructure services and provided general implementation suggestions that provide necessary information for the ongoing AAI design and implementation.

YD: add aboiut future development, your plans about federation, trust model and infrastructure modelling.

11 References

[1] Generalised Architecture for Dynamic Infrastructure Services (GEYSERS Project) - http://www.geysers.eu/

[2] GEYSERS Project Deliverable D2.1 - Initial GEYSERS Architecture & Interfaces Specification

[3] GEYSERS Project Deliverable D3.1 - Functional Description of the Logical Infrastructure Composition Layer (LICL)

[4] Generic Architecture for Cloud Infastrcuture as a Service (IaaS) provisioning model. SNE Technical Report, 2011.

[5] Demchenko, Y., J. van der Ham, M. Ghijsen, M. Cristea, V. Yakovenko, C. de Laat, "On-Demand Provisioning of Cloud and Grid based Infrastructure Services for Collaborative Projects and Groups", The 2011 International Conference on Collaboration Technologies and Systems (CTS 2011), May 23-27, 2011, Philadelphia, Pennsylvania, USA

[6] NIST Definition of Cloud Computing v15. [Online] http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc

[7] OASIS Reference Architecture Foundation for Service Oriented Architecture 1.0, Committee Draft 2, Oct. 14, 2009. http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-cd-02.pdf

[8] Chappell, D., "Enterprise service bus", O'Reilly, June 2004. 247 pp.

[9] OSGi Service Platform Release 4, Version 4.2. - http://www.osgi.org/Download/Release4V42

[10] TMF Service Delivery Framework. http://www.tmforum.org /servicedeliveryframework/4664/home.html

[11] TMF Software Enabled Services Management Solution. - http://www.tmforum.org/BestPracticesStandards/SoftwareEnabledServices/4664/Home.html

[12] Zhao, G., C. Rong, J. Li, F. Zhang, Y. Tang, "Trusted Data Sharing over Untrusted Cloud Storage Providers," IEEE International Conference on Cloud Computing Technology and Science, November 30-December 03, Indianapolis, Indiana. pp. 97-103. ISBN: 978-0-7695-4302-4.

[13] "Assessment of Access Control Systems", by Vincent C. Hu, David F.Ferraiolo, D. Rick Kuhn. Interagency Report 7316. [Online] Available: http://csrc.nist.gov/publications/nistir/7316/NISTIR-7316.pdf

[14] Samarati, P., S.C. de Vimercati, Access Control: Policies, Models, and Mechanisms, in book "Foundations of Security Analysis and Design", LNCS, Springer Berlin/Heidelberg, 2001, Pages 137-196

[15] Sandhu, R. & Samarati, P., 1994. "Access Control: Principles and Practice", IEEE Communication Magazine, September 1994, pp. 40-48.

[16] Sandhu, R., Coyne, E. J., Feinstein, H. L. & Youman, C.E. 1996, "Role-Based Access Control Models", IEEE Computer, February 1996, pp. 38-47.

[17] Information Technology - Role Based Access Control, Document Number: ANSI/INCITS 359-2004, InterNational Committee for Information Technology Standards, 3 February 2004, 56 p.

[18] ISO/IEC 10181-3:1996 Information technology -- Open Systems Interconnection -- Security frameworks for open systems: Access control framework. – Available in "OSG Authorization API". - http://www.opengroup.org/online-pubs?DOC=9690999199&FORM=PDF

[19] ITU-T Rec. X.812 (1995) | ISO/IEC 10181-3:1996, Information technology - Open systems interconnection - Security frameworks in open systems: Access control framework. [Online]. Available: http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.812-199511-I!!PDF-E&type=items

[20] RFC2903 Laat de, C., G. Gross, L. Gommans, J. Vollbrecht, D. Spence, "Generic AAA Architecture," Experimental RFC 2903, Internet Engineering Task Force, August 2000. ftp://ftp.isi.edu/in-notes/rfc2903.txt

[21] RFC 2904 - "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 - http://tools.ietf.org/html/rfc2904

[22] RFC2748: The COPS (Common Open Policy Service) Protocol, Edited Durham, D., January 2000. - http://www.ietf.org/rfc/rfc2748.txt

[23] RFC2753: A Framework for Policy-based Admission Control, January 2000. - http://www.ietf.org/rfc/rfc2753.txt

[24] RFC3621: Framework for Session Set-up with Media Authorization, April 2003. - http://www.ietf.org/rfc/rfc3521.txt

[25] "GAAA Toolkit pluggable components and XACML policy profile for ONRP", Phosphorus Project Deliverable D4.3.1. – September 30, 2008. [Online]. Available: http://www.ist-phosphorus.eu/files/ deliverables/Phosphorus-deliverable-D4.3.1.pdf

[26] Demchenko, Y., M. Cristea, C. de Laat, E. Haleplidis, Authorization Infrastructure for On-Demand Grid and Network Resource Provisioning, Proceedings Third International ICST Conference on Networks for Grid Applications (GridNets 2009), Athens, Greece, 8-9 September 2009. ISBN: 978-963-9799-63-9

[27] Security Guidance for Critical Areas of Focus in Cloud Computing V2.1. Cloud Security Alliance, December 2009. http://www.cloudsecurityalliance.org/csaguide.pdf

[28] Cloud Computing: Benefits, risks and recommendations for information security, Editors Daniele Catteddu, Giles Hogben, November 2009. http://www.enisa.europa.eu/ act/rm/files/deliverables/cloud-computing-risk-assessment

[29] Amazon AWS Security Center. Certification and Accreditation. - http://aws.amazon.com/security/#certifications

[30] Amazon IAM. http://aws.amazon.com/documentation/iam/

[31] Microsoft Azure Cloud Service. http://www.microsoft.com/windowsazure/AppFabric/Overview/default.aspx

[32] Demchenko, Y., C. Ngo, C. de Laat, "Access Control Infrastructure for On-Demand Provisioned Virtualised Infrastructure Services", International Symposium on Security in Collaboration Technologies and Systems (SECOTS2011), Part of CTS2011 Conference, 23-27 May 2011, Philadelphia, USA.

[33] RFC5280 - Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. May 2008. http://www.ietf.org/rfc//rfc5280

[34] Web Services Security: SOAP Message Security 1.1 (WS-Security 2004). OASIS Standard Specification, 1 February 2006. [Online] http://www.oasis-open.org/committees/download.php/ 16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

[35] Trusted Computing Group (TCG). [Online]. Available: https://www.trustedcomputinggroup.org/home

[36] NIST Special Publication 800-14 - Generally Accepted Principles and Practices for Securing Information Technology Systems. National Institute of Standards and Technology. September 1996. http://csrc.nist.gov/ publications/nistpubs/800-27/sp800-27.pdf

[37] Demchenko, Y., D.R. Lopez, J.A. Garcia Espin, C. de Laat, "Security Services Lifecycle Management in On-Demand Infrastructure Services Provisioning", International Workshop on Cloud Privacy, Security, Risk and Trust (CPSRT 2010), 2nd IEEE International Conference on Cloud Computing Technology and Science (CloudCom2010), 30 November - 3 December 2010, Indianapolis, USA.

[38] Demchenko, Y., C. de Laat, T. Denys, C. Toinard, Authorization Session Management in On-Demand Resource Provisioning in Collaborative Applications. COLSEC2009 Workshop, The 2009 International Symposium on Collaborative Technologies and Systems (CTS 2009), May 18-22, 2009, Baltimore, Maryland, USA. IEEE Catalog: CFP0916A-CDR. ISBN: 978-1-4244-4586-8. Pp. 201-208.

[39] *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*, OASIS Standard, 15 March 2005. [Online]. Available: http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf

[40] SAML 2.0 Profile of XACML 2.0, Version 2.0. OASIS Standard, 1 February 2005. [Online]. Available: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-saml-profile-spec-os.pdf

[41] Shibboleth Attribute Authority Service. [Online]. Available from: http://shibboleth.internet2.edu/

[42] RFC2853 - Generic Security Service API Version 2 : Java Bindings. June 2000. http://www.ietf.org/rfc/rfc2853.txt

[43] *eXtensible Access Control Markup Language (XACML) Version 2.0*, OASIS Standard, 1 February 2005. [Online]. Available: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf

[44] Multiple resource profile of XACML 2.0, OASIS Standard, 1 February 2005, available from http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-mult-profile-spec-os.pdf

[45] *Core and hierarchical role based access control (RBAC) profile of XACML v2.0*, OASIS Standard, 1 February 2005. [Online]. Available: http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-rbac-profile1-spec-os.pdf

[46] The Liberty Alliance Project. [Online]. Available from: http://www.projectliberty.org/

[47] Liberty Alliance ID-WSF 1.1 Specifications. [Online]. Available from: http://www.projectliberty.org/resource_center/specifications/liberty_alliance_id_wsf_1_1_specifications

[48] Security Assertion Markup Language (SAML) 2.0 Technical Overview, Working Draft 21, 21 February 2007. [Online]. Available from: http://www.oasis-open.org/committees/download.php/22553/sstc-saml-tech-overview-2%200-draft-13.pdf

[49] Pautasso, C., O.Zimmermann, F.Leymann, "RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision", 17th International World Wide Web Conference (WWW2008), Beijing, China.

[50] Fuse ESB - OSGi based ESB. - http://fusesource.com/products/enterprise-servicemix/#documentation

[51] Apache ServiceMix an Open Source ESB. - http://servicemix.apache.org/home.html

[52] Spring Security. Reference Documentation. http://static.springsource.org/spring-security/site/docs/3.1.x/reference/springsecurity-single.html

[53] GFD.80 "The Open Grid Services Architecture, Version 1.5". Open Grid Forum, September 5, 2006.

[54] Web Services Architecture. W3C Working Group Note 11 February 2004. [Online]. Available: http://www.w3.org/TR/ws-arch/

[55] Web Services Security: SOAP Message Security 1.1 (WS-Security 2004). OASIS Standard Specification, 1 February 2006. [Online] http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf

[56] Web Services Security: SAML Token Profile 1.1, OASIS Standard, 1 February 2006. [Online]. Available from: http://www.oasis-open.org/committees/download.php/16768/wss-v1.1-spec-os-SAMLTokenProfile.pdf

[57] Hierarchical resource profile of XACML 2.0, OASIS Standard, 1 February 2005, available from http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-hier-profile-spec-os.pdf

[58] Privacy policy profile of XACML v2.0. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-privacy_profile-spec-os.pdf

[59] XML Digital Signature profile of XACML v2.0. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-dsig-profile-spec-os.pdf

[60] eXtensible Access Control Markup Language (XACML) Version 3.0, CD-1, 16-Apr-09. http://www.oasis-open.org/committees/download.php/32425/XACML-3.0-cd-1-updated-2009-May-07.zip

[61] XACML v3.0 Administration and Delegation Profile Version 1.0, CD-1, 16-Apr-09. http://www.oasis-open.org/committees/download.php/32425/XACML-3.0-cd-1-updated-2009-May-07.zip

[62] XACML PDP Metadata Version 1.0, OASIS Working Draft, 24 February 2008. http://www.oasis-open.org/committees/download.php/27316/xacml-3.0-metadata-v1-wd-01.zip

[63] Bettini C., S. Jajodia, X. S. Wang, D. Wijesekera, “Provisions and Obligations in Policy Management and Security Applications”, Proceedings of the 28th VLDB Conference, Hong Kong, China, 2002.

[64] OpenSAML library. [Online] Available: https://spaces.internet2.edu/display/OpenSAML/Home/

[65] Generic AAA Toolkit pluggable Java library. [Online] Available: http://www.phosphorus.pl/software.php?id=gaaa_tk

[66] Sun's XACML Implementation. [Online] Available: http://sunxacml.sourceforge.net/

[67] Canh Ngo; Demchenko, Y.; de Laat, C., "Toward a Dynamic Trust Establishment approach for multi-provider Intercloud environment," Cloud Computing Technology and Science (CloudCom), 2012 IEEE 4th International Conference on , vol., no., pp.532,538, 3-6 Dec. 2012

Appendix A Using SAML and XACML to support generic Authorization scenario

The diagram below illustrates where SAML protocol and assertions and XACML Request/Response messages can be used in a typical policy based decision making [40].

The following sections will provide details about SAML and XACML languages and their use for access control in distributed service- oriented applications.

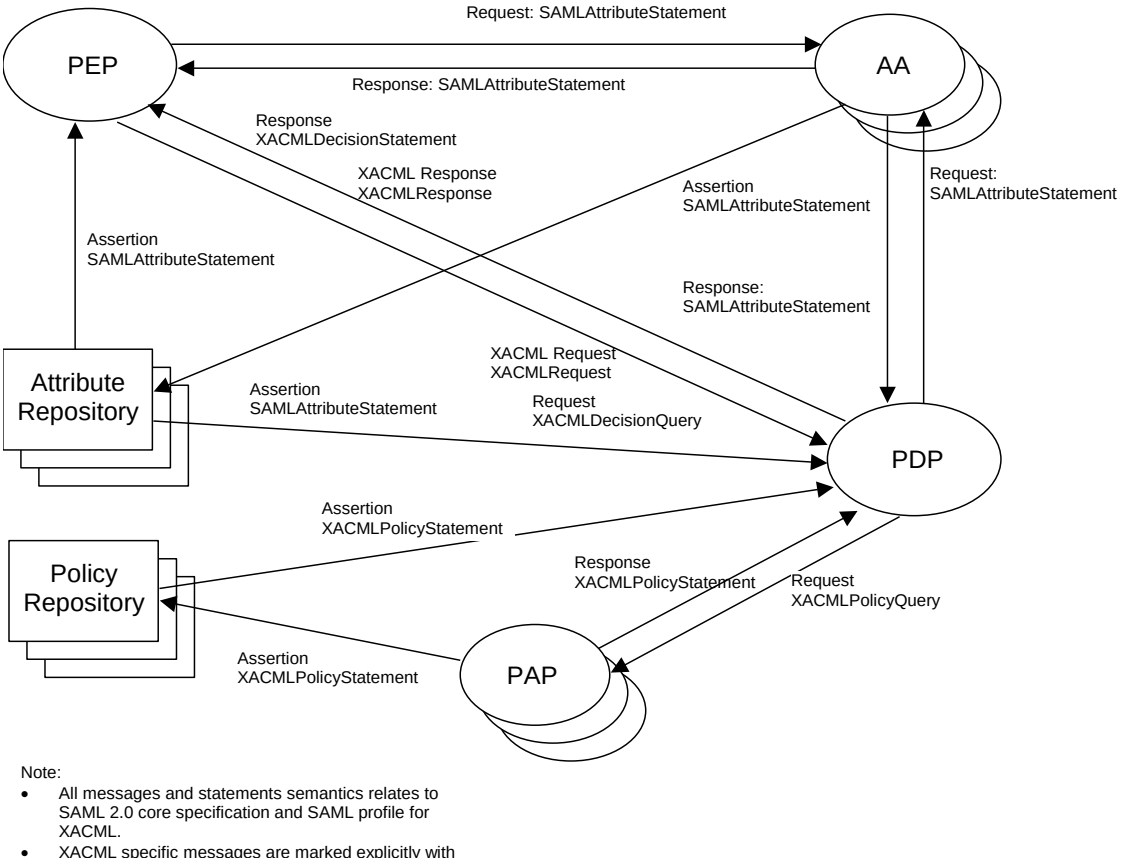


Figure A.1. Using SAML and XACML for messaging and assertions

A.1 SAML security assertions expression and exchange format

A.1.1 SAML Overview

Security Assertion Markup Language (SAML) is a an XML-based standard for expressing and communicating authentication, authorization and attribute information between distributed services.

The SAML operational security model suggests that all participating entities are members of the same security federation that have established business agreements, trust relations and share common attributes semantics [15]. More advanced

SAML and Web Services based protocols can support attributes and assertions exchange between different federations and security domains.

SAML Version 1.1 specification was published in 2003 and has been broadly used in identity management, web access applications and Web services security. Current SAML Version 2.0 specification was published in 2006 and adopted experience of the two major SAML implementation areas such as Shibboleth [41] and Liberty Alliance Identity Federation Framework [46, 47].

The major SAML application areas include:

Web Single Sign-On (WebSSO) allows a user who has authenticated to one web site to access other web sites that are the members of the same federation. SAML enables SSO providing a mean to communicate an authentication assertion from the original login site to other sites a user wants to access or where the user request is forwarded or redirected. The assertion then can be verified and validated and user authentication is confirmed.

Attribute-Based Authorization allows granting or denying user access to the protected resources based on user attributes that can be groups, roles or other specific to applications user characteristics. SAML provides a mechanism to communicate user attributes in addition to the user identity. User identity and attributes are managed and provide by the Identity Provider (IdP) and Attribute Authority Service (AAS) that operates as a part of federation. Separating IdP/AAS from Authentication and Authorization services simplifies typically distributed identity and access control infrastructure management.

Web Services Security (WS-Security) framework uses SAML as one kind of the security tokens within SOAP messages to convey security and identity information between actors in Web services interactions. The WS-Security SAML Token Profile is used by the Liberty Alliance's Identity Web Services Framework (ID-WSF) [18], Web Services Trust and Web Service Federation frameworks to support SSO, identity federation, identity mapping and other services.

A.1.2 SAML Basic Concepts and Components

SAML specification and architecture defines basic building components that allow a number of use cases and supports transfer of identity, attribute and authorization information between autonomous entities that have established trust relations. The core SAML specification defines the structure and content of both assertion and protocol messages used to transfer this information.

The means by which lower-level communication or messaging protocols (such as HTTP or SOAP) are used to transport SAML assertion or protocol messages is defined by the SAML bindings. SAML profiles define constrains and/or extensions to SAML assertions, protocol or binding to support the usage of SAML for a particular use case or application.

Two other concepts used for building and deploying interoperable SAML environment are metadata and authentication context.

Metadata defines a way to express and share configuration information between SAML parties and include the following data: site's supported SAML bindings, operational roles (IdP, Service Provider (SP), etc), identifier information, supporting identity attributes, federation names, and trusted keys information for encryption and signing.

Authentication context defines a way to provide information regarding the type and strength of authentication that a user employed when they authenticated at an identity provider. This information is provided as a part of an assertion's authentication statement. An SP can also include an authentication context in a request to an IdP to request that the user be authenticated using a specific set of authentication requirements, such as a multi-factor authentication.

Figure A.2 below illustrates relations between the basic SAML concepts and components and more details provided below [48].

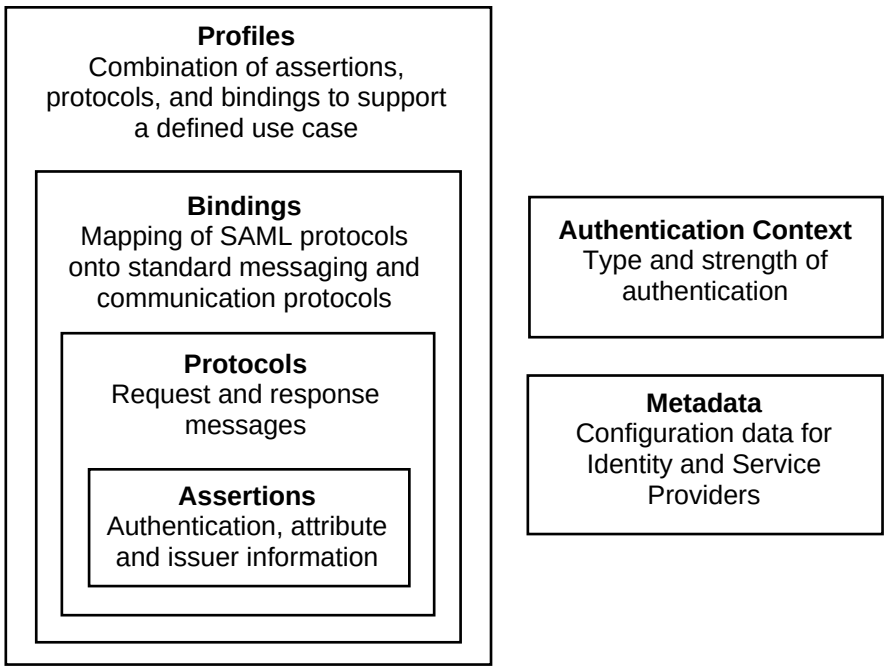


Figure A.2. SAML components [48].

A.1.3 SAML Assertions

SAML allows for one party to assert security information in the form of statements about a subject. An assertion contains some basic required and optional information that applies all assertions, and usually contains a subject of the assertion, conditions used to validate the assertion, and assertion statements. SAML defines three kinds of statements that can be carried within an assertion:

Authentication statements: These are created by the party that successfully authenticated a user. At a minimum, they describe the particular means used to authenticate the user and the specific time at which the authentication took place.

Attribute statements: These contain specific identifying attributes about the subject (for example, that user "John Doe" is a member of "Project A" with role "researcher").

Authorization decision statements: These are issued based on the authorization decision may state what the subject is entitled to do on the given resource (for example, "John Doe" is permitted to "create-reservation", "start-experiment-session" on the resource "Electronic Microscope XPS8076"). Authorization decision statement defined by the SAML2-XACML2 profile may contain full authorization context (see details below).

A.1.4 SAML Protocols

SAML defines a number of generalised request/response protocols:

Assertion Query and Request Protocol: This is the basic SAML protocol that defines a set of queries by which SAML authentication, authorization or attribute assertions may be obtained. The Query form of this protocol defines how a relying party can ask for assertions (new or existing) on the basis of a specific subject and the desired statement type.

Authentication Request Protocol: Defines a means by which a principal (or an agent acting on behalf of the principal) can request assertions containing authentication statements and, optionally, attribute statements. This protocol is used in

Web Browser SSO Profile when redirecting a user from an SP to an IdP in order authenticate user and optionally obtain user attributes.

Single Logout Protocol: Defines a mechanism to allow logout of active sessions associated with a principal. The logout can be directly initiated by the user, or initiated by an IdP or SP because of a session timeout, administrator command, etc.

Artifact Resolution Protocol: Provides a mechanism by which SAML protocol messages may be passed by reference using a small, fixed-length value called an artifact. The artifact receiver uses the Artifact Resolution Protocol to ask the message creator to dereference the artifact and return the actual protocol message.

Name Identifier Management and Name Identifier Mapping Protocols: Provide mechanisms to change or map the value or format of the name identifier used to refer to a principal. The issuer of the request can be either the service provider or the identity provider.

A.1.5 SAML Profiles

SAML profiles define how the SAML assertions, protocols, and bindings are combined and constrained to provide greater interoperability in particular usage scenarios. The profiles usually named by used protocol and a defined application area and include the following major profiles:

Web Browser SSO Profile: Defines how SAML entities use the Authentication Request Protocol and SAML Response messages and assertions to achieve single sign-on with standard web browsers. It defines how the messages are used in combination with the HTTP Redirect, HTTP POST, and HTTP Artifact bindings.

Assertion Query/Request Profile: Defines how SAML entities can use the SAML Query and Request Protocol to obtain SAML assertions over a synchronous binding, such as SOAP.

Enhanced Client and Proxy (ECP) Profile: Defines a specialized SSO profile where specialized clients or gateway proxies can use the Reverse-SOAP (PAOS) and SOAP bindings.

Single Logout Profile: Defines how the SAML Single Logout Protocol can be used with SOAP, HTTP Redirect, HTTP POST, and HTTP Artifact bindings.

Identity Provider Discovery Profile: Defines one possible mechanism for service providers to learn about the identity providers that a user has previously visited.

Other profiles are defined for Artifact Resolution Protocol, Name Identifier Management and Name Identifier Mapping Profile.

A.2 SAML Assertion datamodel and format

A.2.1 SAML top level elements

Figures below provide more detailed breakdown for SAML 2.0 Assertion format. The root element is called Assertion and mandatory contains the Issuer element and attributes Version, ID and IssueInstant. Depending on the profile the Assertion element may contain one or many statements such as defined in the standard AuthnStatement, AuthzDecisionStatement, AttributeStatement, or application defined statement that can be added through the abstract Statement element providing standard extension point. Other optional elements include Subject which is important in many profiles and use cases dealing with the identity information, Conditions and Advice. SAML Assertion may contain attached signature defined by the XML Signature standard.

In the compact XML DTD format the Assertion element can be descried as:

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

The Subject element consists of two basic components – subject ID that can be expressed in different formats and SubjectConfirmation that provides information how the subject identity was verified or authenticated. Both types of information can be encrypted.The Subject element 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
>
```

SAML Assertion provides the facility to describe conditions for assertion/credentials use and validity in the Conditions element that contains time validity constrains attributes, and elements that describe audience/community restriction, proxy/delegation restrictions and can also be extended to other application defined conditions.

The Advice element contains any additional information that the SAML authority wishes to provide. This information may be ignored by applications without affecting either the semantics or the validity of the assertion. Some potential uses of the Advice element include evidence supporting the assertion claims to be cited, either directly (through incorporating the claims) or indirectly (by reference to the supporting assertions), timing and distribution points for updates to the assertion, etc.

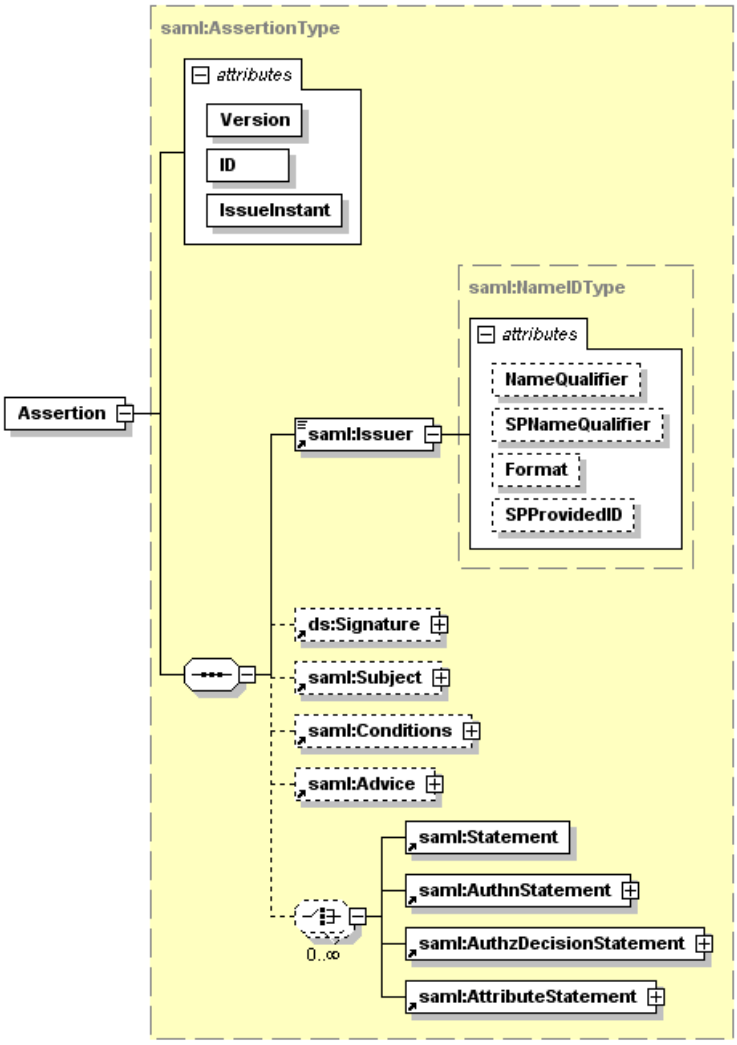


Figure A.3. SAML Assertion top elements

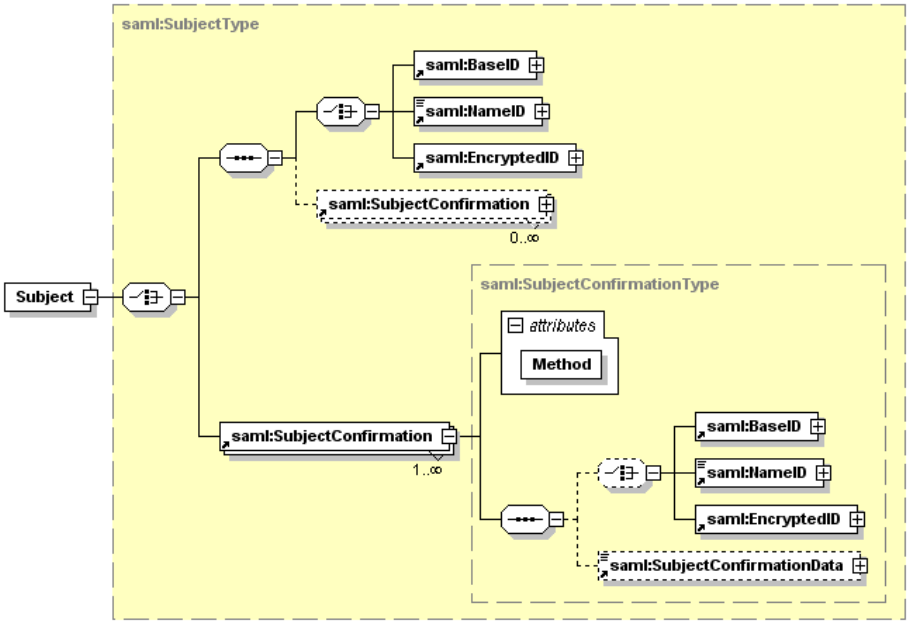


Figure A.4. SAML Subject elements

A.2.2 SAML AuthnStatement and AttributeStatement format

The SAML AuthnStatement is used to convey authentication statement issued by an Identity Provider or an authentication service and 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)>
```

The AuthnStatement has one mandatory attribute AuthnInstant that specifies the time at which the authentication took place, and two optional attributes SessionIndex that specifies the index of a particular session between the principal identified by the subject and the authenticating authority, and SessionNotOnOrAfter that specifies a time instant at which the session between the principal identified by the subject and the

The SubjectLocality specifies the DNS domain name and IP address for the system from which the assertion subject was apparently authenticated. SAML authority issuing this statement must be considered ended. The AuthnContext element specifies the context of an authentication event. The element can contain an authentication context class reference, an authentication context declaration or declaration reference, or both.

Listing below provides an example of the authentication Assertion containing AuthnStatement element.

```
<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:AAAuthority">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>
```

Figure A.5. Example SAML 2.0 Authentication Assertion

The SAML AttributeStatement provides a format for communicating Subject's attributes issued by the Attribute Authority or Identity Provider. Figure A.6 shows the structure of the SAML AttributeStatement element. It contains the following elements:

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

The AttributeStatement element describes a statement by the SAML authority asserting that the assertion subject is associated with the specified attributes. Assertions containing AttributeStatement elements must contain a Subject element. The AttributeStatement element may contain either attribute reference/value or encrypted attribute.

The Attribute element contains The Attribute element is used within an attribute statement to express particular attributes and values associated with an assertion subject, it identifies an attribute by name and optionally includes its value(s). The Attribute element has a obligatory attribute Name that holds the name of attribute, and optional attributes the NameFormat representing the classification of the attribute name in URI format, and the FriendlyName providing a more human-readable form of the attribute's name, which may be useful in cases in which the actual Name is complex or opaque, such as an OID or a UUID.

Listing below provides an example of the authentication Assertion containing AuthnStatement element.

```
<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:AAAuthority">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>
```

Figure A.6. Example SAML 2.0 Attribute Assertion

A.2.3 SAML2.0 profile of XACML: SAML-XACML protocol and Authorization assertions format

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 authorization 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.

SAML2.0 profile of XACML (SAML-XACML) combines well established SAML security assertions format [40] and reach functionality of the XACML policy format [43]. Such a solution provides a good combination between XACML policy expression and evaluation functionality and SAML security assertion management functionality. SAML-XACML profile is supported by the popular Open Source SAML implementation OpenSAML2.

The SAML2.0 profile of XACML defines the queries and assertions to support XACML based AuthZ services.

The XACMLAuthzDecisionQuery and XACMLPolicyQuery provide extension to the SAML protocol. The XACMLAuthzDecisionStatement and XACMLPolicyStatement provide extensions to the SAML assertions.

The XACMLAuthzDecisionQuery is introduced as additional query type for the SAML2.0 protocol. In contrary to the basic SAML2.0 queries, the XACMLAuthzDecisionQuery doesn't contain the Subject element but used as container for the xacml-context:Request message.

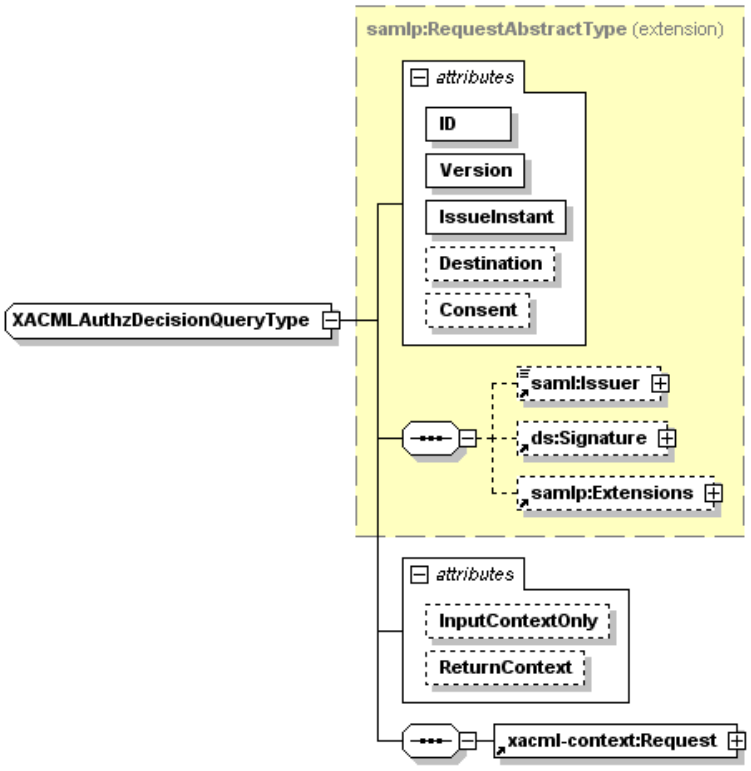


Figure A.7. XACML2.0 XACMLAuthzDecisionQuery format.

The XACMLAuthzDecisionStatement provides a container for XACML Request and Response messages that actually hold all necessary information about the authorization decision in a native XACML format. Figure below illustrates how the XACMLAuthzDecisionStatement is folded into the SAML assertion.

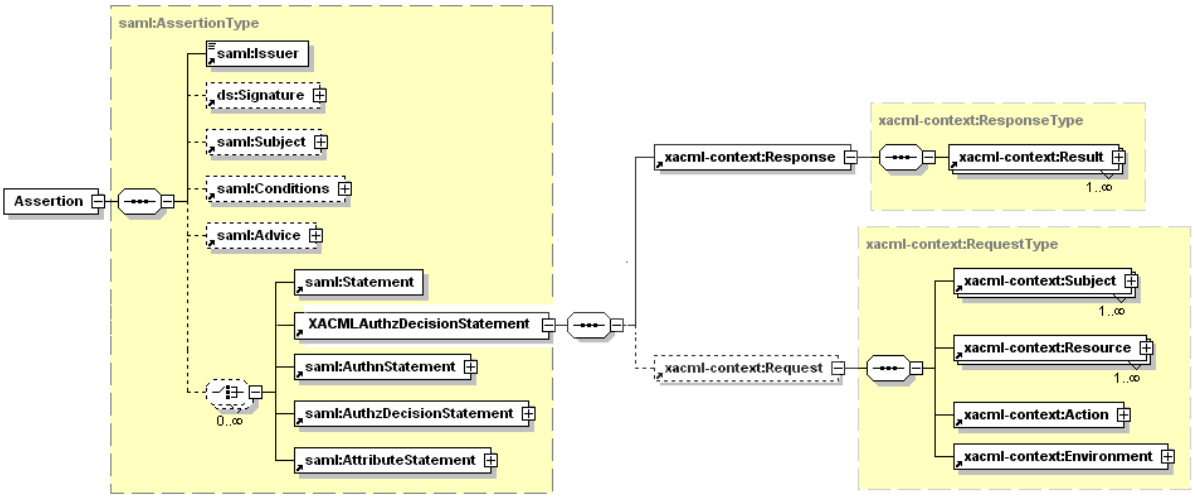


Figure A.8. XACML2.0 Assertion containing XACMLAuthzDecisionStatement.

A.3 XACML policy expression and messaging format

A.3.1 XACML Policy logical model

XACML (eXtensible Access Control Markup Language) is the OASIS standard that provides a well defined policy language with rich functionality to express complex rules for access control to different types of resources. XACML allows defining different application specific profiles. A number of special XACML profiles are discussed below.

The XACML policy logical model in a simple way can be presented as below. A XACML policy is defined for the target tuple "Subject-Resource-Action" (S-R-A) which can also be completed with the Environment (S-R-A-E) component to add additional context to instant policy evaluation. The Target element actually defines a matching expression between (S-R-A-E) of the request and the policy:

```
Target (S, R, A, E) =>
       => Target (M(Sreq,Spol), M(Rreq,Rpol), M(Areq,Apol), M(Ereq,Epol))
```

where M – is a matching function between attributes provided in the request and embedded in the policy. It is important to mention that XACML allows only 2 variables matching functions in the Target element which however can be cascaded [43].

XACML policy may contain a number of rules which in its own turn may contain a number of conditions and a rule Target used for rules matching (or selection). The Conditions can use a wide range of functions defined in the XACML specification [4]. The following describes the structure of the Rule element:

```
Rule(Target (S, R, A, E),
     Cond (F(Sreq, Spol), F(Rreq, Rpol), F(Areq, Apol), F(Ereq, Epol)),
     Obligation)
```

where F – is a logical function between attributes provided in the request and embedded in the policy.

Additional flexibility for XACML policy rules definition is provided by the possibility to use the full functionality of the XPath expressions that can refer to the ResourceContent element of the XACML Request message.

The XACML policy can also specify the policy Obligations as actions that must be taken on positive or negative authorization decisions. Introducing policy obligations allows for more flexible policy definition by separating stateless conditions that are based on the information provided in the access control request and stateful conditions that may depend on the target system/resource state. Obligations are included into the policy definition and returned by PDP to PEP which in its own turn should take actions as prescribed in the Obligation instructions or statements. As an example, policy obligations may prescribe that some actions must be logged or user account must be changed or mapped to another account when accessing the resource.

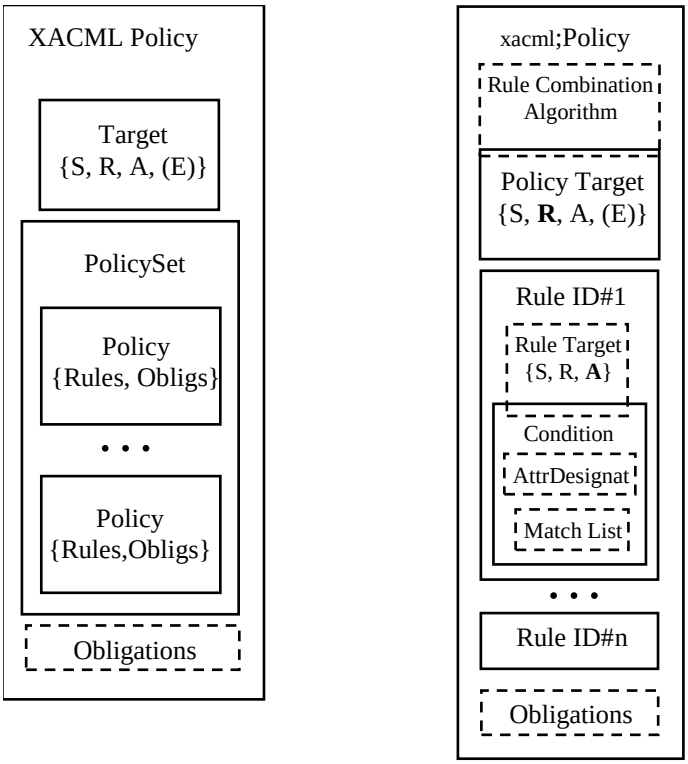


Figure A.8. XACML policy model.

A decision request sent in a Request message provides context for the policy-based decision. The 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 a single policy set that is applicable to the request. XACML specifies a number of policy and rule combination algorithms. The Response message may contain multiple Result elements, which are related to individual Resources.

Any of S-R-A-E elements allow for extensible “Attribute/AttributeValue” definition to support different attributes semantics and data types. Additionally, XACML allows for referencing internal and external XML documents elements by means of XPath functionality.

Two mechanisms can be used to bind the XACML policy to the resource: a Target element can contain any of S-R-A-E attributes and a policy identification attribute IDRef. XACML policy format provides few mechanisms to add and handle domain or session related context during the policy selection and request evaluation:

- Policy identification that is done based on the Target comprising of the Resource, Action, Subject, and optionally Environment elements.
- Attributes semantics and metadata can be namespace aware and used for attributes resolution during the request processing.

In complex authorization scenarios the security context e.g. from the previous authorization decision can be provided as an environment or resource attribute.

A.3.2 XACML 2.0 special profiles

XACML 2.0 RBAC profile [45]

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.

XACML Multiple Resources profile [44]

The conditions under which multiple <Resource> elements are allowed are described in the XACML Profile for Multiple Resources. 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.

XACML 2.0 Profile for Hierarchical Resources [49]

The hierarchical resource profile specifies how XACML can provide access control for a resource that is organized as a hierarchy, which examples are 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.

XACML 2.0 Privacy Policy Profile [50]

This profile provides standard attributes and a standard <Rule> element for enforcing the privacy protection principles, related to the purpose for which personally identifiable information is collected and used.

This specifies the following attributes:

"urn:oasis:names:tc:xacml:2.0:resource:purpose"

> This attribute indicates the purpose for which the data resource was collected. The owner of the resource SHOULD be informed and consent to the use of the resource for this purpose.

"urn:oasis:names:tc:xacml:2.0:action:purpose"

> This attribute indicates the purpose for which access to the data resource is requested.

XACML 2.0 XML Digital Signature Profile [51]

The profile provides a profile for use of the W3C XML-Signature Syntax and Processing Standard in providing authentication and integrity protection for XACML schema instances. The signature information must include a specification of the identity of the signer and a specification of the period during which the signed data object is to be considered valid.

A.3.3 XACML 3.0 Specifications and profiles currently under review

XACML 3.0 specification [52] is currently under review at the final stage as a candidate specification. The new specification provides better definition of the PEP-PDP interaction, adds the Advice element that in contrary to the Obligation element is not obligatory for enforcing by PEP, extends both the Obligation and Advice elements content with the ObligationExpression, AdviceExpression and AttributeExpression elements.

In the new specification the Policy and the Request and Response are defined by common schema with the "xacml" namespace.

The structure of the XACML 3.0 Request was simplified and all attributes are now placed under the single Attributes element, that contains two elements Attribute and Content, and can be distinguished by their AttributeId's. The Request reference multiple requests connected to the current one placed into the MultiRequests element.

The response is extended to contain elements AssociatedAdvice that hold returned by PDP policy advices, Status and PolicyIdentifierList.

The profiles of XACML 2.0 are updated and the following new profiles are proposed.

XACML 3.0 Administration and Delegation Profile [53]

The XACMLv3.0 Administrative and Delegation profile can indicate if the policy is issued by the trusted PolicyIssuer for the particular domain. In this case the PDP will rely on an already assigned or default PAP and established trust relations, otherwise when other entity is declared as a PolicyIssuer, the PDP should initiate checking administrative policy and delegation chain what is a suggested functionality of the PIP module.

XACML PDP Metadata (Working draft) [54]

XML PDP Metadata profile specifies an extensible schema for publishing information about PDP such as the version of XACML implemented, supported standard functions and combining algorithms, supported optional features, and the location of the PDP.

A.3.4 XACML2.0 policy datamodel

XACML provides a format for expressing policy for the generic Attribute Based Access Control (ABAC) model and defines a simple Request/Response messages format.

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.

XACML defines three top-level policy elements: `<Rule>`, `<Policy>` and `<PolicySet>` [4]. 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 **policie**s.

The `<Policy>` element (see Figure 13) 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 **PD**P, and so it is intended to form the basis of an **authorization decisio**n.

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 **rule**s or **policie**s, in particular:

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

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

Information security **policies** operate upon **attributes** of **subject**s, the **resource** and the **action** to be performed on the **resource** in order to arrive at an **authorization decisio**n. In the process of arriving at the **authorization decisio**n, **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.

Figure 14 shows the structure of 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.

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**, **resourc**e, **action** and **environment attributes** or functions of **attribute**s. 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.

XACML re-uses enumerated list of functions and operations defined in Xpath 2.0 [29] and XQuery 1.0 [30] used in the FunctionId attribute of the <Apply>/<Condition> element. Element Target contains matching specification for the attributes of the Subject, Resource and Action.

Example of the XACML policy is provided in Appendix.

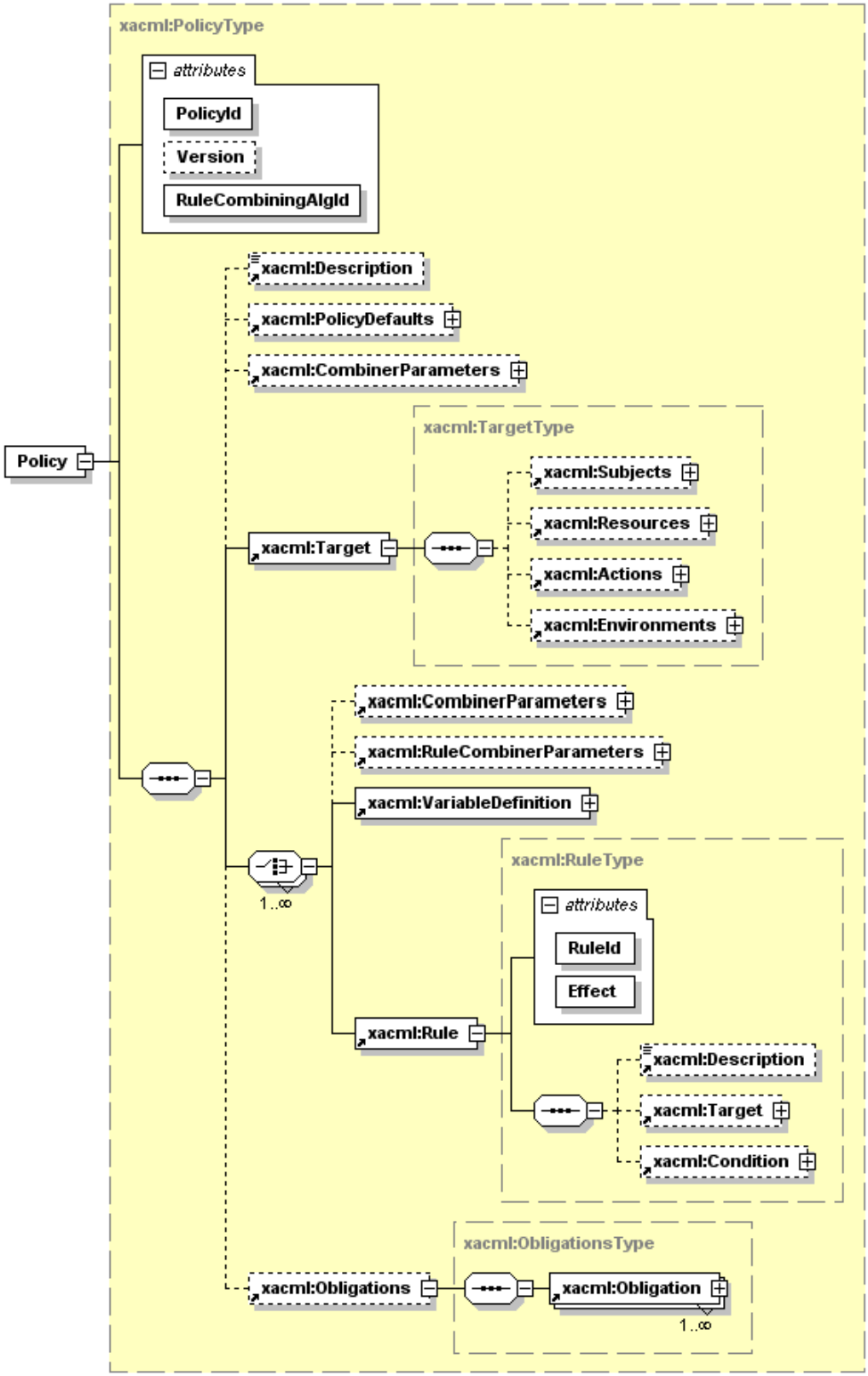


Figure A.9. XACML Policy data model.

A.3.5 XACML Request and Response

XACML 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 four mandatory elements Subject, Resource, Action, and Environment that may contain multiple attributes presented as AttributeId – AttributeValue pairs. The Resource attribute may also contain the ResourceContent element. The XACML2.0 speciation requires that all four elements are present but may be empty.

The Environment element provides a possibility to include a security context information such as a SessionId or an authorization from the previous domain.

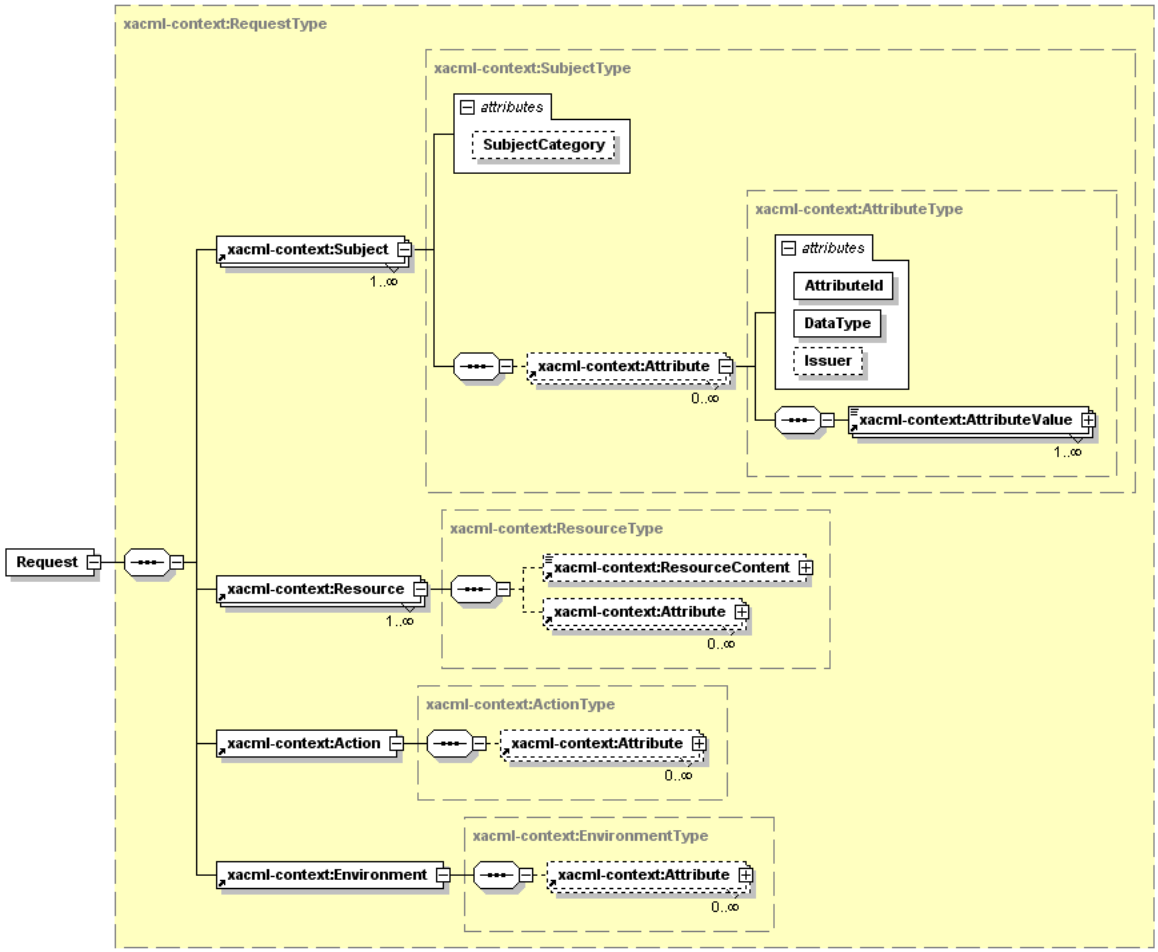


Figure A.10. High-level elements of the XACML 2.0 Request.

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.

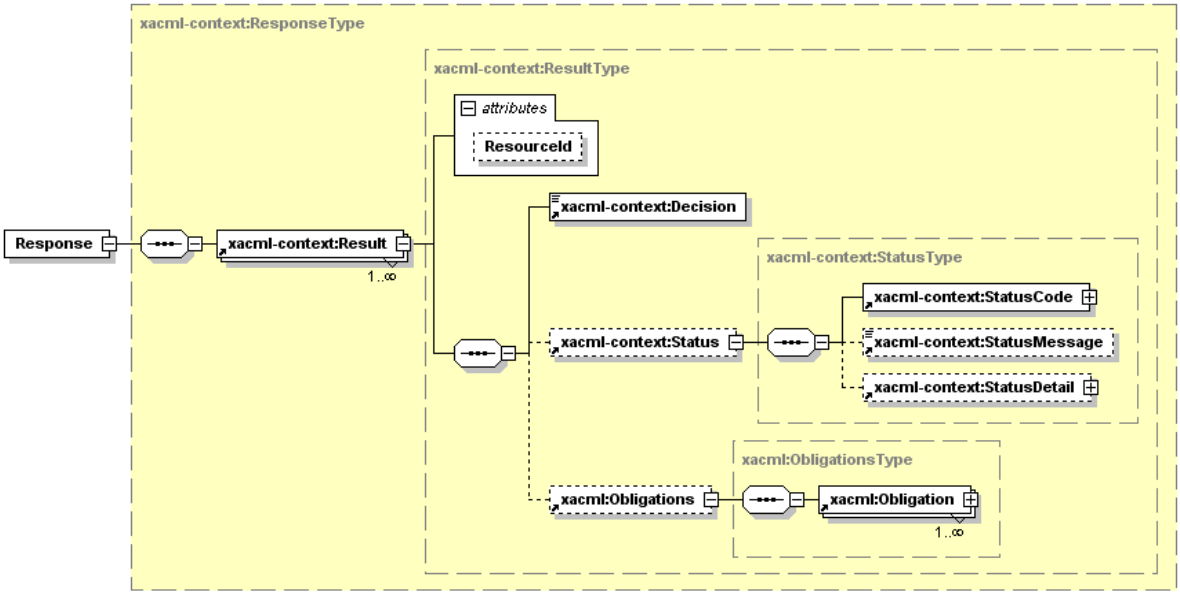


Figure A.11. High-level elements of the XACML 2.0 Response.

A request message sent by a user or an application is an XML message sent with SOAP massage over HTTP protocol. 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).

Appendix B Web Services Security Framework (WS-Security)

Web Services Architecture (WSA) [55] defines service according to Service Oriented Architecture (SOA) concept as a well-defined set of actions, it is self-contained, stateless, and does not depend on the state of other services. WSA includes core specifications SOAP (Simple Object Access Protocol) and WSDL (Web Services Description Language) Specifications from W3C [56] and UDDI (Universal Description, Discovery, and Integration) [34], which together provide service description, discovery and messaging framework for Web Services applications. "The description of a service in a SOA is essentially a description of the messages that are exchanged. This architecture adds the constraint of stateless connections, that is where the all the data for a given request must be in the request" [56]. Recently published WS-Resource Framework (WSRF) [57] standards extend WSA with state management functionality as required for such application arias as Grid Services, utility computing and business process management. WSRF actually provides functionality for managing stateful and transient services required in Grid applications and was accepted as a basic platform for the Open Grid Services Architecture (OGSA) [58].

Extended WSA includes such specifications as WS-Policy, WS-Coordination, WS-Transaction, WS-Inspection, WS-Addressing, and WS-Security framework [59]. Some other components are added to the WSA framework cooperatively with the Grid community, in particular, WS-Agreement as a set of Web services to provide a framework for negotiating agreements [60], WS-Notification and WS-Resource Framework that add the ability to model stateful resources using Web services [57].

WS-based specifications use SOAP header for communicating security context, i.e. initial security token or credential, what is considered to be a solution transparent for applications as SOAP header is processed automatically in most WS/SOAP applications. WS-Security describes enhancements to SOAP messaging to provide quality of protection through message integrity, message confidentiality, and single message authentication. These mechanisms can be used to accommodate a wide variety of security models and encryption technologies. WS-Security also provides a general-purpose mechanism for associating security tokens with messages and describes how to encode binary security tokens, in particular, X.509 certificates, Kerberos tickets, and encrypted keys. The WS-Security Profile for XML-based Tokens describes how to use XML-based tokens such as the SAML with the WS-Security specification. It also includes extensibility mechanisms that can be used to further describe the characteristics of the credentials that are included with a message.

Other specifications from the WS-Security stack include WS-Policy[61], WS-SecurityPolicy [62] that specifies format for the policy assertions, and WS-Trust (WST) [63] that enables Web Services to request and issue security tokens and to manage trust relationships. WS-SecureConversation (WSSC) [64] defines extensions for secure communication by establishing and sharing security contexts, and deriving session keys from security contexts. WS-Trust and WS-SecureConversation, as two complimentary specifications, provide a framework for (dynamic or session based) trust and credentials negotiation for Web Services. Additionally, WS-Federation (WSF) specification [65] proposes a framework for flexible Identity Management and leverages both WS-Trust and WS-SecureConversation specifications. WSF can add more flexible requestor identity management including pseudonymous services, identity and attributes mapping, single sign-on.

WST defines SOAP based mechanisms for brokering trust relationships, requesting and returning security tokens. Requests for security tokens are made by sending a Request Security Token (RST) to the Security Token Service (STS). WST specification defines three possible actions that can be performed: issue a new token, renew a token, or validate a token. It is essential that all these requests must provide initial secure credential or token as a base for issuing a new token.

WS-Federation defines mechanisms for federated identity management that are used to enable identity, attribute, authentication, and authorization federation across different trust realms. The federation model extends WS-Trust model to describe how identity providers act as security token services and how attributes and pseudonyms can be integrated in security token mechanisms to provide federated identity. Tokens can represent the principal's primary identity or some pseudonym. Services can request attribute/identity service based on provide token/pseudonym to obtain authorised information about the identity. WS-Federation Active Requestor and Passive Requestor Profiles define how the cross trust

realm identity, authentication and authorization federation mechanisms can be used by active requestors such as SOAP-enabled applications, or by passive requestors such as Web browsers to provide Identity Services. The functionality provided by WS-Federation is similar to identity federation provided by Liberty Alliance Project – widely used solution for federated Identity management [47].

However, it is important to stress that all these specifications don't deal with the initial trust establishing. Trust relations must be established in one or another way and presented in all WS-* interactions in a form of trust anchor or business anchor (which is in its own turn should be cryptographically proven).

So, even when considering to use well-defined solutions for session/instant security context establishing with WST (or other key management solutions like XKMS [67]) we still need to solve the problem of initial trust relations or establish an initial trust anchor. In currently used solutions and implementation for inter-domain access control the problem is split in two parts – federated trust for the attribute services/management (which is rather static) and confirmed/verifiable trust for the identity (which is dynamically established or invoked). This means that based on explicitly existing and presented trusted attribute credentials the identity credential confirmation/verification can be requested in a separated request to the identity origination site. This model is actually based on the separation of Authentication and Authorization.

Appendix C Conformance to WS-Interoperability Basic Profile and Basic Security Profile

WS-I Basic Security Profile conformance [63] - Extract

In order to conform to the BSP, any artefact that contains a construct that is addressed in the profile must conform to any statements that constrain its use. Conformant receivers are not required to accept all possible conformant messages. Conformance applies to deployed instances of services.

Since major portions of the BSP may or may not apply in certain circumstances, individual URIs may be used to indicate conformance to parts of the BSP including the core profile or additional sections of the BSP for Username token, X.509 token, and SOAP attachments.

The BSP includes statements that are interoperability requirements as well as statements that are security considerations. The normative requirement statements are identified by numbers prefixed with the letter 'R', for example Rnnnn where nnnn is the statement number. These statements contain one requirement level keyword (i.e., "MUST") and one conformance target. The following conformance targets are used in the BSP (Note: The list actually defines SOAP message elements that must be supported by the message processor):

- SECURE_ENVELOPE - a SOAP envelope that contains sub-elements that have been subject to integrity and/or confidentiality protection. A message is considered conformant when all of its contained artifacts are conformant with all statements targeted to those artifacts as appropriate in the Basic Security Profile. Use of artifacts for which there are no statements in the Basic Security Profile does not affect conformance.
- SECURE_MESSAGE: Protocol elements that have WS-Security applied to them. Protocol elements include a primary SOAP envelope and optionally associated SOAP attachments.
- SENDER: Software that generates a message according to the protocol(s) associated with it. A sender is considered conformant when all of the messages it produces are conformant and its behaviour is conformant with all statements related to SENDER in the BSP.
- RECEIVER: Software that consumes a message according to the protocol(s) associated with it. A receiver is considered conformant when it is capable of consuming conformant messages containing the artefacts that it supports and its behaviour is conformant with all statements related to RECEIVER in the BSP.

- INSTANCE: Software that implements a wsdl:port or a uddi:bindingTemplate.
- SECURITY_HEADER: An element included as a child of soap:Envelope/soap:Header and named wsse:Security.
- SOAP_HEADER - an element named soap:Header, included as a child of the SOAP_ENVELOPE.
- TIMESTAMP - an element named wsu:Timestamp, included as a child of a SECURITY_HEADER.
- MIME_BODY - the body of a multipart entity, as defined by MIME Basic Security Profile - Version 1.0 (BdAD) http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html
- MIME_PART: The MIME-defined header fields and contents of one of the parts in the body of a multipart entity in a SECURE_MESSAGE.
- HEADER_ELEMENT - an element included as a child of the SOAP_HEADER.
- MIME_HEADER - a header field of a multipart entity, as defined by MIME.
- MIME_PART - the MIME_BODY and all MIME_HEADERs associated with a single multipart entity, as defined by MIME.
- NONCE - an element named wsse:Nonce, included as a child of a USERNAME_TOKEN.
- PASSWORD - an element named wsse:Password, included as a child of a USERNAME_TOKEN.
- SIGNATURE: An element included as a child of a SECURITY_HEADER and named ds:Signature.
- REFERENCE: A SIGNATURE ds:Reference element.
- ENCRYPTED_DATA: An element named xenc:EncryptedData which is referenced by either an ENCRYPTED_KEY_REFERENCE_LIST or an ENCRYPTION_REFERENCE_LIST.
- ENCRYPTED_KEY: An element included as a child of a SECURITY_HEADER and named xenc:EncryptedKey.28 Rec. ITU-T Y.2232 (01/2008)
- ENCRYPTION_REFERENCE_LIST: An element which is included as a child of a SECURITY_HEADER and named xenc:ReferenceList.
- ENCRYPTED_KEY_REFERENCE_LIST: An element which is included as a child of an ENCRYPTED_KEY and named xenc:ReferenceList.
- SECURITY_TOKEN: Either an INTERNAL_SECURITY_TOKEN or an EXTERNAL_SECURITY_TOKEN (e.g., Username token, X.509 certificate token, REL token, or SAML token).
- SECURITY_TOKEN_REFERENCE: An element included as a descendant of a SECURITY_HEADER or an ENCRYPTED_DATA and which is named wsse:SecurityTokenReference.
- INTERNAL_SECURITY_TOKEN: A security token defined in a security token profile and that is either a child of a SECURITY_HEADER or a child of a wsse:Embedded element in a SECURITY_TOKEN_REFERENCE.
- EXTERNAL_SECURITY_TOKEN: A security token defined in a security token profile that is external to a SECURE_ENVELOPE.
- EXTERNAL_TOKEN_REFERENCE - a SECURITY_TOKEN_REFERENCE that refers to an EXTERNAL_SECURITY_TOKEN.
- USERNAME_TOKEN - a SECURITY_TOKEN named wsse:UsernameToken.
- X509_TOKEN - a BINARY_SECURITY_TOKEN containing an X.509 certificate.
- BINARY_SECURITY_TOKEN - a SECURITY_TOKEN named wsse:BinarySecurityToken.
- SAML_TOKEN - a SECURITY_TOKEN named saml:Assertion which conforms to the SAML 1.1 (via the OASIS Web Services Security SAML Token Profile 1.0).

- INTERNAL_SAML_TOKEN - an INTERNAL_SECURITY_TOKEN that is a SAML_TOKEN.

EXTERNAL_SAML_TOKEN - an EXTERNAL_SECURITY_TOKEN that is a SAML_TOKEN.SAML_AUTHORITY_BINDING - an element named saml:AuthorityBinding, included as a child of an SECURITY_TOKEN_REFERENCE.

WS-I Basic Security Profile conformance [12] – Extract

The Basic Profile (BP) defines the following conformance targets (in addition to BSP):

- MESSAGE - protocol elements that transport the ENVELOPE (e.g., SOAP/HTTP messages)
- ENVELOPE - the serialization of the soap:Envelope element and its content
- DESCRIPTION - descriptions of types, messages, interfaces and their concrete protocol and data format bindings, and the network access points associated with Web services (e.g., WSDL descriptions) (from Basic Profile 1.0)
- CONSUMER - software that invokes an INSTANCE (from Basic Profile 1.0)
- REGDATA - registry elements that are involved in the registration and discovery of Web services (e.g. UDDI tModels) (from Basic Profile 1.0)

Appendix D GSS-API Summary