034115

PHOSPHORUS

Lambda User Controlled Infrastructure for European Research

Integrated Project

Strategic objective:
Research Networking Testbeds

Deliverable reference number: D.4.5

Updated GAAA Toolkit library for ONRP (final project release)

Due date of deliverable: 31-03-2009
Actual submission date: 03-04-2009
Document code: <Phosphorus-WP4-D.4.5>

Start date of project: October 1, 2006 Duration: 30 Months

Organisation name of lead contractor for this deliverable:
University of Amsterdam

Revision [draft, 0.7]

Abstract

This deliverable provides update to the pluggable GAAA Toolkit library (delivered in the D4.3.1 deliverable) that implements the generic AAA Authorisation framework for Optical Network Resource Provisioning (GAAA-NRP).

The report describes the major security mechanisms and functional components that comprise the GAAA-NRP profile such as authorisation tickets and tokens, Token Validation Service (TVS), reference model for policy obligations handling (OHRM), XACML policy profile for NRP, and discusses their implementation in GAAA-TK library.

The report describes in detail a set of APIs used to call the main GAAA-TK services: authorisation service requested via the Policy Enforcement Point (PEP), authorisation request evaluation by the Policy Decision Point (PDP), and the TVS API that provides rich functionalities for handling tokens used for access control and signalling. A separate section is devoted to the GAAA-TK library setup and configuration.

Additionally the document provides short guidelines for developers explaining how to extend the library functionalities or create a special profile for a specific application area.

The current GAAA-TK library implementation provides full functionalities needed to support basic testbed scenarios. It has been tested with the WP1 Harmony testbed and integrated into the WP2 G2MPLS middleware.

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
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)	

Document Revision History

<This page to be deleted before submission to the EC>

Version	Date	Description of change	Person
0.0	23-03-09	Content and a template	Yuri Demchenko
0.1	28-03-09	First draft released	Yuri Demchenko
0.2	28-04-09	First full draft released	Yuri Demchenko
0.3	04-07-2009	Added section of library development and profiling	Yuri Demchenko
0.4	05-07-2009	Internal review and update	Mihai Cristea Yuri Demchenko
0.6	13-05-2009	Project review by Giada Landi	Giada Landi Nicola Ciulli
0.7	13-05-2009	Final release	Yuri Demchenko

REVIEW	Main reviewer		Nicola Ciulli - NXW	
Summary of suggested changes				
Recommendation	1) Major revision[1]	☐	2) Minor revision[2]	☒
Re-submitted for review - if 1)	DD/MM/YY			
Final comments				
Approved[3]:	DD/MM/YY			

[1] Deliverable must be changed and reviewed again before submission to the EC can be considered
[2] Deliverable may be submitted to the EC after the author has made changes to take into account reviewers' comments as appropriate
[3] For submission to EC

Table of Contents

0 Executive Summary

The Authentication, Authorisation and Accounting (AAA) service is an important component of the infrastructure supporting on-demand Optical Network Resource Provisioning (ONRP) across multiple domains and different target consumer applications. A consistent AAA infrastructure requires the interactions among the related AAA components at all networking layers, including the network/forwarding elements, the control plane, the reservation and provisioning service, and the user/target application layer.

This deliverable provides update to the pluggable GAAA Toolkit library (delivered in the D4.3.1 deliverable) that implements the generic AAA Authorisation framework for Optical Network Resource Provisioning (GAAA-NRP).

The report describes the major security mechanisms and functional components that comprise the GAAA-NRP profile such as authorisation tickets and tokens, Token Validation Service (TVS), reference model for policy obligations handling (OHRM), XACML policy profile for NRP, and discusses their implementation in the GAAA-TK library.

The report describes in detail a set of APIs used to call the main GAAA-TK services: authorisation service requested via the Policy Enforcement Point (PEP), authorisation request evaluation by the Policy Decision Point (PDP), and the TVS API that provides rich functionalities for handling tokens used for access control and signalling. A separate section is devoted to the GAAA-TK library setup and configuration.

Additionally the document provides short guidelines for developers explaining how to extend the library functionality.

The current GAAA-TK library implementation provides full functionalities needed to support basic testbed scenarios. It has been tested with the WP1 Harmony testbed and integrated into the WP2 G2MPLS middleware.

1 Introduction

The Authentication, Authorisation and Accounting (AAA) service(s) is considered as an important component of the infrastructure supporting on-demand Optical Network Resource Provisioning (ONRP) across multiple domains and different target consumer applications. A consistent AAA infrastructure requires the interaction among the related AAA components at all networking layers including network/forwarding elements, control plane, reservation and provisioning service, and user/target applications layer.

This deliverable describes the result of the implementation of the AAA Authorisation infrastructure for multi-domain ONRP as the pluggable GAAA-TK Java library. The proposed library is designed in such a way that it can be used at all networking layers: Data plane, Control Plane, Service plane, and can also work with applications, and in particular support the major Phosphorus testbed use cases. The proposed GAAA-NRP architecture and GAAA-TK library also target to ensure future compatibility with the Grid and NREN access control solutions and infrastructures.

The report is organised as follows. Section 2 describes the general design of the AAA Authorisation infrastructure for ONRP (GAAA-NRP). It describes the main functionalities and proposed security mechanisms to support multi-domain ONRP, in particular: an AuthZ token for multi-domain access control and signalling, an AuthZ ticket for extended AuthZ session management, a Token Validation Service (TVS) to support token based policy enforcement during path building and access, and a policy Obligation Handling Reference Model (OHRM).

Section 4 describes a set of APIs used to call the main GAAA-TK functions. The authorisation service can be requested via the Policy Enforcement Point (PEP) or directly from the XACML Policy Decision Point (PDP). The TVS API provides rich functionalities to handle tokens used for access control and signalling that can be called via PEP or directly from TVS.

Section 4 provides information how the XACML-NRP policy and attributes profile is implemented in the GAAA-TK library.

A separate section 5 is devoted to the GAAA-TK library setup and configuration. Section 6 provides short guidelines for future library development and extension.

2 Authorisation Infrastructure for Multidomain Network Resource Provisioning (GAAA-NRP) – General Design

This chapter describes the major security mechanisms used in GAAA-NRP architecture to support multidomain network resource provisioning and implemented in the GAAA-TK library: AuthZ token for multi-domain access control and signalling; AuthZ ticket for extended AuthZ session management; Token Validation Service (TVS) to enable token based policy enforcement, and policy obligations that allow conditional policy decisions in multidomain network provisioning environment. The GAAA-NRP architecture is described in the project deliverable D4.2 and D4.3.1. The GAAA-NRP extends further the generic AAA Authorisation Framework [1, 2] and can be used for the general complex resource provisioning [3, 4].

2.1 GAAA-AuthZ access control mechanisms and functional components

The proposed GAAA-NRP access control mechanisms and components extend the generic model described in GAAA-AuthZ with the specific functionalities for on-demand ONRP, in particular:

- Global Reservation ID (GRI) used to provide the integrity of the whole network resource provisioning process or workflow and ensure consistent reservation and access control sessions management.
- Authorisation tokens used for signalling and access control during the reservation or path building stages.
- Authorisation tickets to support extended AuthZ session management in multidomain resource provisioning.
- Policy obligations to support conditional policy decision and allow resource related context accounting, in particular for local user pool account mapping, or usable/accountable resource access/usage.
- Special XACM-NRP policy and attributes profile that allows for consistent XACML policy expressing and multi-domain authorisation attributes compatibility. Policy obligations to support conditional policy decision

and allow resource related context accounting, in particular for local user pool account mapping, or usable/accountable resource access/usage.

Such simple mechanism as Global Reservation ID (GRI) plays important role in managing the whole provisioning process and all reservation and authorisation sessions. In particular GRI is a mandatory attribute of all tickets and tokens used in GAAA-NRP. It is important convention that GRI is generated at the beginning and maintained during the whole provisioning process or provisioned resource lifecycle.

Figure 2.1 illustrates the major GAAA-NRP/GAAA-AuthZ modules and how they interact when evaluating a service request.

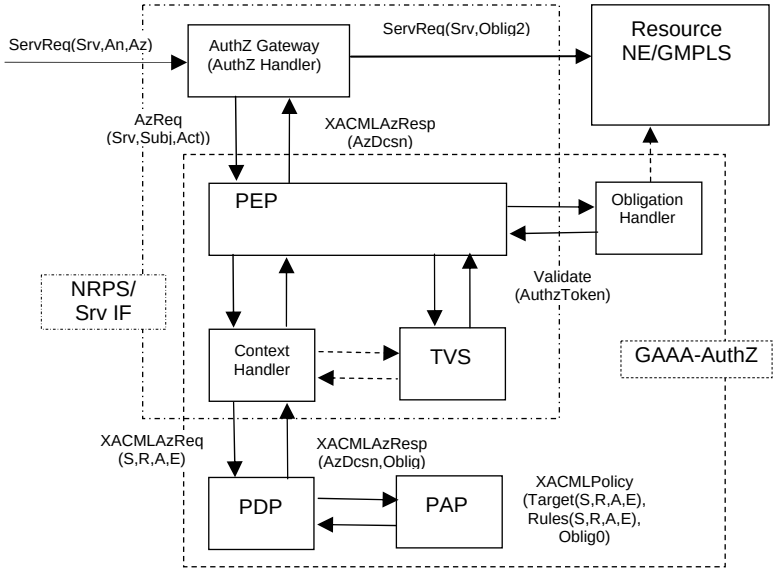

Figure 2.1. GAAA-AuthZ components providing service request evaluation

The authorisation service is called from the service/application interface via the AuthZ gateway (that can be just an interceptor process called from the service or application) that intercepts a service request ServiceRequest (ServiceId, AuthN, AuthZ) that contains a service name (and variables if necessary) and AuthN/AuthZ attributes. The AuthZ Gateway extracts necessary information and sends an AuthZ request AuthzRequest (ServiceId, Subject, Action), that contains a service name ServiceId, the requestor's identification and credentials, and the requested Action(s) to the PEP. The major PEP's task is to convert the AuthZ request's semantics into a PDP request, where semantics is actually defined by the used policy. When using an XACML policy and correspondingly an XACML PDP, the PEP will send an XACML AuthZ request to the PDP in the format (subject, resource, attributes, (environment)). If in general case the XACML policy contains obligations, they are returned in the XACMLAzResponse (AuthzDecision, Obligations). The PEP calls the Obligation Handler to process obligations which are defined as actions to be taken on the policy decision or in conjunctions with the service access (like account mapping, quota enforcing, logging, or accounting).

If the service request contains an AuthZ token that may reference a local or global reservation ID, or just identifies an AuthZ session in which context the request is sent, the token validation is performed by the Token Validation Service (TVS). The TVS is typically called from the PEP and returns a confirmation if the token is

valid. The defintion of the TVS as a separate function or service allows creating flexible token and/or ticket policy enforcement infrastructures for on-demand network resource provisioning.

2.2 Session management in GAAA-NRP

The session management functionality in GAAA-NRP and GAAA-TK is based on the general NRP model proposed in D4.2 that includes such stages as reservation, deployment, access/use, and decommissioning.

The overall network provisioning process initiates the provisioning session inside of which we can also distinguish two other types of sessions: reservation session and access session. Although they may require different security context, all of them are based on the (positive) AuthZ decision, may have a similar AuthZ context and will require a similar functionality when considering distributed multi-domain scenarios.

Figure 2.2 illustrates the relationship between all sessions which are bound by a common GRI. The diagram also indicates what types of policies or protocols are used at each stage. The access control is done at each stage, it may be related to different services but can use the same AuthZ service with different polices. At the reservation stage the AuthZ service can be integrated with one of the existing frameworks Web Services Agreement (WSAG) [5] or Service Level Agreement (SLA) negotiation [6].

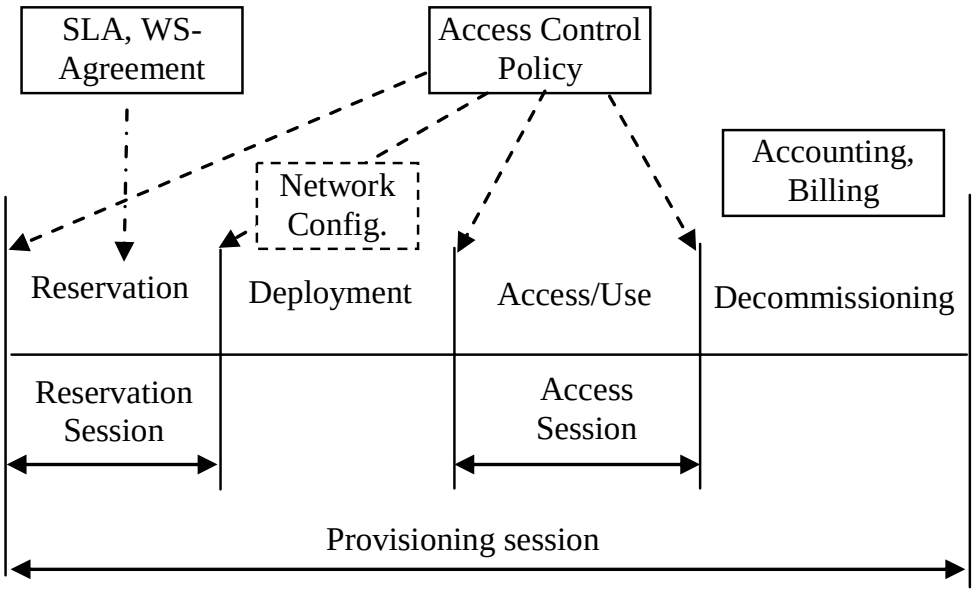

Figure 2.2. CRP stages and session types.

In multidomain NRP authorisation, tickets and tokens (described in details below) are used to transfer necessary security/authorisation context information between domains and serve as a session or access

credentials. Using these mechanisms ensure the integrity and consistency of the provisioning process. When used together, AuthzTicket and AuthzToken share the SessionId attribute which can be either a global or a local reservation/session ID.

2.3 AuthZ Token format to support Access Control and Signalling

2.3.1 Token types definition and XML token datamodel

The proposed GAAA-NRP architecture uses token extensively for access control and signalling at different NRP stages, considering it as a flexible and powerful mechanism for communicating and signalling security context between domains.

The token is defined as an abstract reference to the reservation or the AuthZ session context in domains using an abstract shared token meaning/context that is referenced by the token attributes. This definition is more oriented for the NRP provisioning model/workflow and extends the proposed token definition as shared abstract permission in earlier authors' paper [4].

Tokens can be used for both access control when accessing the reserved resources and for signalling during reservation and deployment stages. Correspondingly we distinguish the two major types of token in the GAAA-NRP architecture: access tokens and pilot tokens. Access tokens are used in rather traditional manner and described in details in [4]. Pilot tokens functionality and format were proposed and defined as a result of the current development of the AuthZ infrastructure as an integral component of the NRP.

After initial implementation in the GAAA-TK library released in the D4.3.1 deliverable (M22) the both access token and pilot token concepts have been integrated and tested in WP1 Harmony/NSP and WP2 G^{2}MPLS testbeds. This motivated changes both in extending the token data-model and adding methods to support new AuthZ scenarios that are discussed below.

Figure 2.3 illustrates the common data model (updated) of both access tokens and pilot tokens. 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 AuthZ service components, they can be distinguished by the presence or value of the token type attribute which is optional for access token and mandatory for pilot token.

Access tokens used in GAAA-NRP have a simple format and contain three mandatory elements: the *SessionId* attribute that holds the GRI, the *TokenId* attribute that holds the unique token ID attribute and is used for token identification and authentication, and the *TokenValue* element, and two optional elements: the *Condition* element that may contain two time validity attributes *notBefore* and *notOnOrAfter*, and the *Decision* element that holds two attributes *ResourceId* and *Result*, and an optional element *Obligations* that may hold policy obligations returned by the PDP.

The following access token types are defined:

AType1 – this pilot token type is used as authorisation session credentials and cryptographically binds SessionId/GRI, domainId and TokenId.

AType2 – extends ATP1 with the Obligations element that allows communicating policy obligations between domains.

The GAAA-NRP architecture defines four types of pilot tokens that have different profiles of the common data model and different processing/handling procedures:

PType1 – this pilot token type is used just as a container for communicating the GRI during the reservation stage. It contains the mandatory SessionId attribute and an optional Condition element (it does not contain a TokenValue element).

PType2 – this pilot token type is the origin/requestor authenticating token. Its TokenValue element contains a value that can be used as the authentication value for the token origin. The token value is calculated on the GRI by applying e.g. an HMAC function to the GRI together with the requestor symmetric secret or private key.

PType3 – this pilot token type extends the Type2 with a Domains element that allows to collect domains security context information (in the Domains/Domain element) when passing multiple domains during the reservation process. Such information includes the previous token and the domain's trust anchor or public key.

PType4 – this pilot token type is used at the deployment stage and can communicate between domains security context information about all participating in the provisioned lightpath or network infrastructure resources. This token type can be used for programming/setting up a TVS infrastructure for consistent access control tokens processing at the resource access stage.

When used together with an AuthzTicket the ticket and token identification elements `TokenID`, `SessionID`, and `Issuer` can be shared.

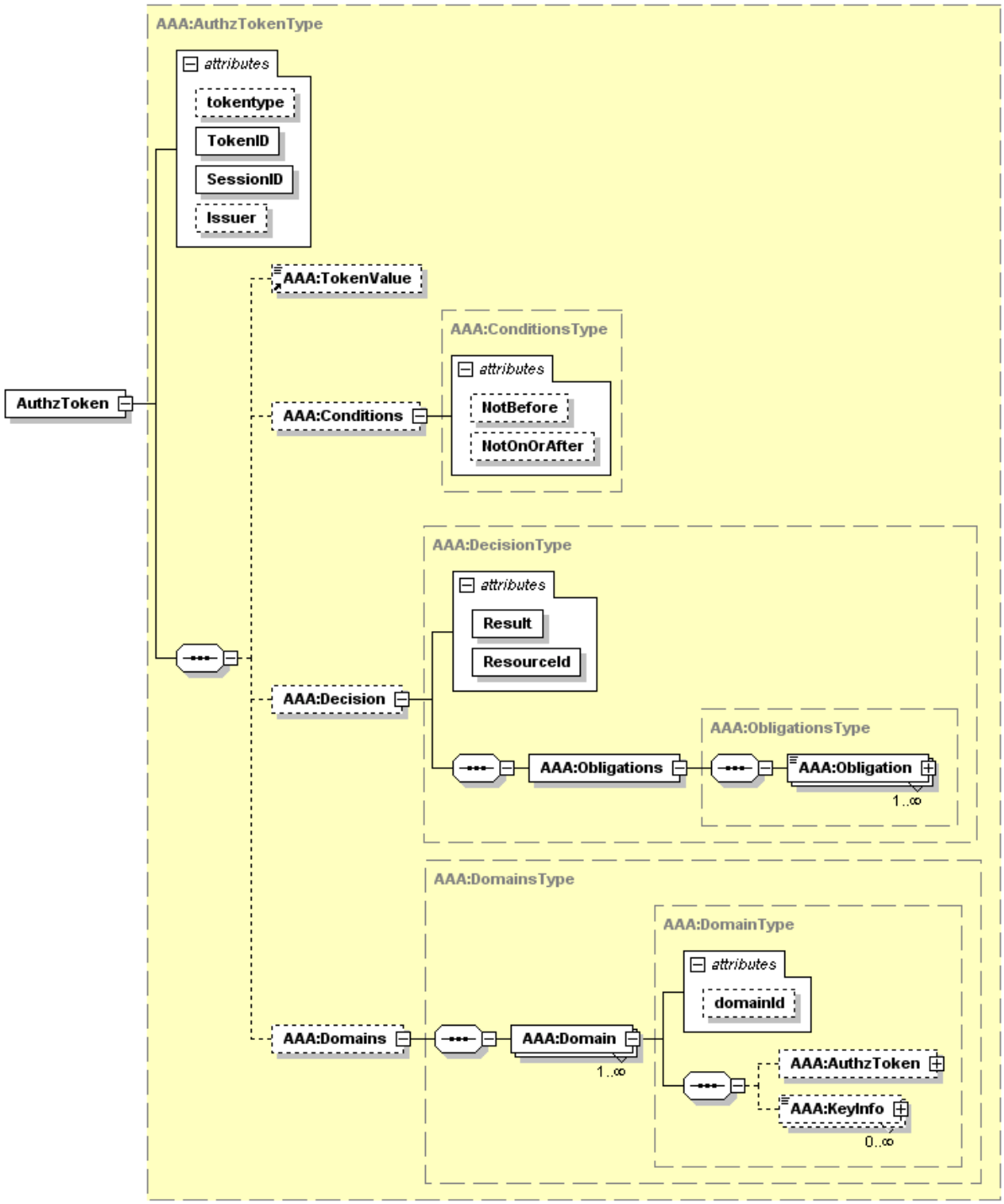

Figure 2.3. The common access and pilot tokens data model (updated).

The AuthzToken contains the following elements:

- The Root element attributes `TokenID`, `SessionID`, and `Issuer` that allow for the ticket unique identification and defines its binding to the session and domains related processes/authorities.
- The `TokenValue` element that holds the token value that cryptographically binds `TokenID`, `SessionID`, and `DomainID`.
- The `Conditions` element that contains actions which are permitted for the subject or its delegates.
- The `Decision` element that holds the PDP AuthZ decision bound to the requested resource and optionally can contain the `Obligations/Obligation` element.
- The `Domains` extendable element that may hold the security context (public key or trust anchor and the pilot token from that domain) from all previous domains that confirmed resource allocation for particular GRI.

When used together with an AuthzTicket, the ticket and token identification elements `TokenID`, `SessionID`, and `Issuer` can be shared. Examples of different token types are provided in section 4.

2.4 AuthZ Ticket formats to support extended AuthZ Session Management

An AuthzTicket has a complex but flexible format. The current AuthzTicket format and its implementation in the GAAAPI supports extended functionalities for distributed multi-domain hierarchical resources access control and user roles/permissions management, capabilities delegation and conditional AuthZ decision assertion (to support XACML policy obligations). It is one of the general design suggestions that an AuthzTicket should be easy to map to the SAML AuthzDecision Assertion [7] or to XACMLAuthzDecision Assertion defined by the SAML profile of XACML [8,9].

Diagram 2.4 illustrates the top and main AuthzTicket elements. The AuthZ ticket example is provided in Appendix C. Note, the Resource element is defined as an extendable in the AuthzTicket schema what allows to incorporate with different types of target resources (to which AuthZ decision is applied).

The AuthzTicket contains the following major groups of elements that allow for extended AuthZ session security context management:

- The Root element attributes `TicketID`, `SessionID`, and `Issuer` that allows for the ticket unique identification and defines its binding to the session and domains related processes/authorities.
- The `Decisions/Decision` element that holds the PDP AuthZ decision bound to the requested resource or service expressed as the `ResourceID` attribute.
- The `Resources` extendable element that may hold proprietary description of the reserved resource.
- The `Actions/Action` complex element that contains actions which are permitted for the subject or its delegates.
- The `Subject` complex element that contains all information related to the authenticated subject who obtained permission to do the actions, including sub-elements: `Role` (holding subject's capabilities), `SubjectConfirmationData` (typically holding AuthN context), and extendable sub-element

`SubjectContext` that may provide additional security or session related information, e.g. subject's VO, project, or federation.

- The `Delegation` element allows delegating the capabilities defined by the AuthzTicket to another subject(s) or community. The attributes define restriction on type and depth of delegation.
- The `Conditions` element specifies the validity constrains for the ticket, including validity time and the AuthZ session identification and additionally context. The extensible `ConditionAuthzSession` element provides rich possibilities for AuthZ context expression.
- The `Obligations/Obligation` element can hold obligations that a PEP/resource should perform in conjunction with the current PDP decision.

The semantics of AuthzTicket elements is defined in such a way that allows an easy mapping to related elements in SAML and XACML. The first three elements the `Decision`, the `Actions/Action`, and the `Subject` have a direct mapping to the related SAML elements. Other AuthzTicket elements the `Delegation`, the `Conditions`, and the `Obligations/Obligation` element, which are originated from XACML, can be implemented as extensible elements of the SAML `Condition` element.

The `SessionID` attribute, although defined as a general AuthZ session identifier in currently discussed Phosphorus use cases, holds either a global or local (to a domain) reservation identifier (GRI/LRI).

The AuthzTicket is digitally signed (as shown in the example) and cached by the resource's AuthZ service. To reduce communication overhead when using an AuthzTicket for consecutive request validations, the associated AuthZ token (AuthzToken) can be generated from the AuthzTicket. The AuthzToken may contain just two elements: `TokenID = TicketID` and `TokenValue = SignatureValue`, needed for the identification of the cached AuthzTicket.

The current AuthzTicket functionality is supported by the GAAA-TK library (see chapter 3 for details).

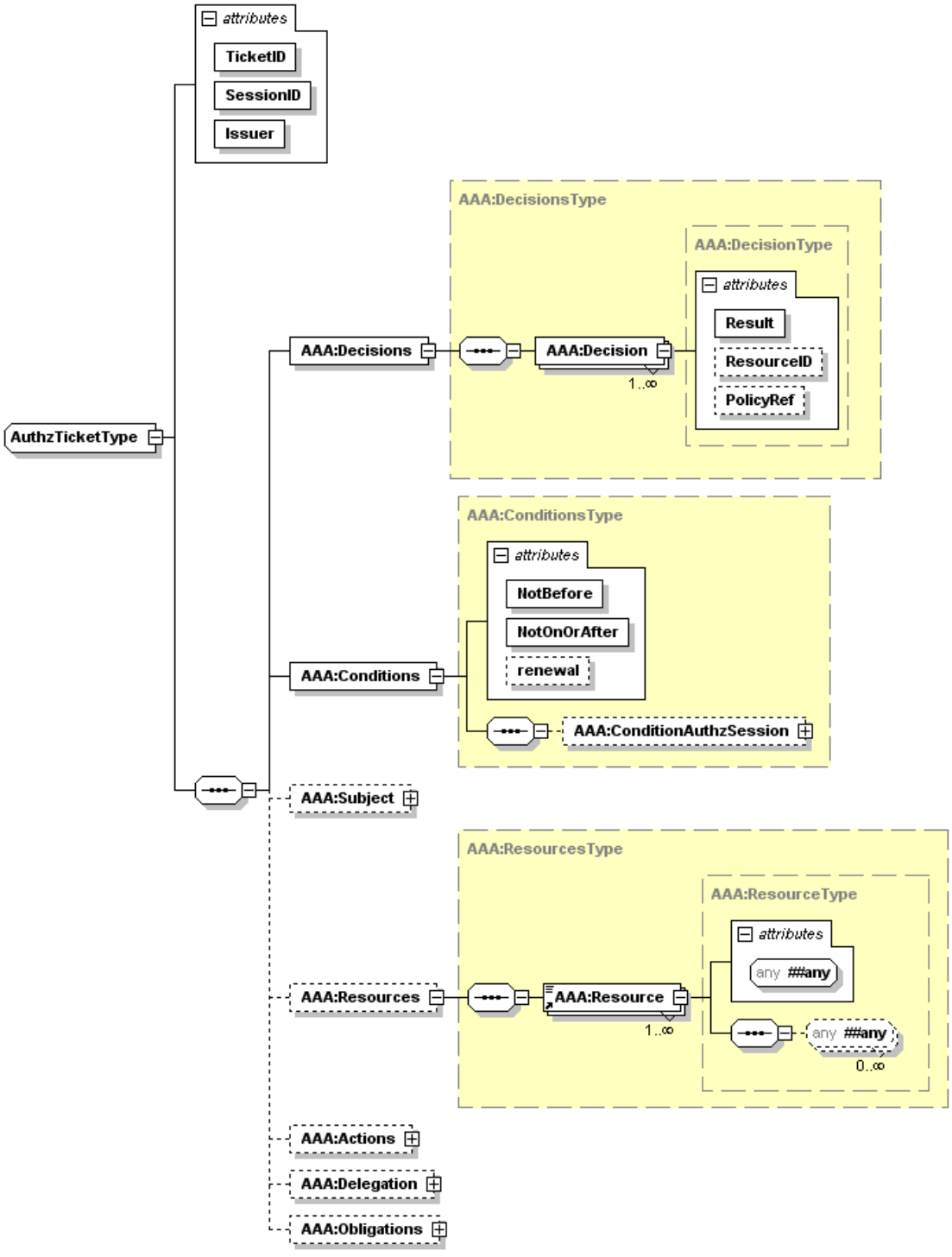

Fig. 2.4 AuthzTicket top and main elements (note, the Resource element is defined as an extendable).

2.5 Token Validation Service (TVS)

2.5.1 Basic TVS Functionality

The Token Validation Service (TVS) is a component of the GAAA-AuthZ infrastructure supporting token based policy enforcement mechanism during the user access of the reserved service or network. Basic TVS functionality allows checking if a service/resource requesting subject or other entity, that posses/presents current token, has right/permission to access/use a resource based on advance reservation to which this token refers. During its operation the TVS checks if the presented token has reference to a previously reserved resource and a request resource/service confirms to a reservation condition.

When using pilot tokens for signalling during interdomain path building, TVS can combine token validation from the previous domain and generation of the new token with local domain attributes and credentials. This scenario is supported by a special method “Validate&Relay”. This method requires checking incoming pilot token’s authenticity, which should be a part of the validation process.

Token handling scenarios and functionality are implemented as part of the PEP AuthZ calls (main GAAA-TK interface) or via direct calls to TVS.

In a simple/basic scenario, the TVS operates locally and checks a local reservation table directly or indirectly using a reservation ID (typically a Global Reservation Id - GRI). It is also suggested that in a multi-domain scenario each domain may maintain its Local Reservation ID (LRI) and its mapping to the GRI.

In more advanced scenarios the TVS should allow creation of a TVS infrastructure to support tokens and token related keys distribution to support dynamic resource, users or providers federations.

2.5.2 Token handling scenarios supported by TVS

The current TVS and GAAA-TK library design can support in-band token based policy enforcement (used in Token Based Networking (TBN) [10]), Control Plane token based signalling in G^{2}MPLS networks, and Service Plane access control and signalling.

The token generation and handling model can use both shared secret cryptography and public key cryptography and uses HMAC-SHA1 algorithm for calculating token value [11]. Current implementation uses shared secret, which for the sake of simplicity of testbed implementation is provided as a part of the TVS/GAAA-TK library distribution. The TokenKey is generated in the following way:

```
TokenKey = HMAC(GRI, tb_secret)
```

where

GRI – global reservation identifier,

tb_secret – shared Token Builder secret.

A token value is computed in a similar way but using TokenKey as a HMAC secret. For the purpose of authenticating token origin, the pilot token value is calculated of concatenated DomainID, GRI, and TokenID. This approach provides a simple protection mechanism against the pilot token duplication in the framework of the same reservation/authorisation session.

The following expressions are used to calculate the TokenValue for the access token and pilot token:

```
TokenValue = HMAC(GRI, TokenKey) – access token
```

```
TokenValue = HMAC(concat(DomainId, GRI, TokenId), TokenKey) – pilot token type 2 and 3
```

This algorithm allows for chaining token generation and validation process, e.g.:

```
"GRI-TokenKey-TokenValue => LRI-l_TokenKey-l_Token"
```

The key management model is not discussed at this stage of the project. The token handling model relies on the shared secret that is installed at all participating NRPS nodes. It is being investigated that current model can be replaced with the IBC (Identity Based Cryptography) [12, 13] that will allow to replace shared secret token handling model that has known manageability problems.

The current TVS implementation allows handling both types of tokens, access tokens and pilot tokens, and also supports access tokens in binary and XML format. In both cases reservation token is tuple of GRI and TokenKey that should be included into the request or service request.

2.5.3 Storing session context in the TVSTable

To provide consistency in using XML tokens for managing provisioning or authorisation session context, the TVS stores token related to domain specific security context in the TVSTable that is used as a kind of cache for storing XML tokens and related local (for domain) security context. During XML token processing, TVS retrieves from the TVSTable stored security context based on the presented token, in particular, using token's sessionID/GRI and token's domain, and compares it with the information provided in the AuthZ request.

TVSTable provides a simple solution for storing/caching session security context and in the future can be replaced with database solution if it is available with application. Figure 2.5 below illustrates the structure of the TVSTable information (full TVSTable example for two domains can be found in Appendix D).

```
<TVSTable DomainLocal="http://testbed.ist-phosphorus.eu/viola">
  <TVSEntry DomainId="http://testbed.ist-phosphorus.eu/viola">
    <SessionContext
SessionId="186c435871bb50df6ab69d2e244f856cd7e9d84896b0dbe1792993ae18f9d423">
      <Conditions NotBefore="2009-03-06T12:41:54.687Z" NotOnOrAfter="2009-03-
08T12:41:54.687Z"/>
      <Action>create-path</Action>
      <Subject Id="subject">
        <SubjectId>WHO740@users.testbed.ist-phosphorus.eu</SubjectId>
```

```
        <SubjectRole>researcher</SubjectRole>
        <SubjectContext>demo041</SubjectContext>
      </Subject>
      <Resource>
        <ResourceId>http://testbed.ist-phosphorus.eu/viola/harmony</ResourceId>
        <ResourceSource>10.3.1.16</ResourceSource>
        <ResourceTarget>10.7.3.13</ResourceTarget>
      </Resource>
      <KeyInfo keytype="public">http://testbed.ist-
phosphorus.eu/viola/_public_key_/186c435871bb50df6ab69d2e2484...9d423</KeyInfo>
    </SessionContext>
  </TVSEntry>
</TVSTable>
```

Figure 2.5. Datastructure of the TVSTable.

TVSTable stores information for each of domains participating in the provisioning process by creating a separate TVSEntry element with DomainId attribute, each TSVEntry element may contain multiple SessionContext elements identified by SessionId = GRI attribute. Currently stored session related information contains the following information:

- Conditions element that specifies the token validity time
- Subject related attributes
- Resource related attributes
- Action related attributes
- KeyInfo element that may contain either public key or Public Key Certificate

2.6 Policy Obligations to support inter-domain GAAA-NRP scenarios

Policy obligation is one of the authorisation policy enforcement mechanisms that allows adding AuthZ decision enforcement components that can not be defined in the policy at the moment of making policy decision by the PDP, or may not be known to the PDP or policy administrator/writer. The obligations can be also included in the extended access token context (see token data-model in Fig. 2.2). The GAAA-TK library implements the Obligations Handling Reference model described in the deliverable D4.3.1.

Suggested functionality that can be achieved by using obligations includes but are not limited to:

- Intradomain network/VLAN mapping for cross-domain connections, that can be used to map external/interdomain border links/TNA's to internal VLAN and sub-network
- Account mapping
- Type of service (or QoS) assigned to a specific request or policy decision
- Quota assignment

- Service combination with implied conditions (e.g., computing and storage resources)
- Usable resources/quota

The text below provides current suggestions for the obligations definition. More details will be provided with wider use and acceptance of the XACML-NRP profile.

a) Intradomain network/VLAN mapping

This may be needed for defining specific intra-domain mapping of cross-domain connections depending on specific reservation, path or user attributes.

b) Network user identity mapping

This obligation is returned by the PDP in case of positive decision with instruction to what type of or a specific pool account the user identity should be mapped when accessing a requested network resource.

The need of account mapping may exist in cases when domain based Network Resource Provisioning Systems (NRPS) have pre-installed/built-in pool accounts to which are different types or quality of service are assigned. In such situations, an authorised user needs to use one of such accounts, e.g. “silver”, “golden”, “platinum”. A number of different individual accounts of the same type may be limited; consequently a dynamically assigned account should be selected from the pool of available or free accounts. Such dynamic account assignment can not be specified in the typically stateless policy and cannot be done by PDP. However, the access control policy may contain instruction to PEP to do such mapping.

c) Usability and accounting

Usability and accounting obligations allow that some usability attributes (e.g. number of downloads, total time of using network resource, amount of data transferred) assigned or accounting instruction are applied to the specific request decision.

2.7 XACML policy profile for NRP

The GAAA-NRP authorisation infrastructure uses and GAAA-TK library implements the XACML-NRP attributes and policy profile for network resource provisioning described in details in the project deliverable D4.3.1.

This section only provides the general assumptions for the XACML-NRP profile definition, which provided background requirements for the GAAA-TK library design.

The XACML-NRP profile specifies a set of resource, subject, actions and environment attributes that are considered relevant to the policy definition and are critical for inter-domain interoperability. Policy example and corresponding Request and Response messages are provided in Appendix B.

2.7.1 Access Control in NRP – Basic Use Cases

In general, the access control policy comprises of rules and conditions that specify what user with what attributes may access or execute what action on the resource with what attributes.

Two basic use cases for access control in Network Resource Provisioning (NRP) can be expressed in a simple narrative form:

Use case 1: "User A is only allowed to use user endpoints X, Y and Z", or

Use case 2: "User A is only allowed to use endpoints in domain N and M".

Two other use cases are related to the more complex access control scenarios that take place during the multidomain/multiple resources reservation and require simple delegation functionality:

Use Case 3: User/Group A is only allowed to invoke method X, Y, and Z

Use Case 4: User/Group A is only allowed to invoke method X,Y, and Z based on session delegation

Supporting these use cases requires introducing simple group or session based delegation with full or limited delegation profile that should be supported by both new PEP and related policy definition.

The following assumptions are made to satisfy the above use cases and to ensure effective management of policies, user and resource identities and attributes:

1) Users and resources are identified by their unique ID's and may have also assigned attributes, which for the user may include such attributes as user group, role, or federation, and for the resource may include such attributes as domain/subdomain, resource type, level of service; in more complex scenarios the requested service may specify full or partial path;

2) Users and resources (domains and endpoints) may be organised/associated into administrative and or security domains or federations, i.e. a user and a resource can be a member of one or multiple associations;

3) Different domains and endpoints participating in network connection (for which the authorisation is requested) may belong to different federations or security associations;

4) Only authenticated user may have access to protected resources; user authentication is confirmed by issuing AuthN assertion by trusted AuthN service or creating user related security context environment of the started process.

5) User authentication may be resulted in the following: service or process session initiation; or release of the user attributes or credentials,

6) Depending on the user attributes (federations, groups, roles) the user can be assigned specific level of service;

7) A simple policy format will contain rules that express conditions on what user attributes are required to access the resource with specific attributes or execute a specific action on the resource with specific attributes.

8) Policy may need to handle security/authorisation context from the previous domain and be able to specify information that should be sent to the next domain, if conditional decision need to be enforced.

9) Delegation intra and inter-domain may use either policy rules or special PEP methods.

3 GAAA Toolkit Library

This section provides information about the current implementation of the basic GAAA-NRP functionalities in the pluggable GAAA Toolkit library.

3.1 GAAA-TK library packages structure

The GAAA-TK library uses the "**org.aaaarch**" namespace and has the following structure of packages:

Package	Functions
org.aaaarch.gaaapi	The main GAAAPI package that includes PEP class and all supporting and helper classes that constitute the ContextHandler functionality
org.aaaarch.gaaapi.tvs	The TVS package comprising TVS class and other classes supporting TVS operation including XML token and TVS table or database connectors.
org.aaaarch.gaaapi.impl.pdp org.aaaarch.gaaapi.impl.pep	The supporting packages that include classes adopting main GAAAPI functionality or external libraries, in particular application specific PEP implementation or for using XACML library as PDP.
org.aaaarch.policy	Supports policy resolutions and retrieval from the local repository or callout to external policy repository.
org.aaaarch.gaaapi.obligations	Supports policy obligations handling and includes handlers for supported obligation types.
org.aaaarch.gaaapi.authn	Currently provides verification and validation of different authentication credentials and supports identity and attributes mapping as a part of validation process.
org.aaaarch.config org.aaaarch.crypto	Supporting packages for the whole library that handle common configuration including constants and namespaces and cryptographic functionality including keystore interface
org.aaaarch.impl.saml	Supporting packages that provide support for different types

org.aaaarch.impl.unicore org.aaaarch.impl.edugain org.aaaarch.impl.signature	credentials and adopt external original libraries, in particular SAML2 assertions, UNICORE6 assertions, and eduGAIN credentials
org.aaaarch.gaaapi.session org.aaaarch.gaaapi.ticktok	The packages supporting authorisation sessions management and used for handling AuthZ tickets and tokens as session credentials.
org.aaaarch.utils org.aaaarch.policy.utils	Common utility classes supporting common input/output and data converting functions. Policy utility providing simple policy generation and editing tools.

3.2 General GAAA-NRP API and programming examples

The GAAAPI package provides all necessary functionalities to smoothly integrate AAA/AuthZ services into target application. GAAAPI package is provided together with the PEP implementation and simple XACML PDP implementation.

PEP-GAAAPI is called from the application AuthZ gateway that extracts necessary information from the service request and creates AuthZ request to PEP. GAAAPI functionality supports all necessary communication between PEP and PDP and depending on implementation may include also external callout to such components as PDP, PAP, Attribute Authority Service (AAS), TVS, and Obligation Handlers (OH).

The current updated version of the GAAA-TK library supports basic session management functionality that includes two core methods that use AuthZ tickets directly and two other extended methods that support either simple AuthZ session management with AuthZ ticket or can be used in more advanced scenarios that allow tickets and tokens renewal and re-generation. The latter extension came out of practical GAAA-TK library integration into the WP1 and WP2 testbeds using NSP/Harmony and G^{2}MPLS systems correspondingly. The token renewal and re-generation functionality is specifically targeted for interdomain security context handling during reservation or path creation process in G^{2}MPLS.

The authorisation scenarios used in WP2 G^{2}MPLS are described in the WP2 deliverable D2.8 “Design of the Grid-GMPLS Control Plane to support the Phosphorus Grid AAI” that was developed in tight cooperation between WP2 and WP4. The GAAA profile for G^{2}MPLS uses both access tokens for access control and pilot token types 2 and 3 for signalling. In more advanced scenario, the AuthZ ticket can be communicated as a part of the pilot token type 3 domain context (see token types definition above).

The TVS provides a number of methods to support access tokens and pilot tokens handling and related session context management. In typical GAAA-TK library use these methods are called from the PEP, however they can be also called directly using the TVS interface.

3.2.1 PEP-GAAAPI interface

PEP-GAAAPI interface provides a few commands/methods to request policy based AuthZ decision depending on the set of provided information:

a) Method #1 should either return a logical value "True" or "False", or throw the appropriate exception

```
       Boolean org.aaaarch.gaaapi.PEP.authorizeAction
              (String resourceURI, String actions, HashMap subjmap)
       throws java.lang.Exception,
       org.aaaarch.gaaapi.NotAuthenticatedException,
       /* user subjconfdata (i.e. authenticationToken) is not valid */
       org.aaaarch.gaaapi.NotAvailablePDPException;
       /* PEP could not reach PDP, or other internal PDP error*/
where
@ resourceURI – Resource ID in a form of URI
@ actions – requested actions (currently supported only one action)
@ {subjmap} set of values (subject-id, subject-confdata, subject-role, subject-context)
@ subject-id - subject Id in form of RFC822
@ subject-confdata - AuthN token or SAML AuthN assertion
@ subject-role - role for the particular request (may be in a form either simple
     attribute or RQAN)
@ subject-context - subject context, e.g. Experiment, VO, or VLab in which
     the subject and resource attributes are defined
```

Note: This method uses complex resource URI that may consist of ResourceId part and additional parameters in a form of “name=value” pairs (see section 4.2.4 for attribute expression conventions).

b) Method #2 should either return a logical value "True" or "False", or throw the appropriate exception

```
       Boolean org.aaaarch.gaaapi.PEP.authorizeAction
              (HashMap resmap, HashMap actmap, HashMap subjmap)
       throws java.lang.Exception,
       org.aaaarch.gaaapi.NotAuthenticatedException,
       org.aaaarch.gaaapi.NotAvailablePDPException;
where
@ resmap – set of the Resource related attributes in a form or HashMap
@ resmap = (resource-id, resource-domain, resource-type) and other resource
             related attributes
@ actmap – requested actions (currently supported only one action)
```

c) Method #3 should either return a logical value "True" or "False", or throw the appropriate exception

```
       Boolean org.aaaarch.gaaapi.PEP.authorizeAction
              (String resourceId, String actions, String subjectId, String subjconfdata,
               String roles, String subjctx)
       throws java.lang.Exception,
       org.aaaarch.gaaapi.NotAuthenticatedException,
       org.aaaarch.gaaapi.NotAvailablePDPException;
```

Note: This method uses simple resource ID format. All additional parameters will be ignored and not used for policy resolution.

d) Method #4 should either return a logical value "True" or "False", or throw the appropriate exception

```
      Boolean org.aaaarch.gaaapi.PEP.authorizeAction
            (String authzToken, HashMap resmap, HashMap actmap, HashMap subjmap)
      throws java.lang.Exception,
      org.aaaarch.gaaapi.NotAuthenticatedException,
      org.aaaarch.gaaapi.NotAvailablePDPException;
where
@ authzToken – access token in a form of XMLToken
```

e) Method #5 should either return a valid AuthorisationTicket or AuthorisationToken (refer to section 2 and Appendix for AuthzTicket and AuthzToken format and examples), or throw the appropriate exception

```
      String org.aaaarch.gaaapi.PEP.authorizeAction
            (String authzTicketToken, String sessionId, String resourceURI,
             String actions)
      throws java.lang.Exception,
      org.aaaarch.gaaapi.NotAuthenticatedException,
      org.aaaarch.gaaapi.NotAuthorizedException,
      org.aaaarch.gaaapi.NotAvailablePDPException;
were
@ authzTicketToken – AuthZ ticket or token containing all necessary AuthZ session context
@ sessionId – Session ID that can be also a Global or Local reservation ID (LRI/GRI)
```

f) Method #6 should either return a valid AuthorisationTicket or AuthorisationToken (refer to section 2 and Appendix for AuthzTicket and AuthzToken format and examples), or throw the appropriate exception

```
      String org.aaaarch.gaaapi.PEP.authorizeAction
            (String authzTicketToken, String sessionId, String resourceURI,
             String actions, HashMap subjmap)
      throws java.lang.Exception,
      org.aaaarch.gaaapi.NotAuthenticatedException,
      org.aaaarch.gaaapi.NotAuthorizedException,
      org.aaaarch.gaaapi.NotAvailablePDPException;
```

The following new methods are suggested to support more flexible session based AuthZ scenarios in WP1 Harmony testbed:

g) Method #7 should either return a boolean value Permit or Deny, or throw the appropriate exception

```
      boolean org.aaaarch.gaaapi.PEP.authorizeActionSession (String authzToken,
            String griREq, int delegtype,
            HashMap resmap, HashMap actmap, HashMap subjmap)
      throws java.lang.Exception,
      org.aaaarch.gaaapi.NotAuthenticatedException,
      org.aaaarch.gaaapi.NotAuthorizedException,
      org.aaaarch.gaaapi.NotAvailablePDPException;

where
      @ delegtype - enumerated delegation types (for the resource)
```

This method allows for flexible session based access control and delegation where AuthzToken is used as a session credential. It supports the following simple delegation scenarios where the session permissions

obtained by a privilege user (e.g. researcher, principal investigator) can be delegated to other user depending on session-delegation modes.

The delegation type attribute defines the following session delegation scopes:

0 - strict session based delegation (only authorised roles for only authorised actions - PDP/policy based evaluation)
Use: Privilege role can start session/reserve path and all authorised users can use
1 - full session delegation (all actions for all role, i.e. just checking validity of token)
Use: token based access control: any owner of the token can perform any action (Warning: recommended only in the controlled environment)
2 – policy allowed actions for all legitimate roles
(Note: resmap can contain only (resource-realm, resource-domain; resource Ctx is retrieved based on token)
3 - controlled delegation (require extended AuthzTicket format; delegation defined by AuthzTicket context)
(Note: resmap can contain only (resource-realm, resource-domain), subjmap can be null or contain only subject-context, resource and subject Ctx is retrieved based on token)
4 - controlled delegation, defined by the special delegation policy or AuthzTicket context (not supported yet)

h) Method #8 should either return a new session/AuthZ token (the same or different type depending on configuration), or string “Deny” or "Permit" depending on the PDP decision

```
String org.aaaarch.gaaapi.PEP.authorizeActionSession (String authzToken,
       String griReq, int sescred, boolean renew,
       HashMap resmap, HashMap actmap, HashMap subjmap)
throws java.lang.Exception,
org.aaaarch.gaaapi.NotAuthenticatedException,
org.aaaarch.gaaapi.NotAuthorizedException,
org.aaaarch.gaaapi.NotAvailablePDPException;

where
       @ secred – session security credentials type (enumerated) that is expected
              to be returned
       @ renew – indicates if the presented credentials should be renewed
```

This method supports either local domain session based access control or can be used for “chained” AuthZ decisions request like in case of multidomain path creation in G^{2}MPLS. This method relays on the TVS method validateAndRelayPilotToken (String pilotToken, byte[] tokenKey) described below.

The following sescredtype enumerated types are supported:

1 - pilot token type=1 (not supported in current version)
2 - pilot token type=2
3 - pilot token type=3 (not supported by this method)
10 - access token type=0
11 - access token type=1 (not supported by this method)

20 - AuthzTicket (not supported in current version)
30 - SAML/SAML-XACML assertion (not supported in current version)

Usage suggestions:

1) authorizeActionSession (null, *, sescred, resmap, actmap, subjmap)
- return session cred of sescredtype, or string ("Permit" | "Deny") if sescredtype id not supported

2) authorizeActionSession (authzToken, delegtype, *, resmap, actmap, subjmap)
- return "Permit" if (authzToken is VALID for the domain AND PDPdecision=True), or "Deny" if either authzToken INVALID OR policy/delegation is negative

3) authorizeActionSession (authzToken, 1, *, resmap, *, *)
- return "Permit" if authzToken VALID for the domain

k) Method #9 should either return a renewed session/AuthzToken if renew = (1,2) or token is not provided and requested sescred supported, or string “Deny” or "Permit" depending on the PDP decision. Extends method #8 for inter-domain reservation/access control scenario, e.g. request contains pilot token (or local AuthzToken or AuthzTicket) from the previous domain as session credential, and supports inter-domain delegation types.

```
String org.aaaarch.gaaapi.PEP.authorizeActionSession (String authzToken,
       String griReq, int delegtype, int sescredtype, int renew,
       HashMap resmap, HashMap actmap, HashMap subjmap)
throws java.lang.Exception,
org.aaaarch.gaaapi.NotAuthenticatedException,
org.aaaarch.gaaapi.NotAuthorizedException,
org.aaaarch.gaaapi.NotAvailablePDPException;

where
       @ secred – session security credentials type (enumerated) that is expected
            to be returned
       @ renew – indicates if the presented credentials should be renewed
         0 - no renewal, return the same token
         1 - renew with the same GRI/sessionId
         2 - renew with new GRI (old GRI will be included into PTT(3,4) context)
```

Positive AuthZ decision is made if the session credentials are valid (Note: no session context is available for the previous/other domain), and policy/PDP decision is "Permit" and new session credentials are returned. The method also supports basic delegation scenarios.

3.2.2 Simple XACML PDP API

The main GAAA-TK use suggests that the XACML PDP is requested via PEP that converts the application specific AuthZ request to XACML Request format. However, it is possible to request the XACMLPDPsimple class directly via the following API:

a) Method #1 return XACML Response message as String, or throws an exception

```
String org.aaaarch.gaaapi.impl.pdp. XACMLPDPsimple.requestPDP
       (RequestCtx request, String policyref)
throws java.lang.Exception
```

a) Method #2 return XACML Response message as String, or throws an exception

```
       String org.aaaarch.gaaapi.impl.pdp. XACMLPDPsimple.requestPDP
              (String requestStr, String policyref)
       throws java.lang.Exception
were
@ request, or requestStr – XACML request in a form of the XACML RequestCtx or String
@ policyref – path to policy file location
```

Examples of the XACML Request/Response messages and corresponding XACML policy are provided in Appendix B.

3.2.3 Subject authentication verification with AuthenticateSubject class

The AuthenticateSubject class supports 3 basic methods that are typically called from the PEP but can also be called directly.

1) the main method that receives a set of subject attributes, including SubjectId and SubjectConfirmationData, and return either enumerated value "aaa:authn:gaaapi:subject:valid" or "aaa:authn:gaaapi:subject:invalid" as a return value for the SubjectConfirmationData attribute.

```
       public static HashMap validateSubjectAttributes (HashMap subjmap)
              throws Exception
were
@ subjmap – subject HashMap containing a set subject attributes in a form of
       “name-value” pairs
```

If configured this method can also call a local or remote Attribute Authority to do attributes translation or mapping, in particular if there is a need to convert/translate attributes from one domain to another or in case internal resource system operation requires mapping subject attributes to one of internal pre-defined pool accounts

2) binary authentication method that generates SubjectConfirmationData crypto string as a result of applying either Des or HMAC transformation to the SubjectId value

```
       public static String getSubjectAuthnCrypto
                     (String subjectId, String authnMethod, String keypasswd)
              throws HMACProcessorException, NotSupportedAuthnMethodException
were
@ subjectId – subject ID in a form of X.521/LDAP, RFC822 or arbitrary URN/URI string
@ keypass – private keystore pass; if “null” used default private key
@ authMethod – binary AuthN method either HMAC or DES that correspondingly indicated by
enumarated value "aaa:authn:gaaapi:method:hmac" or "aaa:authn:gaaapi:method:des"
```

3) XML authentication method that creates Subject Authentication assertion in a form of XML AuthZ ticket or token, Unicore6 SAML assertion, or SAML2 AuthN assertion.

```
       public static String getSubjectAuthnXML
```

```
                    (HashMap subjmap, String credtype, String keypasswd)
             throws Exception
were
@ subjmap – subject HashMap containing a set subject attributes in a form of
       “name-value” pairs
@ keypass – private keystore pass; if “null” used default private key
@ credtype – defines type of returned XML AuthN assertion;
       the following enumerated types are supported:
       1 - "AAA:AuthzTicket"
       2  "AAA:AuthzToken"
       3 - "AAA:AuthnTicket"
       4 - "AAA:AuthnToken
       10 - "urn:Assertion"
       20 - "saml:Assertion"
```

3.2.4 GAAA-PEP API programming examples

1) Preparing PEP request data

Two types of data are used as input to PEP.authoriseAction methods – string variables and HashMap attributes set. The example below illustrates how to put String variables into HashMap and how to extract individual attribute from HashMap.

```
HashMap<String, String> subjmap = new HashMap<String, String>();
HashMap resmap = new HashMap();
HashMap actmap = new HashMap();

// Obtaining test set of the subject attributes
subjmap = SubjectSet.getSubjSetTest();
// extracting subject attrs from the subjmap
String subjectId = subjmap.get(ConstantsNS.SUBJECT_SUBJECT_ID).toString();
String subjconfdata = subjmap.get(ConstantsNS.SUBJECT_CONFDATA).toString();
String roles = subjmap.get(ConstantsNS.SUBJECT_ROLE).toString();
String subjctx = subjmap.get(ConstantsNS.SUBJECT_CONTEXT).toString();
// modifying subjctx for experiments

//Example Subject attributes
String subjectId = "WHO740@users.collaboratory.nl";
String subjconfdata = "SeDFGVHYTY83ZXxEdsweOP8Iok";
String roles = "researher";
String subjctx = "demo001";

//Putting/replacing Subject attributes into subjmap
subjmap.put(ConstantsNS.SUBJECT_ROLE, subjrole);}
subjmap.put(ConstantsNS.SUBJECT_SUBJECT_ID, "WHO750@users.testbed.ist-phosphorus.eu");

//Obtaining resource map from the ResourceHelper class
resourceInputURI = "http://testbed.ist-
phosphorus.eu/viola/harmony/source=10.3.1.16/target=10.7.3.13";

resmap = ResourceHelper.parseResourceURI(resourceInputURI);

// Putting action attributes into actmap
actmap.put(ConstantsNS.ACTION_ACTION_ID, action);
```

Note this example uses Subject, Resource, Action constants that define attributes name/ID's in correspondence to the XACML-NRP profile.

2) Requesting PEP with (Subject, Resource, Action) information (using method#1 and method#2)

The following two examples explain how to call PEP method #1 and method #2 using prepared data like above.

```
// Calling PEP method #1

boolean decision = PEP.authorizeAction (resourceInputURI, actions, subjmap);

// Calling PEP method #2

boolean decision = PEP.authorizeAction (resmap, actmap, subjmap);
```

2) Requesting PEP with (Subject, Resource, Action) information and XMLToken obtained from TVS (using method #4)

To enable XML token operation, the following steps need to be performed (this a combined use of the PEP and TVS functionality explained in details below):

a) obtain positive decision from PEP-PDP (see examples 1 and 2 above)
b) save session/reservation context in the TVS table
c) generate XML token that contains GRI (global reservation identifier) and token value generated cryptographically of GRI and the domain token key
d) present this token in all consequent AuthZ requests to PEP. In this case PEP will request AuthZ request evaluation with TVS, TVS will retrieve session context from TVS table and compare it with the request context without requesting PDP.

```
// Obtaining or setting domain ID information

String domainViola = ConfigDomainsPhosphorus.DOMAIN_PHOSPHORUS_VIOLA;
String domainId = domainViola;
// Obtaining sessionId if not received with the pilot token
String griprefix = "";
String sessionId = GRIgenerator.generateGRI(32, griprefix);

// Composing session context vector
Vector sessionCtx = TVS.getSessionCtxVector (domainId, gri, resmap, actmap, subjmap);

/* If it is necessary, the TVSTable can be purged completely of for a selected domain
 * Use this method
 *  TVSTable.purgeTVSTable(null, 0);
 */

// Adding a new TVS entry
TVS.setEntryTVSTable(domainId, gri, sessionCtx);

// checking TVS table content
String tablefile = TVS.getTVSTableFile ();
Document tabdoc = HelpersXMLsecurity.readFileToDOM(tablefile);
HelpersXMLsecurity.printDOMdoc(tabdoc); // pring TVSTable context if you want
```

```
// Generating XMLToken (type 0)
// Set or obtain necessary variable for XMLToken generation
Boolean simple = false; // simple token format doesn't contain time validity

Int validtime = TVS. getConfigValidityTimeDefault();
Int validtime = 1440; // validity time is 24 hrs

String tokenxml = TokenBuilder.getXMLToken(domainId, gri, null, validtime, simple);

//Request PEP with XMLToken (method #4)

boolean decision = PEP.authorizeAction (tokenxml, resmap, actmap, subjmap);
```

3.3 TVS API and programming examples

TVS functions are normally called through PEP when using corresponding methods and can also be called directly using TVS methods described below.

3.3.1 TVS interfaces

a) Token Builder commands

```
public static byte[] TokenBuilder.getBinaryToken(String gri, byte[] tokenkey)

public static String TokenBuilder.getXMLToken(String domainId, String gri, byte[]
tokenKey, int validtime, boolean simple)

public static String TokenBuilder.getXMLTokenPilot(String domainId, String gri, String
domain, int validtime, byte[] tokenKey, int ptokentype, String tokenCtx)
```

b) TVS token validation interface - validates the binary or XML token thenselves;

```
public static boolean validateBinaryToken (String token, String gri, byte[] tokenKey)
       throws Exception

public static boolean validateXMLToken (Document aztdoc, byte[] tokenKey)
       throws Exception,
       MalformedXMLTokenException,
       NotValidAuthzTokenException

public static boolean validateXMLToken (String authzToken, byte[] tokenKey)
       throws Exception,
       MalformedXMLTokenException,
       NotValidAuthzTokenException
```

c) PEP-TVS interface: is called from PEP and validates AuthZ Request (resmap, actmap, subjmap) against XML token;

```
public static boolean validateAuthzRequestByToken (String authzToken,
             HashMap resmap, HashMap actmap, HashMap<String, String> subjmap)
      throws Exception,
      MalformedXMLTokenException,
      NotValidAuthzTokenException
```

d) Internal TVSTable programming interface use the following basic commands:

```
TVS.setEntryTVSTable(String domainId, String gri,
                     HashMap resmap, HashMap actmap, HashMap subjmap)

TVS.getEntryTVSTable(String domainId, String gri)

TVS.deleteEntryTVSTable(String domainId, String gri)

public static boolean purgeTVSTable (String domainId, int expireTime)
```

Note, that TVS programming calls will be exposed as We Services.

e) Interdomain signaling with XML pilot tokes

The new TVS token validation method validates input pilot token and, in case of its validity, generates a new token using pre-configured local domain properties such as DomainId, domain tokenKey and can also be configured to either use the same GRI or generate a new one.

```
public static String validateAndRelayPilotToken (String pilotToken, byte[] tokenKey)
      throws Exception
```

f) External/WS TVS programming interface

External TVS interface will allow programming TVS table by sending particular reservation information in a SOAP message.

3.3.2 TVS programming examples

1) Generating binary token
To request token generation from the calling application, use these commands/methods:

```
// Prepare data for token generation
String gri = "".concat(org.aaaarch.gaaapi.common.IDgenerator.generateID(20).toString());
byte[] tokenkey = TokenKey.generateTokenKey(gri);

byte[] token = TokenBuilder.getBinaryToken(gri, null);
```

2) Generating XML token

```
// Set parameter for token generation
boolean simple = true; // simple token doesn't contain Conditions element
int validtime = 0;     // this variable sets token validity in minutes; default is 24 hrs
String domainId = "http://testbed.ist-phosphorus.eu/viola/harmony";
```

```
//
String gri = GRIgenerator.generateGRI(20).toString();

String tokenxml = TokenBuilder.getXMLToken(domainId, gri, null, validtime, simple);
```

3) Validating binary token

```
boolean valid = TVS.validateBinaryToken (token, gri, null);
```

4) Validating XML token itself. Two methods can be used to validate full token validity and time validity:

```
XMLTokenType token = new XMLTokenType (tokendoc);
// Checking token validity time
boolean timevalid = token.isTimeValid(token);

// checking token validity
boolean valid = TVS.validateXMLToken (tokendoc, null);
```

4) Validating AuthZ request context against XML token

```
//Prepare input data for requesting TVS

// If XMLToken is received as String
Document tokendoc = HelpersXMLSecuritry.readStringToDOM(String) (tokenString);

XMLTokenType token = new XMLTokenType (tokendoc);

// Simple token time validity check
boolean timevalid = token.isTimeValid(token);

// Validating token against stored in TVS table session context
TVS.validateXMLToken(tokendoc, null);

// Validating AuthZ request against XML token and stored session context
boolean confirmed = TVS.validateAuthzRequestByToken (aztstr, resmap, actmap, subjmap);
```

3.4 Attribute expression conventions

Information provided in the AuthZ request to PEP-PDP contains information about Resource, Subject, Action, and optionally Environment.

a) Resource attributes

Currently the Resource variable in the AuthZ request contains one attribute ResourceId in the form of URI string that includes the network resource identifier and a list of parameters used for policy-based request evaluation. When sending a XACML Request to XACML PDP the input URI string is converted into the HashMap `resmap` that contains a set of resource related attributes. The names for some of the relevant resource attribute identifiers are taken from the XACML-NRP profile, such as “resource-id”, “resource-domain”,

"resource-realm", "resource-type", "source", "target", etc., however it is responsibility of the application developer to correctly format the ResourceId URI string.

The following ResourceId formats are supported:

a) http://testbed.ist-phosphorus.eu/{domain}/{device | service}/{parameters}

Example: http://testbed.ist-phosphorus.eu/viola/harmony/source=10.7.12.2/target=10.3.17.3,

This URI will be converted into the following set of attributes and put into the resmap:

```
resource-id = http://testbed.ist-phosphorus.eu/viola/harmony
resource-realm = http://testbed.ist-phosphorus.eu
resource-domain = viola
resource-type = harmony
source = 10.7.12.2
target = 10.3.17.3
```

b) http://testbed.ist-phosphorus.eu/resource-type/{resource-type-name}

Example: http://testbed.ist-phosphorus.eu/resource-type/harmony

c) http://testbed.ist-phosphorus.eu/resource-context/{(project | association) name} (optional)

Example: http://testbed.ist-phosphorus.eu/resource-context/phosphorus

b) Subject attributes

The Subject variable of the AuthZ request contains the following attributes (which are either sent to PEP separately or put into the subjmap):

a) SubjectId (attribute identifier "subject-id")

Subject identifier in RFC822 (email) or X.521 (LDAP or X.509 Public Key Certificate) formats (must be the same as used in the SubjectConfirmationData)

Example: WHO740@users.testbed.ist-phosphorus.eu

SubjectId using X.521 format

b) SubjectConfirmatioData (attribute identifier "subject-confdata") – Authentication assertion or token provided by the trusted AuthN service (can be also SAML AuthN Assertion), or crypto-string provided local AuthN service.

c) SubjectRole (attribute identifier "subject-role") – subject role, currently supporting a single value.

Example: admin, or researcher@project01, or admin@viola.testbed.ist-phosphorus.eu

Future GAAA-TK release will support coma-separated list of roles, SAML Attribute assertion and VOMS Attribute Certificate

d) SubjectContext (attribute identifier “subject-context”) – this attribute is used for providing additional information about a user (and a resource) association like VO, project, experiment/job.

Example: demo001; or VO-Phosphorus

Potentially this attribute can be extended to provide instant reservation context for dynamically configured AuthZ service.

c) Action attributes

The Action element contains single attribute/value that defines requested action in a form of simple string attribute or using fully qualified name for some enumerated action ID’s like proposed in the XACML-NRP profile.

Note. It is important to note that corresponding attributes in the AuthZ request and in the policy must use the same attribute names/ID’s and format.

3.5 Policy identification and policy resolution

3.5.1 General suggestions

When evaluating AuthZ request, the ContextHandler or PDP need to find/select an applicable policy. This is typically done based on the request parameters such as Resource or Subject attributes.

The policy selecting/finding comprises of two steps: policy resolution and policy retrieval. Policy resolution means extracting such information from the AuthZ request that can be used for further policy selection in the storage/repository. Based on this information, a repository request or query can be constructed to retrieve necessary policy.

Note: it is a SunXACML implementation convention that only one Policy or PolicySet should be supplied to PDP for evaluation, and only one component Policy must be selected if using PolicySet.

The following components of the XACML-NRP profile can be used for policy resolution:

- resource ID and resource attributes;

- subject attributes defining the context in which the request should be evaluated, e.g. project or VO (this information is typically a part of the subject attributes);
- attributes and policy profile namespace, which can actually be a part of the resource ID if expressed in Fully Qualified Attribute Name format (FQAN format).

Depending on the policy storage/repository implementation, the following components can be used for policy identification:

- policy file name and directory, if policy is stored as a file;
- PolicyId attribute of the PolicySet or Policy element;
- policy Target element that can include any of Subject, Resource, Action, Environment elements.

Although using basically different ways of storing policies, the first case and second identification methods can be based on similar approach to composing PolicyId attribute and (defining) policy file location path. When using third option, the policy repository should be capable to query policy database by the policy Target content.

3.5.2 Policy resolution convention in the GAAA-TK library

Two components are used for policy resolution in GAAA-TK library profile for Network Resource Provisioning (GAAA-NRP):

- ResourceId expressed in the URL-style URI format, actually specifying the resource FQAN;
- Subject context (SubjectCtx) that specifies subject (and resource) association (e.g., VO, experiment, or project)

General ResourceId expression format:

```
<<ns-type>><Realm>/<ns-domain>/(<device-type> | <resource-context>)/<variables-name-value-pairs>
```

Examples of the URL style:

```
http://testbed.ist-phosphorus.eu/viola/nsp/source=10.1.1.3/target=10.3.1.3
http://testbed.ist-phosphorus.eu/resource-type/harmony
http://testbed.ist-phosphorus.eu/resource-context/phosphorus
```

where

```
"http://" - URL style namespace identifier;
"testbed.ist-phosphorus.eu" – namespace realm;
"viola", "resource-type", "resource-context" – namespace domain;
"nsp", "harmony", "phosphorus" – device type or resource context;
"source", "target" – device variable presented in a form of "name-value".
```

Note: quotation marks are not allowed in URI string.

The same identifier strings expressed in URN style will use URN specific prefix "x-urn:authz-interop:nrp:resource-id" (or its shorter option "x-urn:nrp:resource-id"):

```
x-urn:authz-interop:nrp:resource-id:testbed.ist-
phosphorus.eu:viola:nsp:source=10.1.1.3;target=10.3.1.3
x-urn:authz-interop:nrp:resource-id:testbed.ist-phosphorus.eu:resource-type:harmony
x-urn:authz-interop:nrp:resource-id:testbed.ist-phosphorus.eu:resource-context:phosphorus
```

SubjectCtx can be expressed either in a FQAN format or just contain a simple name without namespace prefixes, e.g. Demo001, or PhosphorusVO, EGEE-VO, etc.

Given above examples the AuthZ Request containing ResourceId and SubjectCtx attributes will be resolved into the following policy path/files:

```
<<root-dir>>/policy/nrp/testbed.ist-phosphorus.eu/viola-policy-nsp-demo001.xml
<<root-dir>>/policy/nrp/testbed.ist-phosphorus.eu/harmony-policy-demo001.xml
<<root-dir>>/policy/nrp/testbed.ist-phosphorus.eu/phosphorus-policy-demo001.xml
```

3.5.3 Policy identification

It is suggested that the PolicyId or PolicySetId is created in the same way using typical for URL/URN style conventions:

```
PolicyId = <<url-namespace-prefix/>>testbed.ist-phosphorus.eu/viola/harmony/demo001/policy
PolicyId = <<urn-namespace-prefix:>>testbed.ist-phosphorus.eu:viola:harmony:demo001:policy
```

where

`<<namespace-prefix>>` - can be dropped;
`namespace-prefix = http://authz-interop.org/nrp/xacml`
`or namespace-prefix = x-urn:authz-interop.org:nrp:xacml`

Example PolicyId expression:

URL style:

```
PolicyId = http://authz-interop.org/nrp/xacml/testbed.ist-
phosphorus.eu/phosphorus/demo001/policy
PolicyId = http://testbed.ist-phosphorus.eu/viola/harmony/demo001/policy
PolicyId = http://testbed.ist-phosphorus.eu/phosphorus/demo001/policy
```

URN style:

```
PolicyId = x-urn:authz-interop.org:nrp:testbed.ist-
phosphorus.eu:viola:harmony:demo001:policy
PolicyId = x-urn:authz-interop.org:nrp:testbed.ist-phosphorus.eu:phosphorus:demo001:policy
```

3.6 Test classes and supporting tools

3.6.1 GAAAPI test classes

The GAAA-TK library is provided with a set of test classes that allows interactively checking all functional components of the library. These test classes provide also programming examples how to call all public PEP and TVS methods and how to use other components of the library.

The following classes are recommended to understand how to use and program GAAA-TK (names are self-descriptive):

org.aaaarch.gaaapi.test.TestGAAAPI.java
org.aaaarch.gaaapi.test.TestXACMLPDP.java
org.aaaarch.gaaapi.test.TestTVS.java
org.aaaarch.gaaapi.test.TestAuthzTicket.java
org.aaaarch.gaaapi.test.TestAuthnTicket.java
org.aaaarch.gaaapi.test.TestUC6Security.java

3.6.2 Simple XACML policy generation tools

GAAA-TK library provides simple XACML policy generation tool that allows to automatically generate XACML policy for few predefined logical models. Currently XACMLPolicyMaker class provided as a part of the org.aaaarch.policy.utils package supports two basic policy models and their combination: RBAC policy model and policy controlling TNA range. More complex policies will require manual policy writing e.g. using any text or XML editor such as XMLSpy.

It is suggested that when more policy enforcement use cases will be defined, new policy models will be included into the XACMLPolicyMaker options. Further XACMLPolicyMaker development will allow reading and editing policies in a simple way. However it is not a scope of current GAAA-TK development to create a full policy editor.

4 GAAA-TK library Installation and configuration

This section provides basic information about configuration parameters and how the GAAA-TK library can be installed and integrated into the main application that needs to be protected by the AuthZ service.

4.1 Configuration

4.1.1 Directories styructure

GAAAPI/TVS installation requires configuration of a few folders that contain a keystorage or used as a temporal directories when processing AuthZ session credentials.

The following directories are used in current implementation and can be configured via the ConfigSecurity.java class (currently hard coded):

```
LOCAL_DIR_ROOT = "" - GAAAPI installation directory

LOCAL_DIR_SECURITYCONFIG = LOCAL_DIR_ROOT + "data/config/";
LOCAL_DIR_KEYSTORE = LOCAL_DIR_ROOT + "etc/security/keystore/";
LOCAL_DIR_KEYSTORE_TRUSTED = LOCAL_DIR_KEYSTORE + "trusted/";
LOCAL_DIR_SYMKEYSTORE = LOCAL_DIR_KEYSTORE + "cnlsec/symkeystore/";
LOCAL_DIR_KEYSTORE_IBC = LOCAL_DIR_KEYSTORE + "ibc/";
LOCAL_DIR_POLICY = LOCAL_DIR_ROOT + "data/policy/";
LOCAL_DIR_SCHEMAS = LOCAL_DIR_ROOT + "data/schemas/";
LOCAL_DIR_AAADATA_CACHE_AZTICKETS = LOCAL_DIR_ROOT + "_aaadata/cache/aztickets/";
LOCAL_DIR_AAADATA_TMP = LOCAL_DIR_ROOT + "_aaadata/tmp/";
```

Note. Provided GAAA-TK package contains all necessary directories structure and also crypto keys. TVS shared secret is hard coded into the token building classes.

```
<installation-root>
+-- data
|   +-- config
```

```
|       +-- gaaapi-nrp-config001.xml
|       +-- xacml1.2-config.xml
|   +-- docs
|   +-- policy
|       +-- nrp
|           +-- testbed.ist-phosphorus.eu
|   +-- schemas
|   +-- sql
+-- +-- wsdl
+-- etc
|   +-- security
|       +-- keystore
|           +-- ibc
|           +-- trusted
|           +-- xmlsec
|           +-- unicore6
|           +-- symkeystore
+-- gaaa-bin
|   +-- gaaapi-nrp-v0*-release-date*.jar
+-- gaaa-lib
|   +-- endorsed
|   +-- lib-ibc
+-- x-output
+-- _aaadata
    +-- cache
    |   +-- aztickets
    |   +-- sessions
            +-- tvs-table-simple.xml
    +-- tmp
```

where gaaapi-nrp-v0*-release-date*.jar is the GAAA-TK library of the recent release (should be checked at the WP4 wikipage http://www.ist-phosphorus.eu/wiki/index.php/Pluggable_GAAA-TK_library).

4.1.2 Configuring domain related information with gaaapi-nrp-config001.xml file

The GAAA-TK configuration facility allows configuring domain specific information in the gaaapi-nrp-config001.xml file. Listing below provides example of such configuration that allows to specify: local domain and neighbour domains, domain's related public key information (which is treated as trusted), identifiers for domain related services AAAServer, TVS, AARR, and other information related to profile, namespace and other type of metadata (see example below). It is considered that this information will be extended with directory configuration information in the next release of the library.

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
<configuration xmlns="http://aaaarch.org/schema/config-gaaapi-0.1"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://aaaarch.org/schema/config-gaaapi-0.1 ../schemas/gaaapi-config-0.1.xsd">
-->
<Configuration>
  <Domains>
    <Domain domaintype="neighbour" domainId="http://testbed.ist-phosphorus.eu/phosphorus">
      <KeyInfo keytype="public">http://testbed.ist-
phosphorus.eu/phosphorus/_public_key_/a8b7573ff8a820fe31b9a67858d7ad37a756818c5756fca04a7f4e9334f92
      </KeyInfo>
      <Service servicetype="AAAServer" serviceId="http://testbed.ist-phosphorus.eu/phosphorus/aaa"/>
      <Service servicetype="TVS" serviceId="http://testbed.ist-phosphorus.eu/phosphorus/aaa/TVS"/>
      <Service servicetype="AARR" serviceId="http://testbed.ist-phosphorus.eu/phosphorus/AARR"/>
    </Domain>
    <Domain domaintype="neighbour" domainId="http://testbed.ist-phosphorus.eu/i2cat">
```

```
      <KeyInfo keytype="public">http://testbed.ist-
phosphorus.eu/phosphorus/_public_key_/a8b7573ff8a820fe31b9a67858d7ad37a756855756fca04a7536f4e9334f92
      </KeyInfo>
      <Service servicetype="AAAServer" serviceId="http://testbed.ist-phosphorus.eu/i2cat/aaa"/>
      <Service servicetype="TVS" serviceId="http://testbed.ist-phosphorus.eu/i2cat/aaa/TVS"/>
      <Service servicetype="AARR" serviceId="http://testbed.ist-phosphorus.eu/i2cat/AARR"/>
    </Domain>
    <Domain domaintype="local" domainId="http://testbed.ist-phosphorus.eu/viola">
      <KeyInfo keytype="public">http://testbed.ist-
phosphorus.eu/phosphorus/_public_key_/a8b7573ff8a820fe31b9a67858d7ad37a756818c5756fca07536f4e9334f92
      </KeyInfo>
      <Service servicetype="AAAServer" serviceId="http://testbed.ist-phosphorus.eu/viola/aaa"/>
      <Service servicetype="TVS" serviceId="http://testbed.ist-phosphorus.eu/viola/aaa/TVS"/>
      <Service servicetype="AARR" serviceId="http://testbed.ist-phosphorus.eu/viola/AARR"/>
    </Domain>
    <Domain domaintype="network" domainId="http://testbed.ist-phosphorus.eu/internet2">
    </Domain>
    <Domain domaintype="application" domainId="http://testbed.ist-phosphorus.eu/viola/demo01">
    </Domain>
    <Domain domaintype="resource">
    </Domain>
  </Domains>
  <Directories>
    <KeyStore keystoretype="trusted">
    </KeyStore>
    <PolicyDirectory policytype="xacml">
    </PolicyDirectory>
  </Directories>
  <Profiles>
    <Profile profiletype="gaaapi" profileId="x-urn:gaaapi:pep-pdp"/>
    <Profile profiletype="attribute" profileId="x-urn:gaaapi:pep-pdp"/>
  </Profiles>
  <ConfigurationData>
    <DeviceConfig devicetype="PEP">
      <ConfigParam name="profileId">x-urn:gaaapi:pep-pdp</ConfigParam>
      <ConfigParam name="sescredtype">azticket</ConfigParam>
    </DeviceConfig>
    <DeviceConfig devicetype="TVS">
      <ConfigParam name="notbefore">0</ConfigParam>
      <ConfigParam name="validtime">86400</ConfigParam>
      <ConfigParam name="validtime-pilot">3600</ConfigParam>
    </DeviceConfig>
  </ConfigurationData>
</Configuration>
```

4.2 Installation

Current GAAA-TK library requires manual installation.

The installation package consists of the 3 archives:

gaaa-tk-lib-external-libraries.zip – all required libraries including GAAA-TK library itself.
gaaa-tk-lib-directories.zip – all necessary supporting directories
gaaa-tk-lib-test-classes.zip – test classes that contains examples how to call the library functions.

Installation procedure is simple. To install GAAA-TK library, you need to unpack provided archives into the selected <root-directory> from which the GAAA-TK functions will be run.

To run GAAA-TK library functions, use programming examples described in section 4.

4.3 Required external libraries

The list of currently used libraries to support core GAAAPI and TVS functionality:

```
bcprov-jdk15-140.jar
commons-codec-1.3.jar
commons-logging-1.0.3.jar
commons-logging-api.jar
dom3-xercesImpl-2.5.0.jar
dom3-xml-apis-2.5.0.jar
jaxrpc-1_1-fr-spec-api.jar
jaxrpc-sec.jar
joda-time-1.4.jar
junit-3.8.1.jar
log4j-1.2.12.jar
opensaml-2.2.0.jar
openws-1.2.2.jar
saaj-api.jar
saaj-impl.jar
soapprocessor.jar
slf4j-api-1.5.5.jar
slf4j-log4j12-1.5.5.jar
xmldsig.jar
sunxacml-cvs1.6.jar
sunxacml-support-cvs1.6.jar
sunxacml-test-cvs1.6.jar
xmlsec-1.4.1.jar
xmlsecSamples.jar
xalan-2.6.jar
xercesImpl.jar
xmltooling-1.2.0.jar
```

The following libraries must be placed into “endorsed” directory:

```
endorsed/resolver-2.9.1.jar
endorsed/serializer-2.9.1.jar
endorsed/xalan-2.7.1.jar
endorsed/xercesImpl-2.9.1.jar
endorsed/xercesSamples.jar
endorsed/xml-apis-2.9.1.jar
ext-unicore6-assertion-utils01.jar
```

The following libraries are required to support use of Unicore6 SAML assertions and Unicore6 Security Framework:

```
lib-unicore/SAMLtypes-1.1.jar"/>
lib-unicore/xbean.jar
lib-unicore/axiom.jar
lib-unicore/axis2-kernel-1.4.1.jar
lib-unicore/wss4j-1.5.5.jar
lib-unicore/ext-unicore6-assertion-utils01.jar – contains necessary classes from
                the Unicore Security Framework library
```

Updated GAAA Toolkit library for ONRP

The following libraries are required to support use of Identity Based Cryptography for token key distribution in multi-domain environment:

```
lib-ibc/IdentityBasedEncryptionJCA.1.0.38.jar
lib-ibc/library/jakarta-regexp-1.4.jar
lib-ibc/library/bcel-head.jar
lib-ibc/library/FieldTracker.jar
lib-ibc/library/artima/suiterunner-1.0beta6.jar
lib-ibc/library/nuimcscg/tender-dev.jar
lib-ibc/library/nuimcscg/ArtimaSuiteRunnerAntTask.1.1.3.jar
lib-ibc/library/nuimcscg/blitz-dev.jar
lib-ibc/library/nuimcscg/fault-dev.jar
```

Full set of libraries is provided next to the GAAA-TK jar-file at the WP4 wikipage http://www.ist-phosphorus.eu/wiki/index.php/Pluggable_GAAA-TK_library

5 Extending GAAA-TK library

This section provides information for developers on how the library can be extended with new functionalities and new supported attributes and credential types. Currently the GAAA-TK library uses XACML policy language and two types of attribute identifiers' expression format: common SAML2-XACML URN-style identifiers' format and URL style format using "http://authz-interop.org" namespace.

5.1 Extending supported attribute profiles

The following classes define the supported attribute ID's and types that include common set of attributes to support library configuration and messaging and related to special attribute profiles, currently XACML-NRP and XACML-Grid:

```
org.aaaarch.config.ConstantsXACMLprofileNRP
org.aaaarch.config.ConstantsXACMLprofileGrid
org.aaaarch.config.ConstantsNS
```

It is suggested that a new attribute profile can be added by adding new Constants* class. A possibility to manage attributes' identifiers as an external metadata file will be considered in future library development.

5.2 Extending supported authentication credentials and attributes types

As described in section 3.2.3 the GAAA-TK supports the following AuthN assertion types

Authorisation ticket ("AAA:AuthzTicket")
Authorisation token ("AAA:AuthzToken")
Authentication ticket ("AAA:AuthnTicket")
Authentication token ("AAA:AuthnToken")
Unicore6 SAML Assertion ("urn:Assertion")

Native SAML2 Assertions (“saml:Assertion")

New types of assertions can be added by extending AuthenticateSubject class and providing necessary helper classes organised as org.aaaarch.impl.{new_supported_type} package (like it is done for Unicore6 assertions).

6 Conclusion

The deliverable describes the functionality and implementation of the GAAA Toolkit pluggable Java library, which provides a reference implementation of the proposed Generic AAA Authorisation framework for Network Resource Provisioning (GAAA-NRP).

The GAAA-TK Java library is designed in a such way that it could support the major Phosphorus testbed use cases and can be used at all networking layers: Data plane, Control Plane, Service plane, and can be also integrated with applications. The proposed architecture should allow a smooth integration with other authorisation frameworks as currently used and developed by NRENs and Grid community.

The deliverable describes a set of APIs used to call the main GAAA-TK services. The Authorisation service can be requested via the Policy Enforcement Point (PEP) or directly from the XACML Policy Decision Point (PDP). The TVS API provides rich functionality for handling tokens used for access control and signalling.

The current report also provides guidelines for library setup and configuration. A separate section provides information for potential contributors and future developers on how to add new attribute profiles and extend supported authentication and attribute credential types.

After the end of the Phosphorus project, the GAAA-TK library will be available as an Open Source software under the Apache license.

7 References

[1] 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

[2] 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. - ftp://ftp.isi.edu/in-notes/rfc2904.txt

[3] Demchenko Y, A. Wan, M. Cristea, C. de Laat, "Authorisation Infrastructure for On-Demand Network Resource Provisioning", The 9th IEEE/ACM International Conference on Grid Computing (Grid 2008), Tsukuba, Japan, Sept. 29 - Oct. 1, 2008. Accepted paper.

[4] Gommans, L., L. Xu ,Y. Demchenko, A. Wan, M. Cristea, R. Meijer, C. de Laat, "Multi-domain Lightpath Authorization using Tokens", Future Generation Computer Systems, Special issue on OptiPuter. Accepted paper.

[5] Web Services Agreement Specification (WS-Agreement). [Online]. http://www.ogf.org/documents/GFD.107.pdf

[6] Yuanming, C., W. Wendong, G. Xiangyang, Q. Xirong, "Initiator-Domain-Based SLA Negotiation for Inter-domain QoS-Service Provisioning", Proc. 4th Int. Networking and Services, 2008, 16-21 March 2008. Pp. 165 - 169.

[7] 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

[8] 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

[9] SAML 2.0 Profile of XACML 2.0, Version 2. Working Draft 2, 26 June 2006. [Online]. Available: http://docs.oasis-open.org/xacml/2.0/xacml-2.0-profile-saml2.0-v2.zip

[10] TBN "ForCES Token Based Switch Design and Implementation", Phosphorus Project Deliverable D4.3.2. – September 30, 2008. [Online]. Available: http://www.ist-phosphorus.eu/files/ deliverables/Phosphorus-deliverable-D4.3.2.pdf

[11] A. Menezes, P. van Oorschot, S. Vanstone, “Handbook of Applied Cryptography”. ISBN: 0-8493-8523-7, October 1996.

[12] A. Shamir. “Identity-based cryptosystems and signature schemes In G.R. Blakley and D. Chaum, editors, Advances in Cryptology”. In Proceedings of CRYPTO'84 on Advances in cryptology. Springer-Verlag LNCS 196, 1985.

[13] H. Tanaka. “A realization scheme for the identity-based cryptosystem”. In Proceedings of CRYPTO'87 Advances in Cryptology. Springer-Verlag LNCS 293, 1988.

[14] XACML Attribute and Obligation Profile for Authorization Interoperability in Grids. [Online] Available https://edms.cern.ch/document/929867/1

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

Appendix A Acronyms

AAA Authentication, Authorisation, Accounting
AAI Authentication, Authorization Infrastructure
AuthZ Authorization
AuthN Authentication
CRP Complex Resource Provisioning
DCAS Domain Central Authorisation Service
GAAA-AuthZ Generic AAA Authorisation Framework
GAAAPI Generic Authentication/Authorisation Application Programming Interface
GEANT2 Pan-European Gigabit Research Network
gLite EGEE Grid middleware
GMPLS Generalized MPLS (MultiProtocol Label Switching)
IdP Identity Provider
NREN National Research and Education Network
NRP Network Resource Provisioning
OLPP Optical LightPath Provisioning
NRPS Network Resource Provisioning System
OHRM Obligation Handling Reference Model
PAP Policy Authority Point
PDP Policy Decision Point
PEP Policy Enforcement Point
PIP Policy Information Point
PKC X.509 Public Key Certificate
PKI Public Key Infrastructure
SAAS Shibboleth Attribute Authority Service
SAML Security Assertion Markup Language
SCAS Site Central Authorisation Service
S-R-A (-E) Subject – Resource – Action (- Environment) in relation to the XACML policy and context definition
SSO Single Sign-On
TBN Token Based Networking
TBS Token Based Switch
TB Token Builder
TVS Token Validation Service
VO Virtual Organisation
VOMS Virtual Organization Membership Service
UNICORE European Grid Middleware (UNIform Access to COmpute REsources)
VLAN Virtual LAN (as specified in IEEE 802.1p)
VIOLA A German project funded by the German Federal Ministry of Education and Research (Vertically Integrated Optical Testbed for Large Applications in DFN)
VPN Virtual Private Network
XACML eXtensible Access Control Markup Language
XML eXtensible Markup Language

Appendix B XACML Policy examples

B.1 Example 1 - XACML policy and corresponding request/response messages

a) Policy example evaluating user roles and network source and target TNA's.

```
<Policy PolicyId="http://testbed.ist-phosphorus.eu/viola/harmony/demo040/policy"
RuleCombiningAlgId="urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:permit-overrides">
  <Description>Permit actions for Phosphorus testbed users with specific roles. Added range (10.3.*,
10.4.*, 10.7.*,  10.8.*)</Description>
  <Target>
    <Subjects>
      <AnySubject/>
    </Subjects>
    <Resources>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">http://testbed.ist-
phosphorus.eu/viola/harmony</AttributeValue>
          <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-
id" DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
        </ResourceMatch>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">10.4.</AttributeValue>
          <ResourceAttributeDesignator AttributeId="http://authz-
interop.org/AAA/xacml/resource/source" DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </ResourceMatch>
      </Resource>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">http://testbed.ist-
phosphorus.eu/viola/harmony</AttributeValue>
          <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-
id" DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
        </ResourceMatch>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">10.3.</AttributeValue>
          <ResourceAttributeDesignator AttributeId="http://authz-
interop.org/AAA/xacml/resource/source" DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </ResourceMatch>
      </Resource>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">http://testbed.ist-
phosphorus.eu/viola/harmony</AttributeValue>
          <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-
id" DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
        </ResourceMatch>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">10.7.</AttributeValue>
          <ResourceAttributeDesignator AttributeId="http://authz-
interop.org/AAA/xacml/resource/source" DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </ResourceMatch>
      </Resource>
      <Resource>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
```

```
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">http://testbed.ist-
phosphorus.eu/viola/harmony</AttributeValue>
          <ResourceAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-
id" DataType="http://www.w3.org/2001/XMLSchema#anyURI"/>
        </ResourceMatch>
        <ResourceMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">10.8.</AttributeValue>
          <ResourceAttributeDesignator AttributeId="http://authz-
interop.org/AAA/xacml/resource/source" DataType="http://www.w3.org/2001/XMLSchema#string"/>
        </ResourceMatch>
      </Resource>
    </Resources>
    <Actions>
      <AnyAction/>
    </Actions>
  </Target>
  <Rule RuleId="http://testbed.ist-phosphorus.eu/viola/harmony/demo001/policy/rule/action-
type/create-path" Effect="Permit">
    <Target>
      <Subjects>
        <AnySubject/>
      </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">create-
path</AttributeValue>
            <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
    <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-at-least-one-member-of">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">admin</AttributeValue>
        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">researcher</AttributeValue>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">professor</AttributeValue>
      </Apply>
      <SubjectAttributeDesignator AttributeId="http://authz-interop.org/AAA/xacml/subject/subject-
role" DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </Condition>
  </Rule>
  <Rule RuleId="http://testbed.ist-phosphorus.eu/viola/harmony/demo001/policy/rule/action-
type/activate-path" Effect="Permit">
    <Target>
      <Subjects>
        <AnySubject/>
      </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">activate-
path</AttributeValue>
            <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
    <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-at-least-one-member-of">
```

```
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">researcher</AttributeValue>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">professor</AttributeValue>
      </Apply>
      <SubjectAttributeDesignator AttributeId="http://authz-interop.org/AAA/xacml/subject/subject-
role" DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </Condition>
  </Rule>
  <Rule RuleId="http://testbed.ist-phosphorus.eu/viola/harmony/demo001/policy/rule/action-
type/cancel" Effect="Permit">
    <Target>
      <Subjects>
        <AnySubject/>
      </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">cancel</AttributeValue>
            <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
    <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-at-least-one-member-of">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">admin</AttributeValue>
      </Apply>
      <SubjectAttributeDesignator AttributeId="http://authz-interop.org/AAA/xacml/subject/subject-
role" DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </Condition>
  </Rule>
  <Rule RuleId="http://testbed.ist-phosphorus.eu/viola/harmony/demo001/policy/rule/action-
type/access" Effect="Permit">
    <Target>
      <Subjects>
        <AnySubject/>
      </Subjects>
      <Resources>
        <AnyResource/>
      </Resources>
      <Actions>
        <Action>
          <ActionMatch MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
            <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">access</AttributeValue>
            <ActionAttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string"/>
          </ActionMatch>
        </Action>
      </Actions>
    </Target>
    <Condition FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-at-least-one-member-of">
      <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">admin</AttributeValue>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">professor</AttributeValue>
        <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">researcher</AttributeValue>
        <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">student</AttributeValue>
      </Apply>
      <SubjectAttributeDesignator AttributeId="http://authz-interop.org/AAA/xacml/subject/subject-
role" DataType="http://www.w3.org/2001/XMLSchema#string"/>
    </Condition>
```

```
  </Rule>
</Policy>
```

b) XACML request message example:

```
<Request>
  <Subject SubjectCategory="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject">
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="http://testbed.ist-
phosphorus.eu/phosphorus/aaa/AttributeIssuer" IssueInstant="2008-07-29T14:34:44.796000000+02:00">
      <AttributeValue>WHO740@users.testbed.ist-phosphorus.eu</AttributeValue>
    </Attribute>
    <Attribute AttributeId="http://authz-interop.org/AAA/xacml/subject/subject-context"
DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="http://testbed.ist-
phosphorus.eu/phosphorus/aaa/AttributeIssuer" IssueInstant="2008-07-29T14:34:44.796000000+02:00">
      <AttributeValue>demo041</AttributeValue>
    </Attribute>
    <Attribute AttributeId="http://authz-interop.org/AAA/xacml/subject/subject-confdata"
DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="http://testbed.ist-
phosphorus.eu/phosphorus/aaa/AttributeIssuer" IssueInstant="2008-07-29T14:34:44.796000000+02:00">
      <AttributeValue>aaa:authn:gaaapi:subject:confirmed</AttributeValue>
    </Attribute>
    <Attribute AttributeId="http://authz-interop.org/AAA/xacml/subject/subject-role"
DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="http://testbed.ist-
phosphorus.eu/phosphorus/aaa/AttributeIssuer" IssueInstant="2008-07-29T14:34:44.796000000+02:00">
      <AttributeValue>researcher</AttributeValue>
    </Attribute>
  </Subject>
  <Resource>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
DataType="http://www.w3.org/2001/XMLSchema#anyURI" Issuer="http://testbed.ist-
phosphorus.eu/phosphorus/aaa/AttributeIssuer" IssueInstant="2008-07-29T14:34:44.765000000+02:00">
      <AttributeValue>http://testbed.ist-phosphorus.eu/viola/harmony</AttributeValue>
    </Attribute>
  </Resource>
  <Action>
    <Attribute AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
DataType="http://www.w3.org/2001/XMLSchema#string" Issuer="http://testbed.ist-
phosphorus.eu/phosphorus/aaa/AttributeIssuer" IssueInstant="2008-07-29T14:34:44.765000000+02:00">
      <AttributeValue>create-path</AttributeValue>
    </Attribute>
  </Action>
</Request>
```

c) Example Response message with returned decision “Permit”

```
<Response>
  <Result ResourceId="http://testbed.ist-phosphorus.eu/viola/harmony">
    <Decision>Permit</Decision>
    <Status>
      <StatusCode Value="urn:oasis:names:tc:xacml:1.0:status:ok"/>
    </Status>
  </Result>
</Response>
```

Appendix C Authorisation ticket and token examples

C.1 TVS XML Token format and examples

Refer to the token data model and token types definition in section 2.2

a) Access token (type 0)

XML token format uses a special “TVS/TBN” profile of the more general AuthzToken format. Example of the full TVS XML token is shown below:

```
<AAA:AuthzToken xmlns:AAA="http://www.aaauthreach.org/ns/#AAA"
       Issuer="urn:aaa:gaaapi:token:TVS"
       SessionId="a9bcf23e70dc0a0cd992bd24e37404c9e1709afb"
       TokenId="d1384ab54bd464d95549ee65cb172eb7">
    <AAA:TokenValue>ebd93120d4337bc3b959b2053e25ca5271a1c17e</AAA:TokenValue>
    <AAA:Conditions NotBefore="2007-08-12T16:00:29.593Z"
       NotOnOrAfter="2007-08-13T16:00:29.593Z"/>
</AAA:AuthzToken>
```

where the element `<TokenValue>` and attributes `SessionId` and `TokenId` are mandatory, and the element `<Conditions>` and attributes `Issuer, NotBefore, NotOnOrAfter` are optional.

Minimum token format is illustrated in the following example:

```
<AAA:AuthzToken xmlns:AAA="http://www.aaauthreach.org/ns/#AAA"
      SessionId="a9bcf23e70dc0a0cd992bd24e37404c9e1709afb"
      TokenId="d1384ab54bd464d95549ee65cb172eb7">
    <AAA:TokenValue>ebd93120d4337bc3b959b2053e25ca5271a1c17e</AAA:TokenValue>
</AAA:AuthzToken>
```

Attributes “Issuer” allow for distinguishing different AuthzToken profiles. The TVS profile is identified by the URN "urn:aaa:gaaapi:token:TVS".

b) Pilot token type 1

The pilot token type 1 is used just as a container for communicating GRI during the reservation stage. It contains mandatory SessionId attribute and optional Condition element (it doesn’t contain TokenValue element).

```
<AAA:AuthzToken xmlns:AAA="http://www.aaauthreach.org/ns/AAA"
```

```
     Issuer="http://testbed.ist-phosphorus.eu/phosphorus/aaa/TVS"
     SessionId="7172533c4e83ae7f19c13e015e07e244bb986dee"
     TokenId="cc99a687df8ef6aeb661e6579f8f209b">
</AAA:AuthzToken>
```

c) Pilot token type 2

The pilot token type 2 is the origin/requestor authenticating token. Its TokenValue element contains a value that can be used as the authentication value for the token origin. The token value is calculated of GRI by applying e.g. HMAC function to the GRI together with the requestor symmetric secret or private key.

```
<AAA:AuthzToken xmlns:AAA="http://www.aaauthreach.org/ns/AAA"
        Issuer="http://testbed.ist-phosphorus.eu/phosphorus/aaa/TVS/token-pilot"
        SessionId="7f841d4ec3e802fd7852a8db35906abaff53f79a"
        TokenId="b5e4f3386bdfeb7c9ff9f76e67d30957" type="pilot-type2">
   <AAA:TokenValue>2e31717173e63002360294a9175388c3138299f0</AAA:TokenValue>
   <AAA:Conditions NotBefore="2008-07-19T22:59:40.281Z"
        NotOnOrAfter="2008-07-20T22:59:40.281Z"/>
</AAA:AuthzToken>
```

d) Pilot token type 3

The pilot token type 3 extends the type2 with Domains element that allows to collect domains security context information (in the Domains/Domain element) when passing multiple domains during the reservation process. Such information includes the previous token and the domain's trust anchor or public key.

```
<AAA:AuthzToken xmlns:AAA="http://www.aaauthreach.org/ns/AAA"
       Issuer="http://testbed.ist-phosphorus.eu/phosphorus/aaa/TVS/token-pilot"
       SessionId="0912182e7f9c7d156028e77e3d6b460de8e4937c"
       TokenId="a99b91e70307bdd329c9a0aec18bb4a3" type="pilot-type3">
  <AAA:TokenValue>3923c7ecb979e7078ab8745191a7b25348cdcb48</AAA:TokenValue>
  <AAA:Conditions NotBefore="2008-07-25T09:38:39.890Z"
       NotOnOrAfter="2008-07-26T09:38:39.890Z"/>
  <AAA:DomainsContext>
    <AAA:Domain domainId="http://testbed.ist-phosphorus.eu/viola">
      <AAA:AuthzToken Issuer="http://testbed.ist-phosphorus.eu/viola/aaa/TVS/token-pilot"
             SessionId="b0b6202d7bd7fb7b591b5de29950d21fdb8bf375"
             TokenId="e7c88fad8cff42d7faaa961b96411ae6">
        <AAA:TokenValue>f09194bbddeef95bc4acb187f71b0bb20b2d0b44</AAA:TokenValue>
        <AAA:Conditions NotBefore="2008-07-18T21:55:15.296Z"
               NotOnOrAfter="2008-07-18T21:55:15.296Z"/>
      </AAA:AuthzToken>
      <AAA:KeyInfo>http://testbed.ist-phosphorus.eu/viola/_public_key_</AAA:KeyInfo>
    </AAA:Domain>
  </AAA:DomainsContext>
</AAA:AuthzToken>
```

C.2 AuthzTicket example – Proprietary Format

GAAAPI supports AuthZ tickets (and additionally AuthN tickets) generation in a proprietary XML format and by using the SAML assertion format. AuthZ ticket format is discussed in section 2.3.

The listing below provides an example of the AuthzTicket containing also AuthZ session data elements and Obligations.

```
<AAA:AuthzTicket xmlns:AAA="http://www.aaauthreach.org/ns/#AAA" Issuer="urn:cnl:trust:tickauth:pep"
<AAA:AuthzTicket xmlns:AAA="http://www.aaauthreach.org/ns/AAA" Issuer="x-urn:aaa:trust:tickauth:pep"
PolicyRef="nsp-policy-demo001" SessionId="Demo001-2008-06-16"
TicketId="bf35c450951183ba27525a1975d21bc5">
  <AAA:Decision ResourceId="http://testbed.ist-phosphorus.eu/resource-type/nsp">Permit</AAA:Decision>
  <AAA:Validity NotBefore="2008-06-17T15:14:50.750Z" NotOnOrAfter="2008-06-18T15:14:50.750Z"/>
  <AAA:Subject Id="subject">
    <AAA:SubjectId>WHO740@users.testbed.ist-phosphorus.eu</AAA:SubjectId>

<AAA:SubjectConfirmationData>IGhA11vwa8bUktYhuU9que+d4XLUvJFHrtDC/OE3Ui1bxtmuCxLldw==</AAA:SubjectCon
firmationData>
    <AAA:SubjectAttribute AttributeId="subject-role">researcher</AAA:SubjectAttribute>
    <AAA:SubjectContext>demo001</AAA:SubjectContext>
  </AAA:Subject>
  <AAA:Resource>http://testbed.ist-phosphorus.eu/resource-type/nsp</AAA:Resource>
  <AAA:Actions>
    <AAA:Action>cnl:actions:CtrlInstr</AAA:Action>
    <AAA:Action>cnl:actions:CtrlExper</AAA:Action>
  </AAA:Actions>
  <AAA:Delegation MaxDelegationDepth="3" restriction="subjects">
    <AAA:DelegationSubjects>
      <AAA:SubjectID>team-member-2</AAA:SubjectID>
      <AAA:SubjectID>team-member-1</AAA:SubjectID>
    </AAA:DelegationSubjects>
  </AAA:Delegation>
  <AAA:Conditions NotBefore="2006-06-08T12:59:29.912Z"
                       NotOnOrAfter="2006-06-09T12:59:29.912Z" renewal="no">
    <AAA:ConditionAuthzSession PolicyRef="PolicyRef-GAAA-RBAC-test001" SessionID="JobXPS1-2006-001">
      <AAA:SessionData>put-session-data-Ctx-here</AAA:SessionData>
    </AAA:ConditionAuthzSession>
  </AAA:Conditions>
  <AAA:Obligations>
    <AAA:Obligation>put-policy-obligation(2)-here</AAA:Obligation>
    <AAA:Obligation>put-policy-obligation(1)-here</AAA:Obligation>
  </AAA:Obligations>
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
      <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
      <ds:Reference URI="">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
          <ds:Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments"/>
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <ds:DigestValue>opRdTeFA8xcxg+fdqIPIPdpA+50=</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue>
CtmfIVg/CDVnHBgYuPwtFcWDOdkSEGE/OINp4My0wDEPa8DWR05yg5kh96E1Hr5s6DB7N3jc8fjk
4EBMZYFj7sQaLmPPb+YvJyDojPTIVqtwt2CQWA4WXDeNBLVD7qL+o4fdxCW/y/VKoM0IjdbZ3Pu5
4DSFefShsYIZnaJto2w=
</ds:SignatureValue>
  </ds:Signature>
</AAA:AuthzTicket>
```

Appendix D TVSTable example

Refer to the token data model and token types definition in section 2.2

```
<TVSTable DomainLocal="http://testbed.ist-phosphorus.eu/viola">
  <TVSEntry DomainId="http://testbed.ist-phosphorus.eu/viola">
    <SessionContext SessionId="186c435871bb50df6ab69d2e244f856cd7e9d84896b0dbe1792993ae18f9d423">
      <Conditions NotBefore="2009-03-06T12:41:54.687Z" NotOnOrAfter="2009-03-08T12:41:54.687Z"/>
      <Action>create-path</Action>
      <Subject Id="subject">
        <SubjectId>WHO740@users.testbed.ist-phosphorus.eu</SubjectId>
        <SubjectRole>researcher</SubjectRole>
        <SubjectContext>demo041</SubjectContext>
      </Subject>
      <Resource>
        <ResourceId>http://testbed.ist-phosphorus.eu/viola/harmony</ResourceId>
        <ResourceSource>10.3.1.16</ResourceSource>
        <ResourceTarget>10.7.3.13</ResourceTarget>
      </Resource>
      <KeyInfo keytype="public">http://testbed.ist-
phosphorus.eu/viola/_public_key_/186c435871bb50df6ab69d2e244f856cd7e9d84896b0dbe1792993ae18f9d423</Ke
yInfo>
    </SessionContext>
    <SessionContext SessionId="dcca83402c43c00113b124cef55011bbc2f695d7">
      <Conditions NotBefore="2009-03-24T01:25:18.171Z" NotOnOrAfter="2009-03-26T01:25:18.171Z"/>
      <Action>create-path</Action>
      <Subject Id="subject">
        <SubjectId>WHO740@users.testbed.ist-phosphorus.eu</SubjectId>
        <SubjectRole>researcher</SubjectRole>
        <SubjectContext>demo041</SubjectContext>
      </Subject>
      <Resource>
        <ResourceId>http://testbed.ist-phosphorus.eu/viola/harmony</ResourceId>
        <ResourceSource>10.3.1.16</ResourceSource>
        <ResourceTarget>10.7.3.13</ResourceTarget>
      </Resource>
      <KeyInfo keytype="public">http://testbed.ist-
phosphorus.eu/viola/_public_key_/dcca83402c43c00113b124cef55011bbc2f695d7</KeyInfo>
    </SessionContext>
    <SessionContext SessionId="6e7e3578298c88298d91f3d89eff44d1f55be07e">
      <Conditions NotBefore="2009-03-23T17:10:32.890Z" NotOnOrAfter="2009-03-25T17:10:32.890Z"/>
      <Action>create-path</Action>
      <Subject Id="subject">
        <SubjectId>WHO740@users.testbed.ist-phosphorus.eu</SubjectId>
        <SubjectRole>researcher</SubjectRole>
        <SubjectContext>demo041</SubjectContext>
      </Subject>
      <Resource>
        <ResourceId>http://testbed.ist-phosphorus.eu/viola/harmony</ResourceId>
        <ResourceSource>10.3.1.16</ResourceSource>
        <ResourceTarget>10.7.3.13</ResourceTarget>
      </Resource>
      <KeyInfo keytype="public">http://testbed.ist-
phosphorus.eu/viola/_public_key_/6e7e3578298c88298d91f3d89eff44d1f55be07e</KeyInfo>
    </SessionContext>
    <SessionContext SessionId="fe2d1549e8dfbfcb7bd03c306b54a19275a2309e">
      <Conditions NotBefore="2008-11-22T03:06:07.312Z" NotOnOrAfter="2008-11-24T03:06:07.312Z"/>
      <Action>create-path</Action>
      <Subject Id="subject">
        <SubjectId>WHO740@users.testbed.ist-phosphorus.eu</SubjectId>
        <SubjectRole>researcher</SubjectRole>
        <SubjectContext>demo041</SubjectContext>
      </Subject>
      <Resource>
```

```
        <ResourceId>http://testbed.ist-phosphorus.eu/viola/harmony</ResourceId>
        <ResourceSource>10.3.1.16</ResourceSource>
        <ResourceTarget>10.7.3.13</ResourceTarget>
      </Resource>
      <KeyInfo keytype="public">http://testbed.ist-
phosphorus.eu/viola/_public_key_/fe2d1549e8dfbfcb7bd03c306b54a19275a2309e</KeyInfo>
    </SessionContext>
    <SessionContext SessionId="b48b92a36f598dcae18ac93862f66ec6e0e7a9e1">
      <Conditions NotBefore="2008-11-21T02:58:04.218Z" NotOnOrAfter="2008-11-23T02:58:04.218Z"/>
      <Action>create-path</Action>
      <Subject Id="subject">
        <SubjectId>WHO740@users.testbed.ist-phosphorus.eu</SubjectId>
        <SubjectRole>researcher</SubjectRole>
        <SubjectContext>demo041</SubjectContext>
      </Subject>
      <Resource>
        <ResourceId>http://testbed.ist-phosphorus.eu/viola/harmony</ResourceId>
        <ResourceSource>10.3.1.16</ResourceSource>
        <ResourceTarget>10.7.3.13</ResourceTarget>
      </Resource>
      <KeyInfo keytype="public">http://testbed.ist-
phosphorus.eu/viola/_public_key_/b48b92a36f598dcae18ac93862f66ec6e0e7a9e1</KeyInfo>
    </SessionContext>
    <SessionContext SessionId="73c7fdaf09e3eec39e7fe7c6b171adbf178ed2c0315c775d29f0420909f44faa">
      <Conditions NotBefore="2009-01-28T18:07:35.859Z" NotOnOrAfter="2009-01-30T18:07:35.859Z"/>
      <Action>create-path</Action>
      <Subject Id="subject">
        <SubjectId>WHO740@users.testbed.ist-phosphorus.eu</SubjectId>
        <SubjectRole>researcher</SubjectRole>
        <SubjectContext>demo041</SubjectContext>
      </Subject>
      <Resource>
        <ResourceId>http://testbed.ist-phosphorus.eu/viola/harmony</ResourceId>
        <ResourceSource>10.3.1.16</ResourceSource>
        <ResourceTarget>10.7.3.13</ResourceTarget>
      </Resource>
      <KeyInfo keytype="public">http://testbed.ist-
phosphorus.eu/viola/_public_key_/73c7fdaf09e3eec39e7fe7c6b171adbf178ed2c0315c775d29f0420909f44faa</Ke
yInfo>
    </SessionContext>
  </TVSEntry>
  <TVSEntry DomainId="domainId.Template">
    <SessionContext SessionId="GRI-sessionId">
      <Conditions NotBefore="2008-07-28T00:08:23.187Z" NotOnOrAfter="2008-07-28T00:08:23.187Z"/>
      <Action/>
      <Subject Id="subject">
        <SubjectId/>
        <SubjectRole/>
        <SubjectContext/>
      </Subject>
      <Resource>
        <ResourceId/>
        <ResourceSource/>
        <ResourceTarget/>
      </Resource>
      <KeyInfo keytype="public">domainId.Template/_public_key_/GRI-sessionId</KeyInfo>
    </SessionContext>
  </TVSEntry>
  <TVSEntry DomainId="http://testbed.ist-phosphorus.eu/phosphorus">
    <SessionContext SessionId="7534244db03370e2f080630657eec996bb4d6fb0">
      <Conditions NotBefore="2008-07-21T10:38:30.703Z" NotOnOrAfter="2008-07-25T11:38:30.703Z"/>
      <Action>create-path</Action>
      <Subject Id="subject">
        <SubjectId>WHO740@users.testbed.ist-phosphorus.eu</SubjectId>
        <SubjectRole>researcher</SubjectRole>
        <SubjectContext>demo001</SubjectContext>
      </Subject>
      <Resource>
```

```
        <ResourceId>http://testbed.ist-phosphorus.eu/phosphorus/aaa</ResourceId>
        <ResourceSource>10.1.1.16</ResourceSource>
        <ResourceTarget>10.7.3.13</ResourceTarget>
      </Resource>
      <KeyInfo keytype="public">http://testbed.ist-phosphorus.eu/phosphorus/_public_key_/nsp-
domain.uob_417d661511b27bdd589ebdd5aef81340eb62f084da7939d481a284c0a32d6b0f</KeyInfo>
    </SessionContext>
    <SessionContext SessionId="nsp-
domain.uob_a771c0f9d16d05bd0fefe934a2bfbbf6192934503d77199ab7a24b74028bfe17">
      <Conditions NotBefore="2008-07-22T11:09:18.703Z" NotOnOrAfter="2008-07-22T12:09:18.703Z"/>
      <Action>create-path</Action>
      <Subject Id="subject">
        <SubjectId>WHO740@users.testbed.ist-phosphorus.eu</SubjectId>
        <SubjectRole>researcher</SubjectRole>
        <SubjectContext>demo001</SubjectContext>
      </Subject>
      <Resource>
        <ResourceId>http://testbed.ist-phosphorus.eu/phosphorus/aaa</ResourceId>
        <ResourceSource>10.1.1.16</ResourceSource>
        <ResourceTarget>10.7.3.13</ResourceTarget>
      </Resource>
      <KeyInfo keytype="public">http://testbed.ist-phosphorus.eu/phosphorus/_public_key_/nsp-
domain.uob_a771c0f9d16d05bd0fefe934a2bfbbf6192934503d77199ab7a24b74028bfe17</KeyInfo>
    </SessionContext>
  </TVSEntry>
</TVSTable>
```