034115

PHOSPHORUS

Lambda User Controlled Infrastructure for European Research

Integrated Project

Strategic objective:
Research Networking Testbeds

Deliverable reference number: D.2.8

Design of the Grid-GMPLS Control Plane to support the Phosphorus Grid AAI

Due date of deliverable: 2008-09-30
Actual submission date: 2008-09-30
Document code: Phosphorus-WP2-D2.8

Start date of project:
October 1, 2006

Duration:
30 Months

Organisation name of lead contractor for this deliverable: **neXtworks (NXW)**

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)	

Abstract

This document describes the integration of the GAAA-TK library released by the WP4 in the G^{2}MPLS control plane released by the WP2, showing the GAAA-NRP token-based model adopted by the control plane for multi-domain authorization and the associated signalling used to carry the security context information along the end-to-end path. This documents describes also the functionalities and the implementation of the AuthZ gateway, the module that enables the interaction between the G^{2}MPLS control plane and the GAAA-NRP infrastructure.

List of Contributors

Giada Landi	NXW
Gino Carrozzo	NXW
Giacomo Bernini	NXW
Nicola Ciulli	NXW

Reviewed and commented on behalf of WP4 by:

Yuri Demchenko	UvA

Table of Contents

List of Figures

List of Tables

0 Executive Summary

This document describes the usage of the GAAA infrastructure by the G^{2}MPLS control plane to authorize the resource reservation and provisioning services.

In section 1 the objectives of the interworking activities between WP2 and WP4 are stated, as well as the scope of the document.

In section 2 the terminology relevant for GAAA/AuthZ infrastructure and G^{2}MPLS architecture is presented by specifying the main source of information.

In section 3 the integration between the G^{2}MPLS architecture and GAAA-AuthZ framework is described, with reference to the adopted AuthZ model and the RSVP signalling extensions to support the end-to-end multi-domain authorization procedure.

Section 5 presents the PEP-GW module, that enables the communication between the G^{2}MPLS control plane modules and the GAAA-TK library, through the Policy Enforcement Point.

Section 6 presents the NCC functionalities to support the authorization mechanisms.

1 Objectives and Scope

Authentication and Authorization is a fundamental service for the secure and trusted use of both Grid and network infrastructures. Access to the Grid computational resources and to the network transport connections need to be regulated and authorized, in order to preserve the final service integrity and the robustness of the various control plane procedures.

The management of network as well as Grid resources in the multi-domain G^{2}MPLS environment is achieved by the integration of the GAAA-TK library released by the WP4 [**PH-WP4-D4.3.1**] and the G^{2}MPLS control plane developed by the WP2 [**PH-WP2-D2.6**]. A Control Plane architectural extension, compliant with the G^{2}MPLS-related standards, has been designed and developed in order to support the authentication and authorization mechanisms offered by the GAAA framework.

This document presents the token-based authorization model supported by the GAAA-NRP infrastructure and adopted by the G^{2}MPLS control plane, highlighting the benefits provided by this solution. Control plane signalling scenarios for resource reservation and authorization procedures are also described, with reference to the RSVP messages extended to enable the transfer of the adopted security context information. Finally, the document describes also the control plane modules enabling the communication with the GAAA-NRP infrastructure, with details about their functionalities, implementation and interaction.

2 Terminology

No specific terminology is introduced in this document, which refers to:

- WP2 deliverables D2.1, D2.2, D2.6, D2.7 and D2.4 for what concerns the G²MPLS architecture and protocols, and implementations of them;
- WP4 deliverable D4.3.1 for what concerns the GAAA/AuthZ framework.

2.1 Abbreviations

A full list of the abbreviations used in this document is provided in Section 9.

3 AuthN/AuthZ model adopted in G²MPLS

The AuthN/AuthZ model adopted in G²MPLS Control Plane is based on the GAAA AuthZ architecture for Optical Network Resource Provisioning (NRP) described in [**PH-WP4-D4.3.1**] hereafter referred to as GAAA-NRP. This framework allows the control plane functionalities related to on-demand and multi domain resource reservation and provisioning to integrate mechanisms for end-to-end token-based authorization procedures.

As depicted in Figure 3-1, the NSP of the Network Resource Provisioning System (NRPS) situated in each specific domain along the path interacts with the AAA infrastructure through the corresponding Policy Enforcement Point (PEP). The AuthZ framework is based on intra and inter-domain modules, allowing the authorization for both global and local resources: reserved resources are bounded to a Global Reservation Identifier (GRI) with end-to-end validity and associated to an internal Local Reservation Identifier (LRI) specific for each domain. At the Inter Domain Controller (IDC), authorization requests are processed by the Policy Decision Point (PDP) that identifies the applicable policies through the Policy Authority Point (PAP) and takes the AuthZ decision according to the current security context information.

The security context information includes the user AuthN credentials for the AuthZ request in the first domain. If authorized, the PEP/PDP creates an AuthZ ticket that can be used as security context for the policy evaluation in the next domains. The AuthZ ticket can be used to transfer an AuthZ decision and a policy enforcement context between different AuthZ/security domains, however to split different components of the whole NRP process the pilot token was introduced to enable inter-domain signalling and authentication and act as container for communicating interdomain context. Pilot token can include either AuthZ ticket when it is necessary to make an AuthZ decision in the next domain or just reference to the stored AuthZ ticket in a form of AuthZ token. The AuthZ token acts as an unambiguous reference for the original ticket and, accordingly, to the session context stored in a particular domain.

GAAA-NRP uses two types of tokens: the pilot token for interdomain signalling used at the reservation stage and the access token used as access credential at the access/usage stage. A special GAAA-NRP service called Token Validation Service (TVS) supports all token management functions. In a simple scenario the TVS has a local scope and checks the intra-domain local reservation table according to the session GRI. Using such approach in multi domain scenarios, each domain maintains the internal association between the GRI and the LRI. In more advanced scenarios, the TVS provides mechanisms for token and token keys distribution between the NSP-AAA domains, bounding the token to the associated GRI by means of shared or dynamically created inter-domain trust infrastructure.

3.1 GAAA/AuthZ usage by G²MPLS

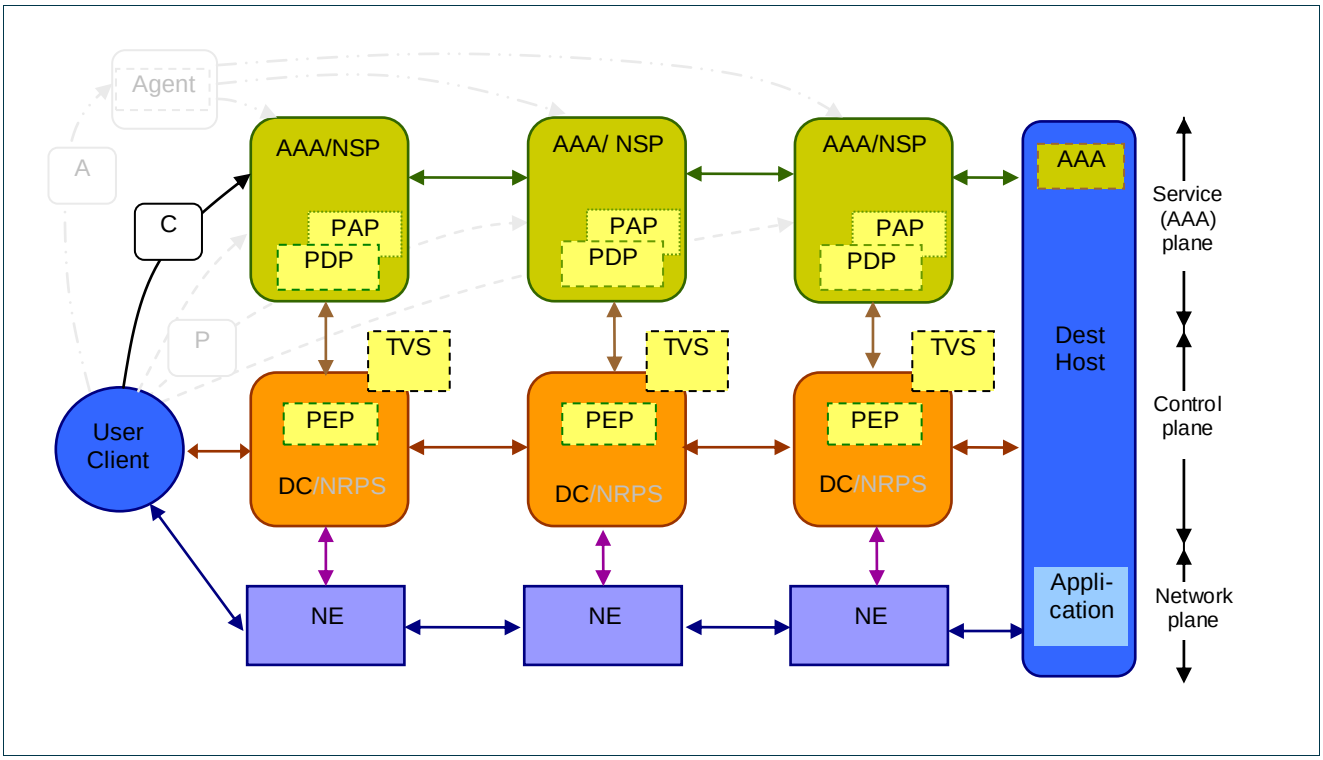

Figure 3-1: Chain reservation sequence.

The G²MPLS control plane adopts the *Chain reservation sequence* model (or provider sequence model - Figure 3-1), where the user contacts only the local network domain and the control plane signalling is in charge to trigger the resource reservation authorization in each consecutive domain. The inter-domain token management is automatically handled by the advanced TVS mechanisms: the distribution of the token and the token keys among the GAAA-NRP infrastructure entities is managed by TVS through the creation of dynamic trust associations. Using this approach, the PEPs located in each domain remain stateless entities that do not store any information about the associations between existing sessions and active tokens, however they can validate AuthZ requests including tokens using the TVS mechanisms.

The TVS dynamic procedures, adopted by the G²MPLS architecture, are enabled through the usage of AuthZ pilot tokens. In particular, the GAAA-NRP framework defines two main types of tokens: access tokens and pilot tokens. While the traditional access tokens include only the *sessionID* (i.e. GRI), the Token Value element and an optional time validity attribute, the more advanced pilot tokens contain a brief description of some context information and can be used to set up a multi-domain TVS infrastructure for distributed token-based access

control. This approach is particularly suitable for the G^{2}MPLS peer model, allowing the control-plane NRPS distributed modules located in each domain to interact with the AAA framework using a single AuthZ token with end to end validity. The complex architecture of the multi-domain distributed GAAA/AuthZ framework is completely transparent for the NRPS and is hidden by the TVS mechanisms associated to the features provided by the pilot token.

3.2 Multi-domain authorization scenario

The G^{2}MPLS framework has been designed to allow the interaction between the NRPS and the GAAA/AuthZ infrastructure at the Network Call Controller (NCC) signalling level. This objective has been achieved placing a specific instance of the GAAA-TK PEP in each client and domain border node, co-located with the corresponding NCC, and enabling the NCC-PEP communication through a PEP-GW, described in section 5. Similarly the end to end call signalling messages have been extended in order to carry the security context information required by the GAAA-AuthZ service for the token based authorization procedures.

Figure 3-2 shows a sample multi-domain scenario where resources need to be reserved in three different domains along the path between the Grid Site A and the Grid Site B. During the session setup, the NCC-1 at the ingress of the first G^{2}MPLS domain interacts with the corresponding PEP through its PEP Gateway (section 5) in order to authorize the resource reservation for the current session. The PEP evaluates the request through the PDP of the GAAA-NRP framework and, if authorized, builds a new pilot token using the mechanisms provided by the TVS. This token is forwarded to the requestor NCC-1 and included in the end-to-end signalling messages.

In particular, the token is transferred between two border NCCs located in different domains through the G.E-NNI RSVP signalling, while G.UNI RSVP signalling is used in the segment between the border NCC of the last domain and the terminating Client Call Controller (CCC-z). The RSVP Policy Object has been extended in order to carry the AuthZ token along the path, as described in section 4. The downstream ingress NCCs and CCC-z receives the token from the related G.E-NNI/G.UNI RSVP instance and use it as a security context information for the AuthZ request to the local PEP. The same approach is used in the deletion phase when resources are released. When the NCCs belong to the same domain, the token is conveyed by the G² Call intra-domain signalling (XML messages over UDP, as specified in D2.3), with a proper extension (see Section 4 for details).

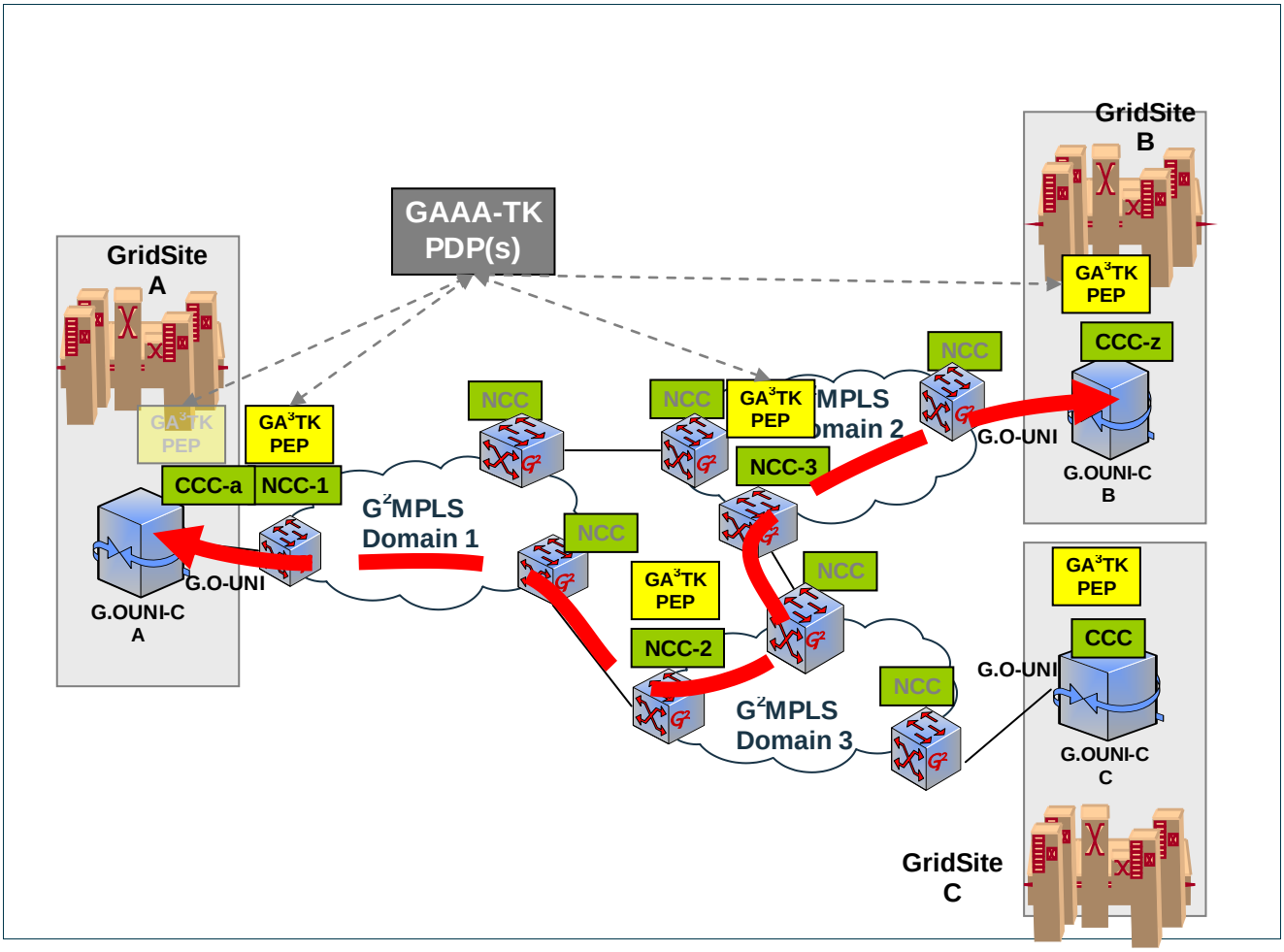

Figure 3-2: Inter-domain signalling

3.3 Interaction between G²MPLS control plane and GAAA framework

Figure 3-3 depicts the messages exchanged between the G²MPLS control plane modules and the GAAA framework; in particular the actors of the interaction are the NCCs and the corresponding AuthZ-GW, that acts as a wrapper for the PEP and the TVS of the GAAA-NRP infrastructure.

During the setup phase, the authorization request on the NCC of the first domain (NCC-1 in Figure 3-2) requires both the user credentials and a complete resource description, including source and destination client nodes. The AuthZ-GW translates these parameters in a set of data structures (*resmap*, *subjmap* and *actmap*) as required by the GAAA-TK library API and triggers the authorization procedure on the PEP. If AuthZ result is positive, the PEP requests TVS to generate a new GRI for the current call and build a pilot token that will be send to the next downstream domain. The TVS token handling functionality ensures end-to-end token based

signalling validity. Each downstream NCCs can use GRI communicated by a token as a unique reference for the call. Depending on the type the pilot token can either fully communicate the associated security context or just reference it.

The token exchange between NCC located in the same domain is achieved by the intra-domain call signalling, while for inter-domain signalling the token is included in the RSVP Policy Object and transferred through the G.E-NNI/G.UNI RSVP signalling.

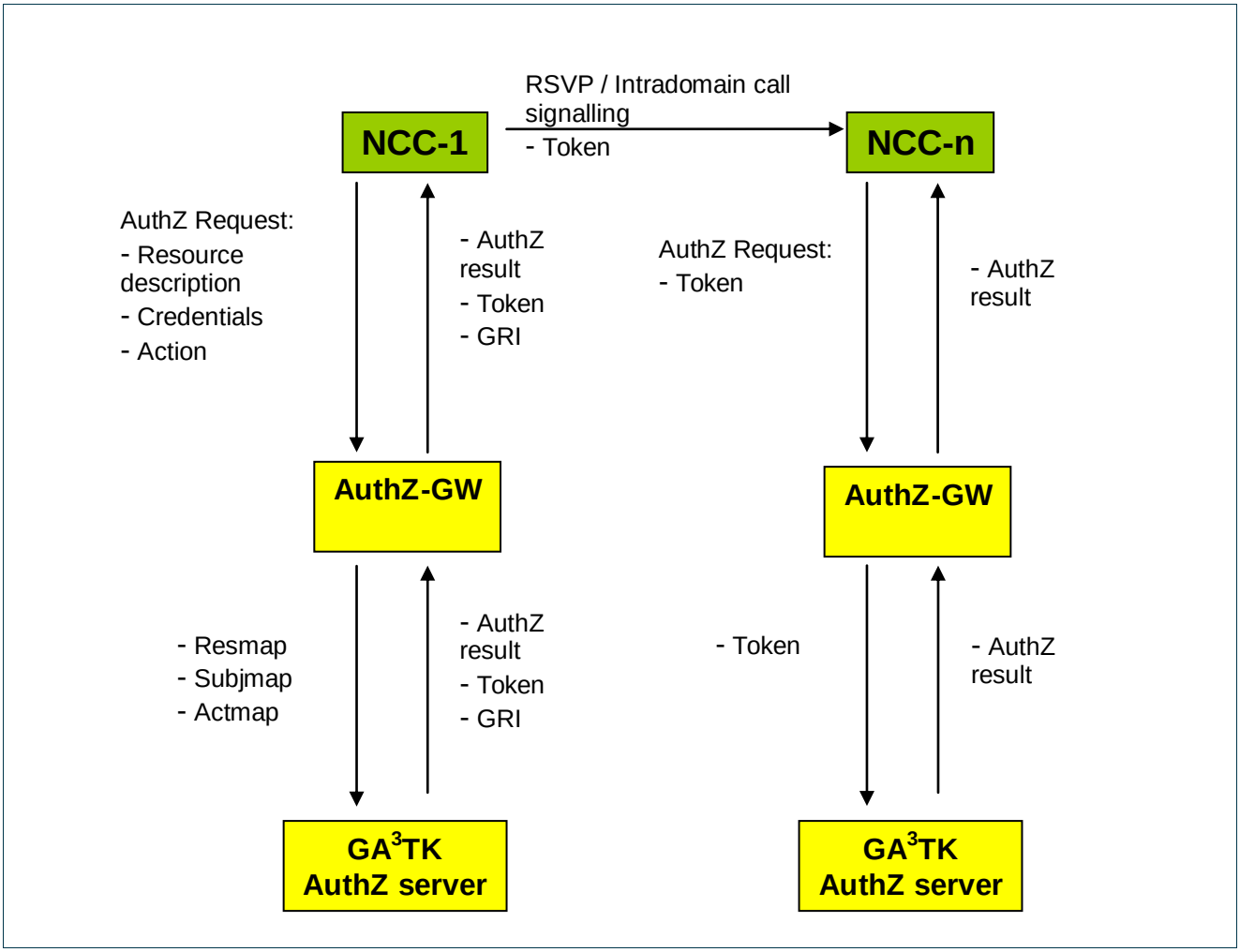

Figure 3-3: NRPS/GAAA communications.

4 Signalling extensions

4.1 G.OUNI and G.E-NNI RSVP Policy Object extensions for token signalling

The token is carried in the RSVP signalling using the RSVP Policy Object depicted in Figure 4-1. This extension complies with its specification in RFC 2750 **[IETF-RFC2750]**.

This extension applies to both G.OUNI RSVP-TE and G.E-NNI RSVP-TE.

Type 1 POLICY_DATA object: Class = 14, C-Type = 1

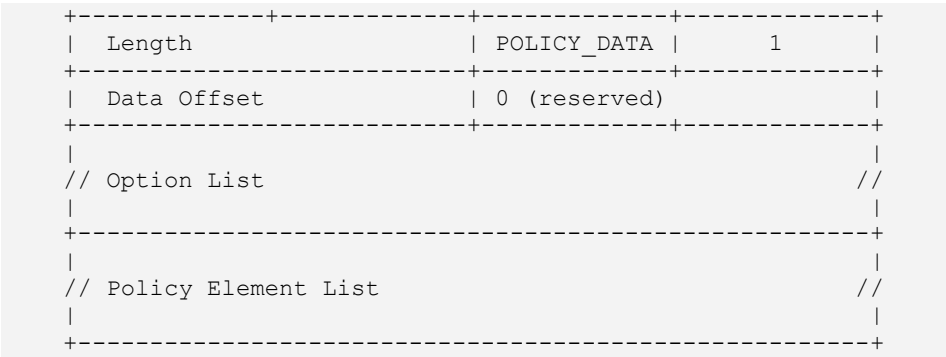

Figure 4-1: RSVP Policy Object.

The Policy Element List field contains one Policy Element (Figure 4-2) that includes the token. The P-Type associated to the token is PTYPE_PHOSPHORUS_AUTHZ_TOKEN = 54000.

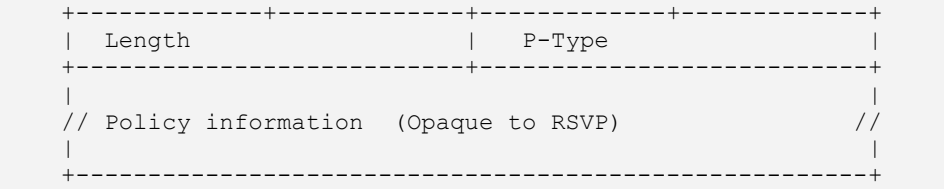

Figure 4-2: Policy Element.

4.2 G² Call intra-domain signalling extensions

When the AuthZ information is exchanged between NCCs belonging to the same G²MPLS domain, it travels in an extension of the G² Call intra-domain XML signalling (defined in D2.3).

The extension applies to the SetupRequest message, as reported in the DTD below; the AuthZ tag is highlighted.

```
<!ELEMENT ccsigmsg (header, body)>

      <!ELEMENT header (type, seqnum, sender)>
            <!ELEMENT type (#PCDATA[1])>
            <!ELEMENT seqnum (#PCDATA)>
            <!ELEMENT sender (#PCDATA)>


      <!ELEMENT body (name, client-name?, call-id?, indirect?, rel-ind-call-id?,
      emulated-if?, call-parms?, lsp-parms?, ero?, reason?, errored-seqnum?,
      authz-token?)>

            <!ELEMENT name (#PCDATA)>
            <!ELEMENT client-name (#PCDATA)>

            <!ELEMENT call-id (type, srcId, localId, segments?)>
               <!ELEMENT type (#PCDATA)>
               <!ELEMENT srcId (#PCDATA)>
               <!ELEMENT localId (#PCDATA)>

            <!ELEMENT indirect (#PCDATA)>
            <!ELEMENT rel-ind-call-id (type, srcId, localId, segments?)>
            <!ELEMENT emulated-if (#PCDATA)>
            <!ELEMENT reason (#CDATA)>
```

[1] A string indicating one of the message types reported above.

```
<!ELEMENT errored-seqnum (#PCDATA)>

<!ELEMENT authz-token (#PCDATA)>

<!ELEMENT ero (eroelem +)>
   <!ELEMENT eroelem (nodeId, teLink, upDataLink, upLabel,
   downDataLink, downLabel, loose)>
      <!ELEMENT nodeId (#PCDATA)>
      <!ELEMENT teLink (#PCDATA)>
      <!ELEMENT upDataLink (#PCDATA)>
      <!ELEMENT upLabel (#PCDATA)>
      <!ELEMENT downDataLink (#PCDATA)>
      <!ELEMENT downLabel (#PCDATA)>
      <!ELEMENT loose (#PCDATA)>

<!ELEMENT call-parms (originator, jobProject, jobName, gnstnas,
disjointness, recoveryType, startTime, endTime, tnares)>
   <!ELEMENT originator (#PCDATA)>
   <!ELEMENT jobProject (#CDATA)>
   <!ELEMENT jobName (#CDATA)>
   <!ELEMENT disjointness (#PCDATA)>
   <!ELEMENT startTime (#PCDATA)>
   <!ELEMENT recoveryType (#PCDATA)>
   <!ELEMENT endTime (#PCDATA)>
   <!ELEMENT tnares (ingress, egress)>
      <!ELEMENT ingress (dataLink, label, tna)>
      <!ELEMENT egress (dataLink, label, tna)>
           <!ELEMENT dataLink (#PCDATA)>
           <!ELEMENT label (#PCDATA)>
           <!ELEMENT tna (#PCDATA)>
   <!ELEMENT gnstnas (ANY)>

<!ELEMENT lsp-parms (lspRole, lspType, swCap, encType, gpid,
bandwidth, tnResAction, rroMode, setupPrio, holdingPrio, linkProtMask,
includeAll, includeAny, excludeAny, useAcks, rapidRetryLimit,
rapidRetransIntval, incrementValueDelta, refreshInterval,
crankbackScope, maxCbackRetrSrc, maxCbackRetrIntmd)>
   <!ELEMENT lspRole (#PCDATA)>
   <!ELEMENT lspType (#PCDATA)>
   <!ELEMENT swCap (#PCDATA)>
   <!ELEMENT encType (#PCDATA)>
   <!ELEMENT gpid (#PCDATA)>
   <!ELEMENT bandwidth (#PCDATA)>
   <!ELEMENT tnResAction (#PCDATA)>
   <!ELEMENT rroMode (#PCDATA)>
   <!ELEMENT setupPrio (#PCDATA)>
   <!ELEMENT holdingPrio (#PCDATA)>
   <!ELEMENT linkProtMask (#PCDATA)>
   <!ELEMENT includeAll (#PCDATA)>
   <!ELEMENT includeAny (#PCDATA)>
   <!ELEMENT excludeAny (#PCDATA)>
   <!ELEMENT useAcks (#PCDATA)>
   <!ELEMENT rapidRetryLimit (#PCDATA)>
   <!ELEMENT rapidRetransIntval (#PCDATA)>
```

```
<!ELEMENT incrementValueDelta (#PCDATA)>
<!ELEMENT refreshInterval (#PCDATA)>
<!ELEMENT crankbackScope (#PCDATA)>
<!ELEMENT maxCbackRetrSrc (#PCDATA)>
<!ELEMENT maxCbackRetrIntmd (#PCDATA)>
```

The *authz-token* tag carries a string information element with the plain token.

5 G²MPLS gateway to GAAA-TK (AuthZ-GW)

5.1 AuthZ-GW basics

The AuthZ Gateway (AuthZ-GW) allows the NCCs to interact with the GAAA-AuthZ infrastructure's Domain Central AuthZ Service (DCAS) through the GAAA-TK PEP, so that resource provisioning procedures like resource reservation and release can be authorized in each node along the multi-domain path using a token-based model. On the first domain each resource reservation request associated to a new session is authorized through the AuthZ-GW. The following information are provided to the GAAA-AuthZ framework DCAS (PEP + PDP) and here evaluated according to the existing policies:

- subject credentials: subject ID, subject role, subject confdata, subject context;
- resource description: resource ID, resource realm, resource domain, resource type, source, target;
- action type; the following actions are supported: create-path, cancel.

If the request is authorized, the AuthZ-GW creates a Global Reservation Identifier for the specific session and builds a new token using the methods exported by the Token Validation Service (TVS) of the GAAA-AuthZ framework. This token acts as a reference for the global security context associated to the current session and can be used by the AuthZ-GWs located on the following NCCs along the path in order to authorize the resource allocation on each segment without providing any specification about the actual context (subject credentials and resource description). The token is carried end-to-end in the NCCs signalling and each involved AuthZ-GW is in charge to validate it with the GAAA-AuthZ framework using the TVS validation mechanisms exposed by the GAAA-TK library.

Further information about security context specifications and token-based authorization mechanism can be found in [**PH-WP4-D4.3.1**].

5.2 AuthZ-GW external interfaces (XML)

The AuthZ-GW interacts with the NCCs modules through an interface based on XML-RPC specification that defines a remote procedure calling using HTTP as transport protocol and XML as encoding protocol [XML-RPC]. This specification allows an efficient interaction between the AuthZ-GW, implemented in Java, and the NCCs, implemented in Python.

The AuthZ-GW acts as a wrapper for the GAAA-TK library, making transparent the interaction with the GAAA-AuthZ framework mechanisms for the NCCs. During NCC-AuthZ-GW communications, the AuthZ-GW acts as a server, while the NCC acts as a client (Figure 5-1).

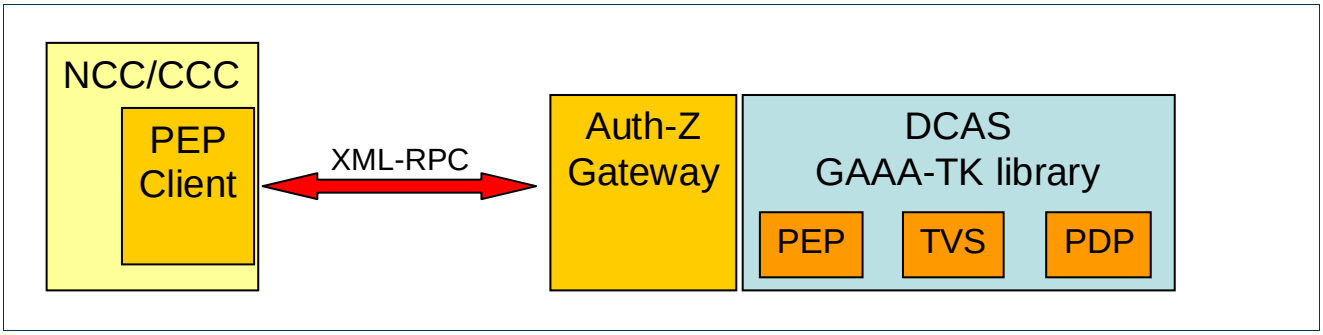

Figure 5-1: NCC – AuthZ-GW interaction.

The methods exposed by the AuthZ-GW for the PEP Client located on the NCCs are detailed in Table 5-1. The “getToken” method is used at the first domain in order to authorize the resource reservation specifying credentials and resource description, allowing the NCC to obtain the token for the current session. The “getAuthorization” method can be used by the following NCCs in order to validate this token and receive the authorization for resource reservation in each specific segment.

Method	Input parameters	Output Parameters
getToken	Domain ID Subject ID Action type Source Target	Authorization result code Session ID (Global Reference ID) Token
getAuthorization	Token	Authorization result code

Table 5-1: Authorization methods exposed by the AuthZ-GW.

The Java documentation for the AuthZ-GW public methods can be found in the appendix.

5.3 AuthZ-GW core behaviour

The internal architecture of the AuthZ-GW is depicted in Figure 5-2.

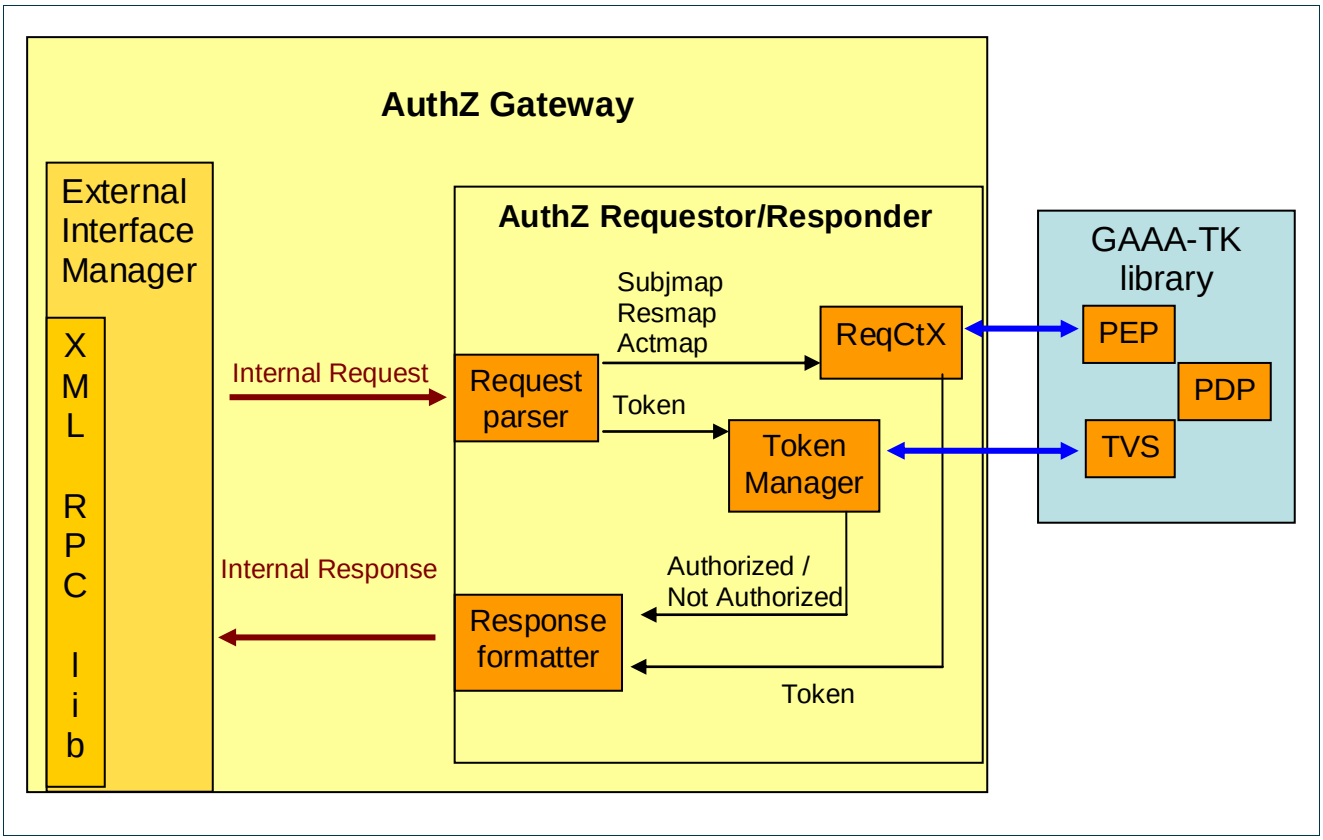

Figure 5-2: AuthZ-GW architecture.

The External Interface Manager handles the interaction with the PEP Client through the XML-RPC library. Incoming requests are translated in internal data structures and processed by the AuthZ Requestor/Responder (ARR). Each request is parsed in order to prepare a set of parameters required by the Request Context Handler (ReqCtX) and the Token Manager to interact with the GAAA-TK library. Responses are then formatted as required by the external interface specifications and sent back to the client through the XML-RPC library.

The ReqCtX is in charge of interacting with the GAAA-TK PEP in order to authorize the first resource reservation, creating the Global Resource ID and the related token. The required input parameters are the following:

- HashMap<String, String> resmap
- HashMap<String, String> actmap

- Hashmap<String, String> subjmap

where the resmap includes the following parameters:

- resource-id
- resource-realm
- resource-domain
- resource-type
- source
- target

The actmap includes the action ID and the action type (“create-path” or “cancel”). The subjmap includes the subject credentials, in particular:

- subject-id
- subject-confdata
- subject-role
- subject-context

The ReqCtX processing consists of the following three different steps: checking with the PEP if resources can be authorized; creating a new Global Resource Identifier using the security methods provided by the GAAA-TK library and finally building the related XML token with the required validity time.

On the other hand the Token Manager is just in charge of validating the received XML tokens using the GAAA-TK TVS token validation interface. Validation result is then encoded in the corresponding result code and sent to the response formatter.

6 AuthZ logic in the G² Call Control layer

The logic for AuthZ procedures is embedded in the G² Call FSM at the Network Call Controller (NCC). This FSM design is compliant with ITU-T rec. G.7713/Y.1704 **[ASON-DCM]**.

Two FSM states are foreseen, one in the Call setup phase (*VerifyCallSetupRequest*) and one in the Call teardown phase (*VerifyCallReleaseRequest*), to perform, respectively, the verification of Call setup requests and Call release requests. The Call setup process does not actually start until the *VerifyCallSetupRequest* has been exited successfully. The same applies to Call release. The negative outcome of these two states leaves the Call in the previous stable state (i.e. *Idle* and *Active,* respectively).

The G² Call FSM has been documented in D2.3. In the following, its specification is reported, with the *VerifyCallSetupRequest* and *VerifyCallReleaseRequest* highlighted.

State	short description
Idle	The Call has been created, but no signalling has occurred on it yet.
VerifyCallSetupRequest	The call setup signalling has been initiated (either a *SetupRequest* was received from the network, or a management command has been issued), and policy verification has started (i.e. an AuthZ request has been sent to the AAI). Waiting for a reply to the policy verification. Depending on the policy configuration, this state can be skipped at some NCCs (e.g. it can be valid only for the ingress ones, downstream of UNI or E-NNIs).
CallSetupRequestInitiated	The policy verification concluded successfully (or it was simply skipped), and the *SetupRequest* message has been propagated downstream. Waiting for an answer to it (*SetupIndication*).
CallSetupResponded	A *SetupIndication* has been received from the downstream NCC (or CCC if the downstream NI is a UNI). Waiting for the Call to be fully completed (i.e. the NCC has to see a *SetupConfirm* concerning this Call).
SetupConnection	The *SetupConfirm* has been received (or sent, if the Call FSM is at NCC-1), and the Call setup signalling has successfully completed. The setup of the network connections has started (i.e. the creation and setup of Recovery Bundles at the Recovery Controller have been commanded). Waiting for this process to successfully complete.

VerifyCall	The Call is now equipped with network connections (i.e. Recovery Bundles and LSPs). This state can be optionally used at some NCCs (e.g. upstream ones) to verify the Call connectivity across the domain. If this is not foreseen, the Call jumps to the *Active* state.
Active	The Call has now reached is up steady state: it has been authorized, signalled, equipped with network connections and (optionally) verified at Data Plane level.
SigError	An alternate steady state w.r.t. the *Active* one: some signalling error has occurred on the Call after its setup.
VerifyCallReleaseRequest	The call teardown signalling has been initiated (either a *ReleaseRequest* was received from the network, or a management command has been issued), and policy verification has started (i.e. an AuthZ request has been sent to the AAI). Waiting for a reply to the policy verification. Depending on the policy configuration, this state can be skipped at some NCCs (e.g. it can be valid only for the ingress ones, downstream of UNI or E-NNIs).
ReleaseConnection	The policy verification concluded successfully (or it was simply skipped); now the teardown has been authorized. The teardown of network connections has started (i.e. proper teardown commands have been issued to the Recovery Controller concerning the Recovery Bundle associated to this Call). Waiting for the network connections to be torn down.
CallReleaseRequestInitiated	All the network connections associated to this Call have been torn down (i.e. no more RBs at RC, and LSPs at G².RSVP-TE), and the *ReleaseRequest* message has been propagated upstream or downstream. Waiting for an answer to it (*ReleaseIndication*); when it will come, the Call will jump back to its *Idle* state and be deleted.

Table 6-1: G².NCC Call FSM: call AuthZ states

Two excerpts of the NCC FSM (setup and teardown states) is reported in the following figures.

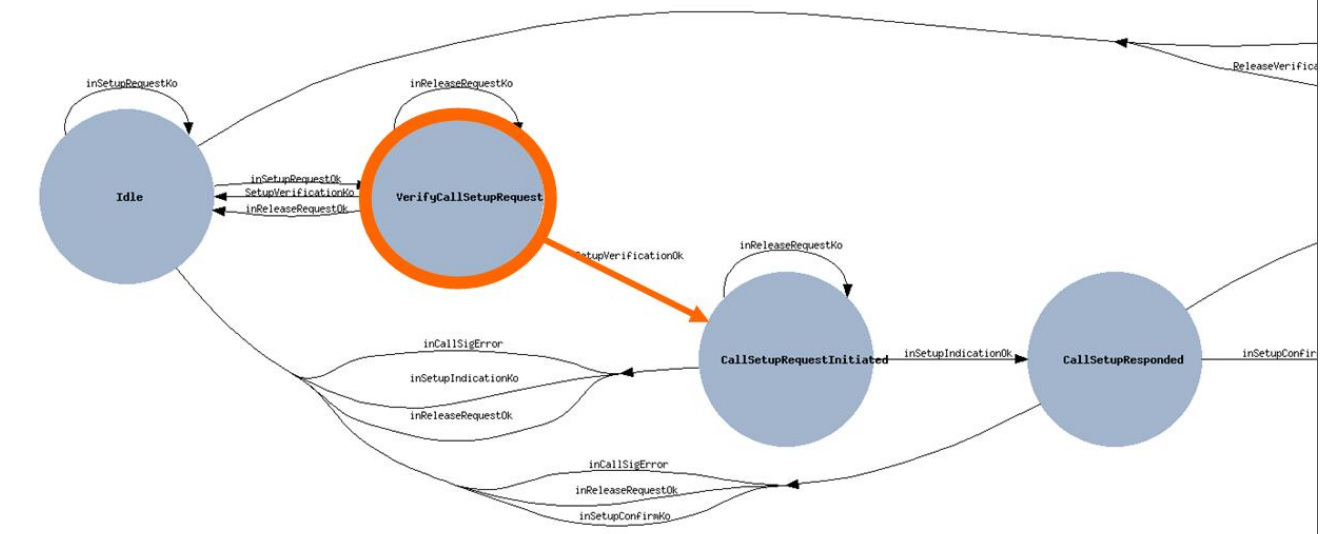

Figure 6-1: AuthZ in G².NCC Call FSM – setup phase.

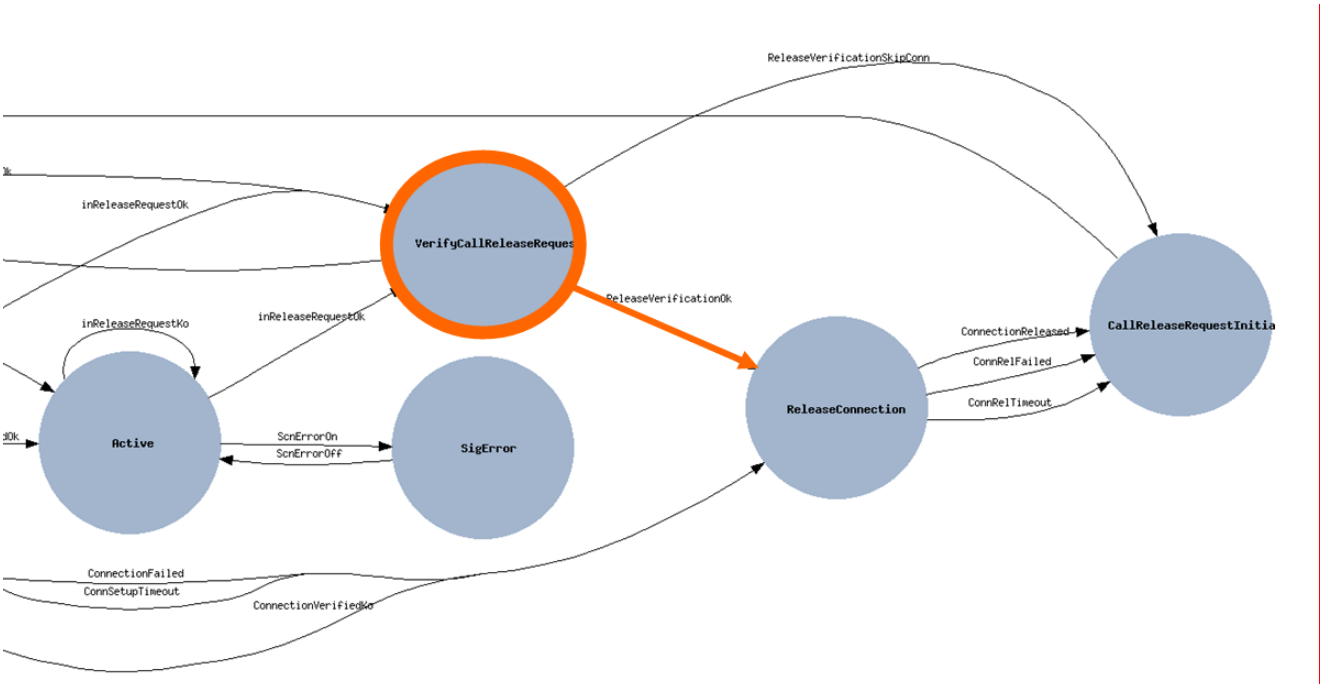

Figure 6-2: AuthZ in G².NCC Call FSM – teardown phase.

The AuthZ procedures (i.e. enabling actions in *VerifyCallSetupRequest* and *VerifyCallReleaseRequest* states) are activated if the PEP-GW URL is configured at the NCC.

When a the G² Call FSM enters the *VerifyCallSetupRequest* state, the authorization procedure is invoked with the *create-path* action.

- At the G.OUNI-N NCC, the subscriber's parameters are processed for authorization. If successful, a token generation procedure is invoked and the token is bound to the Call signalling (in the SetupRequest message for intra-domain signalling and G².RSVP-TE Path message at G.E-NNI and far end G.OUNI).

- At and downstream G.E-NNI NCC, the token (extracted from the G.E-NNI RSVP-TE Policy Object) is authorized and, if successful, forwarded in the ongoing Call signalling

The same applies when the G² Call FSM enters the *VerifyCallReleaseRequest* state on teardown. The authorized action is now *cancel.* The piggybacking Call signalling messages are ReleaseRequest for intra-domain signalling and G².RSVP-TE Path/Resv [A=1, D=1] messages at G.E-NNI and far end G.OUNI.

7 Closing notes

This document specifies the usage of the authorization mechanisms supported by the GAAA-TK library and integrated by the G^{2}MPLS control plane in order to authorize the resource reservation/deletion during the call setup and teardown phases in, both single and multi-domain scenarios. The authorization decisions rely on the usage of a pilot token with end-to-end validity.

In particular, the document presents the AuthZ functionalities and the related implementation of the Network Control Plane modules involved in the authorization procedures, and the interactions among these entities. It provides details on the end-to-end signalling to carry the security context data (i.e. the pilot token) along the path and the specifications of the communication between the xCCs and the AuthZ gateway.

This module acts as a wrapper for the GAAA-TK library exposing the methods required by the control plane to authorize the resource reservation, allowing an easy interaction between the control plane and GAAA-NRP infrastructure.

Overall, the presented design and development of GAAA functions in G²MPLS brings a significant extension in its G.UNI and G.E-NNI network reference points.

In particular, the availability of a G.UNI authorization based on a set of subscriber profile properties and rights makes it a more powerful interface, and could enable its deployment in a number of operational scenarios, with or without commercial purposes.

Similarly, the availability of a G.E-NNI with authorization capabilities on a single Call granularity could promote this originally inter-vendor interface (according to OIF) to an *inter-carrier* network reference points. A number of requirements of an inter-carrier interface could be fulfilled with this extension, such as the enforcement of a Service Level Agreement (SLA), or the dynamic establishment of Service Level Specifications (SLS) based on that SLA.

Therefore, the design reported in this document, and the related development that occurred, go together in the direction of augmenting the capabilities of the G²MPLS architecture, in order to make it a concrete solution for the dynamic establishment of multi-domain transport services.

8 References

The references listed here are only those directly functional to this document. For a full list of the references to standards appearing in this document, please point to D2.1, D2.2, D2.3, D2.4, D2.6 and D2.7.

[PH-WP2-D2.1]	Phosphorus deliverable D2.1, “The Grid-GMPLS Control Plane architecture”.
[PH-WP2-D2.2]	Phosphorus deliverable D2.2, “Routing and Signalling Extensions for the Grid-GMPLS Control Plane”.
[PH-WP2-D2.6]	Phosphorus deliverable D2.6, “Deployment models and solutions of the Grid-GMPLS Control Plane”.
[PH-WP2-D2.7]	Phosphorus deliverable D2.7, “Grid-GMPLS network interfaces”.
[PH-WP2-D2.3]	Phosphorus deliverable D2.3, “Grid-GMPLS high level system design”.
[PH-WP2-D2.4]	Phosphorus deliverable D2.4, “Report on Grid-GMPLS Control Plane functional tests”.
[PH-WP4-D4.3.1]	Phosphorus deliverable D4.3.1, “GAAA toolkit pluggable components and XACML policy profile for ONRP”.
[IETF-RFC2750]	S. Herzog, “RSVP Extensions for Policy Control”, IETF RFC 2750, January 2000.
[ASON-DCM]	ITU-T G.7713/Y.1704 Recommendations, “Generalised Distributed Connection Management”, 2001.
[QUAGGA-DOC]	The Quagga Software Routing Suite documentation. http://www.quagga.net/docs/docs-info.php
[CORBA]	http://www.corba.org/
[omniORB]	http://omniorb.sourceforge.net/
[XML-RPC]	http://www.xmlrpc.com

9 Acronyms

AAA	Authentication, Authorisation, and Accounting
AAI	Authentication and Authorization Infrastructure
ANSI	American National Standards Institute
API	Application Programming Interface
ARGON	Allocation and Reservations in Grid-enabled Optical Networks
ASON	Automatically Switched Optical Network
BB	Bandwidth Broker
BGRP	Border Gateway Reservation Protocol
BoD	Bandwidth on Demand
BR	Border Router
CCC	Client Call Controller
CE	Computing Element
CIM	Computer Integrated Manufacturing
COPS	Common Open Policy Protocol
CORBA	Common Object Request Broker Architecture
CP	Control Plane
CPE	Customer Premises Equipment
CPU	Central Processing Unit
CR-LDP	Constraint-based Label Distribution Protocol
DCAS	Domain Central AuthZ Service
DCM	Distributed Call and Connection Management
DCN	Data Communication Network
DRAC	Dynamic Resource Allocation Controller
DVB	Digital Video Broadcasting
DWDM	Dense Wavelength Division Multiplexing
EGEE	Enabling Grids for E-sciencE
EC	European Commission
EMS	Execution Management Services
E-NNI	Exterior NNI
ERO	Explicit Route Object
ETSI	European Telecommunications Standards Institute
EU	European Union

FCAPS	Fault, Configuration, Accounting, Performance, Security
G.CR-LDP	G^2MPLS CR-LDP
G.OSPF-TE	GMPLS OSPF-TE
G.OUNI	Grid OUNI
G.OUNI-C	G.OUNI - Client
G.OUNI-N	G.OUNI - Network
G.RSVP-TE	GMPLS RSVP-TE
GAAA-AuthZ	Generic AAA Authorization Framework
GAAAPI	Generic Authentication/Authorization Application Programming Interface
G²MPLS	Grid-GMPLS (enhancements to GMPLS for Grid support)
GE	Gigabit Ethernet
GÉANT	Pan-European Gigabit Research Network
GGF	Global Grid Forum
GHPN	Grid High Performance Networking
GIS	Grid Information Service
GLUE	Grid Laboratory Uniform Environment
GMPLS	Generalized MPLS
GNS	Grid Network Service
GRAM	Grid Resource Allocation and Management
GRI	Global Reservation Identifier
GSMP	General Switch Management Protocol
HW	Hardware
IANA	Internet Assigned Numbers Authority
IDM	GÉANT2 Inter-domain Manager
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IGP	Interior Gateway Protocol
I-NNI	Interior NNI
IP	Internet Protocol
IPR	Intellectual Property Right
IPSec	IP security
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
IS-IS	Intermediate System to Intermediate System
ITU	International Telecommunication Union
JSDL	Job Submission Description Language
LAN	Local Area Network
LDP	Label Distribution Protocol
LRI	Local Reservation Identifier
LRMS	Local Resource Management System
LSA	Link State Advertisement
LSDB	Link State Database

LSP	Label Switched Path
LSR	Label Switch Router
MAC	Media Access Control
MAN	Metropolitan Area Network
MP	Management Plane
MPLS	Multi Protocol Label Switching
MPI	Message Passing Interface
NCC	Network Call Controller
NCP	Network Control Plane
NJS	Network Job Supervisor
NMS	Network Management System
NNI	Network to Network Interface
NO	Network Operator
NREN	National Research and Education Network
NRPS	Network Resource Provisioning Systems
NSAP	Network Service Access Point
NSP	Network Service Plane
NTP	Network Time Protocol
OAM	Operations, Administration and Maintenance
OHRM	Obligation Handling Reference Model
OGF	Open Grid Forum
OGSA	Open Grid Services Architecture
OIF	Optical Internetworking Forum
OS	Operating System
OSPF	Open Shortest Path First protocol
OSPF-TE	OSPF with Traffic Engineering extensions
O-UNI	Optical UNI
P2MP	Point to Multi Point
PAP	Policy Authority Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
PKI	Public Key Infrastructure
PON	Passive Optical Network
POSIX	Portable Operating System Interface
QoS	Quality of Service
RC	Routing Controller
RFC	Request for Comments
RSVP	Resource reSerVation Protocol
RSVP-TE	RSVP with Traffic Engineering extensions
RTP	Real-time Transport Protocol
SDO	Standard Developing Organizations
SE	Storage Element

SLA	Service Level Agreement
SLS	Service Level Specification
SME	Small and Medium Enterprise
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
SP	Service Provider
SPF	Sender Policy Framework
SW	Software
TE	Traffic Engineering
TGC	Trusted Computing Group
TL-1	Transaction Language 1
TLS	Transport Layer Security
TLV	Type-Length-Value protocol fields
TMF	Tele Management Forum
TO	Telecom Operator
TP	Transport Plane
TVS	Token Validation Service
UCLP	User-Controlled Lightpath Provisioning system
UNI	User to Network Interface
UML	Unified Modeling Language
URI	Uniform Resource Identifier
VLAN	Virtual LAN
VPN	Virtual Private Network
WAN	Wide Area Network
WG	Working Group
WP	Work Package
WS	Web Service
XML	Extensible Markup Language

Appendix A AuthZ Gateway Java doc

This appendix presents a summary from the PEP Gateway java documentation, showing the method exposed by the AuthZ-GW for authorization and token validation procedures

it.nextworks.PEP-GW

Class PEP-GW

java.lang.Object

extended by **it.nextworks.PEP-GW.PEP-GW**

Constructor Summary

PEP-GW(Configuration conf, Logger log)

Constructor

…

Method Summary	
int	**getAuthorization**(java.lang.String token) Method for the next authorization requests.
java.util.Map <java.lang.String,java.lang.String>	**getToken**(java.lang.String domainId, java.lang.String subjectId, java.lang.String action, java.lang.String source, java.lang.String target) Method for the first authorization request, return token and sessionID.
static void	**main**(java.lang.String[] args

9.2 *getToken()* method

```
public java.util.Map<java.lang.String,java.lang.String>
getToken(java.lang.String domainId,

java.lang.String subjectId,

java.lang.String action,

java.lang.String source,

java.lang.String target)
```

Method for the first authorization request, return token and sessionID.

Return the hashtable with the following String parameter:
key "SessionID"
key "Token"
key "ResultCode"

Available Result Code:
OK = 1;
NOT_AUTHENTICATED = 2;
NOT_AUTHORIZED = 3;
NOT_AVAILABLE_PDP = 4;
WRONG_REQUEST_PARAMETERS = 5;
MISSING_REQUEST_PARAMETERS = 6;
UNABLE_TO_PROCESS_REQUEST = 7;

Parameters:
`domainId` - i.e.: "http://testbed.ist-phosphorus.eu"
`subjectId` - i.e.: "WHO740@users.collaboratory.nl"
`action` - Possible values: "access", "cancel", "create-path", "activate-path"
`source` - IP source address
`target` - IP destination address

Returns:
sessionID, token and result code

9.3 *getAuthorization()* method

```
public int getAuthorization(java.lang.String token)
```

Method for the next authorization requests. It returns an integer with the result code.

Available Result Code:
OK = 1;
NOT_AUTHENTICATED = 2;
NOT_AUTHORIZED = 3;

NOT_AVAILABLE_PDP = 4;
WRONG_REQUEST_PARAMETERS = 5;
MISSING_REQUEST_PARAMETERS = 6;
UNABLE_TO_PROCESS_REQUEST = 7;

Parameters:
`token` - XML token created using the getToken method.

Returns:
the result code.

<END-OF-DOCUMENT>