Subject: **AUTHORIZATION SERVICE DEPLOYMENT PLAN**

Author(s): **Authorization Service Group**
SA1, SA3

Distribution

DRAFT

Abstract:

JRA1 is developing a new authorization service in EGEE-III. This document describes a deployment plan for the remainder of the EGEE-III project, through which this service can be deployed in six distinct steps.

Content:

Abbreviations:

We use the following abbreviations in this document:

CE	Computing Element: a Grid-enabled computing resource
DN	Distinguished Name to uniquely denote a user or entity. The DN is typically used in directories like X.500 and LDAP, but also within X.509 certificates.
EES	Execution Environment Service: Service internal to the new authorization system, which is being implemented within EGEE-III.
FQAN	Fully Qualified Attribute Name: A string containing VOMS group and (optionally) role information.
OSCT	Operational Security Coordination Team: Team that is responsible for ensuring the overall security coordination in EGEE.
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
SE	Storage Element: a Grid enabled storage resource.
Site	Organization having administrative control of resources provided to the Grid. This may be at one physical location of spread across multiple locations.
WMS	Workload Management System: a central broker for job submission and management
WN	Worker Node: the entity where jobs get executed

References:

R1	C.Witzig: Authorization in gLite: https://edms.cern.ch/document/887174/1
R2	Authorization Service Group: Authorization Service Design: https://edms.cern.ch/document/944192/1
R3	M.Altunay, C.Witzig: Security command line tools for system administrators: https://edms.cern.ch/document/931846/1

1 INTRODUCTION

The Authorization Service (in the following called "the service") is being developed by the JRA1 activity of the EGEE-III project. The following institutions are involved: CNAF, HIP, NIKEHF and SWITCH.

A review [R1] of the authorization mechanisms in gLite revealed several inconsistencies in the authorization framework. In addition, different services use today different authorization methods. Thus site administrators must know different authorization frameworks, and they do not have a single point of configuration for authorization at the site. This service addresses these issues.

The design of the service is documented in [R2], to which we refer for further information on technical details. It is currently expected that the service enters the certification stage in March 2009.

The purpose of this document is to describe a series of *proposed* deployment steps through which this service can gradually be deployed in the EGEE production infrastructure. As such, this document shall be the basis for further discussion within EGEE/LCG.

The long-term goal of this service is to implement a consistent authorization management through policies across the infrastructure. This document only describes the *initial* stages of this deployment process through a series of 6 steps. In each step a new functionality is added. Note that some steps are dependent on each other, whereas other steps are independent of each other.

We foresee the following six steps:

- Step 1: Support for glexec on the Worker Node
- Step 2: Global banning list by OSCT
- Step 3: Integration into CREAM
- Step 4: Integration into WMS for authorization
- Step 5: Integration into WMS for match-making
- Step 6: Integration into data management

Figure 1 illustrates the dependencies between these six steps, on which we further elaborate in the different sections of this document.

It should be pointed out that we consider these six steps as the initial steps for introducing the service into the deployment. The service offers more advantages beyond the lifetime of EGEE-III as more services start of use it and more complex authorization policies are being demanded by sites and VOs. However, we clearly state that this is outside the scope of this document and that the advantages of these six steps in our view already warrant the effort to deploy this service.

We see as main **benefits** of deploying this service:

1. The service provides a unified authorization mechanism across Grid services (with an initial focus on the six steps as described in this document)
2. The service has no single point of failure and can be scaled horizontally
3. Command line support for all commonly used administration and debugging commands
4. Support for metrics and monitoring (Nagios plug-ins will be provided as part of the project)

5. True audit capabilities provide details information about who was authorised when and by which policy.
6. Good performance (Numbers will need to be provided in February)
7. Easy installation and proper documentation (to be verified in the certification step)
8. Support for hierarchical and/or site-wide policies
9. Bindings for other programming languages can easily been developed (e.g. python)
10. Support for new execution environments in the future (see below)
11. The service is based on standards (e.g. Web-Services, XACML) and is dependent on well-established libraries such as OpenSAML. This eases the long-term maintainability.
12. The service is designed to be extendable in order to support new credential formats and leverage against other authentication and authorization infrastructures.
13. Four institutions are involved in the design and implementation of the service, which is favourable for its long-term maintainability.

The authorization service also contains a component called Execution Environment Service (EES). Its *initial* focus is on providing the mapping from Grid credential into a set of UID/GIDs as it is already done today. However, in later versions it will provide support for new executions environments beyond the UID/GID mapping. Examples for such environments are virtual machines and/or workspaces. A detailed deployment plan for this component needs to be worked out in the near future and is not part of this document.

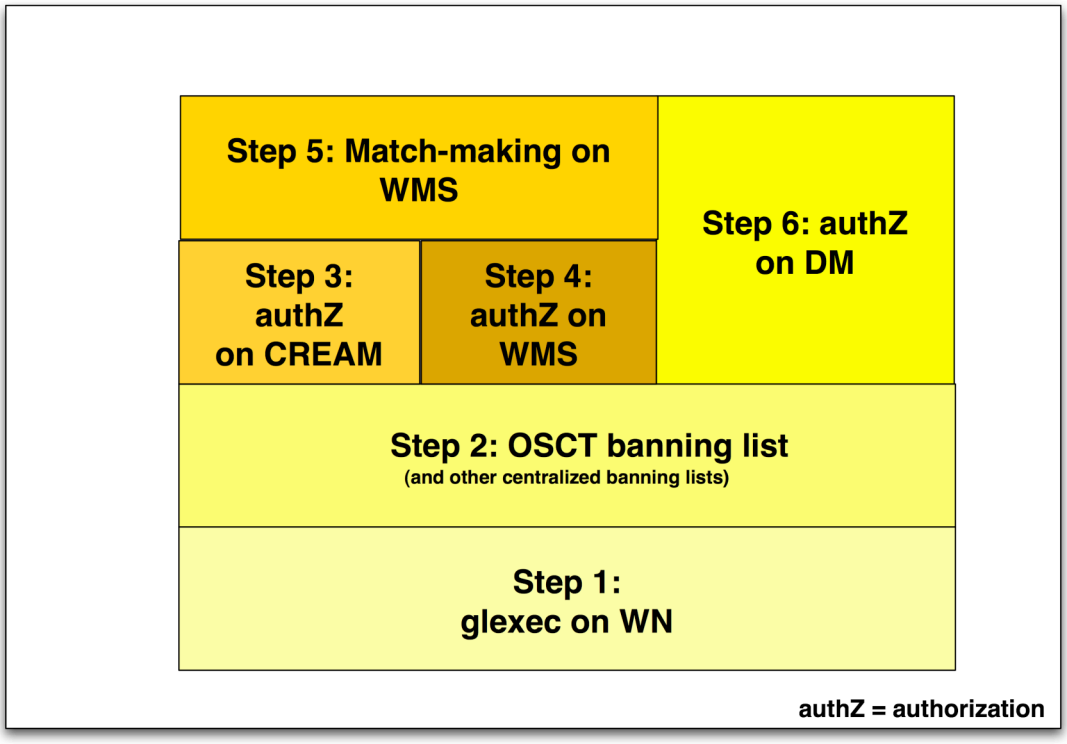

Figure 1 Dependencies of the different deployment steps

2 STEP 1: SUPPORT FOR GLEXEC ON THE WORKER NODE

In step 1 the service is introduced at sites with the sole purpose to enable glexec on the Worker Node (WN). The typical setup for a CE is shown in Figure 2 and consists of adding the following components to a site supporting the gLite CE:

1. The three components (PAP, PDP, PEP daemon) of the service are installed on a single separate host.
2. An upgrade version of glexec that contains the PEP client.

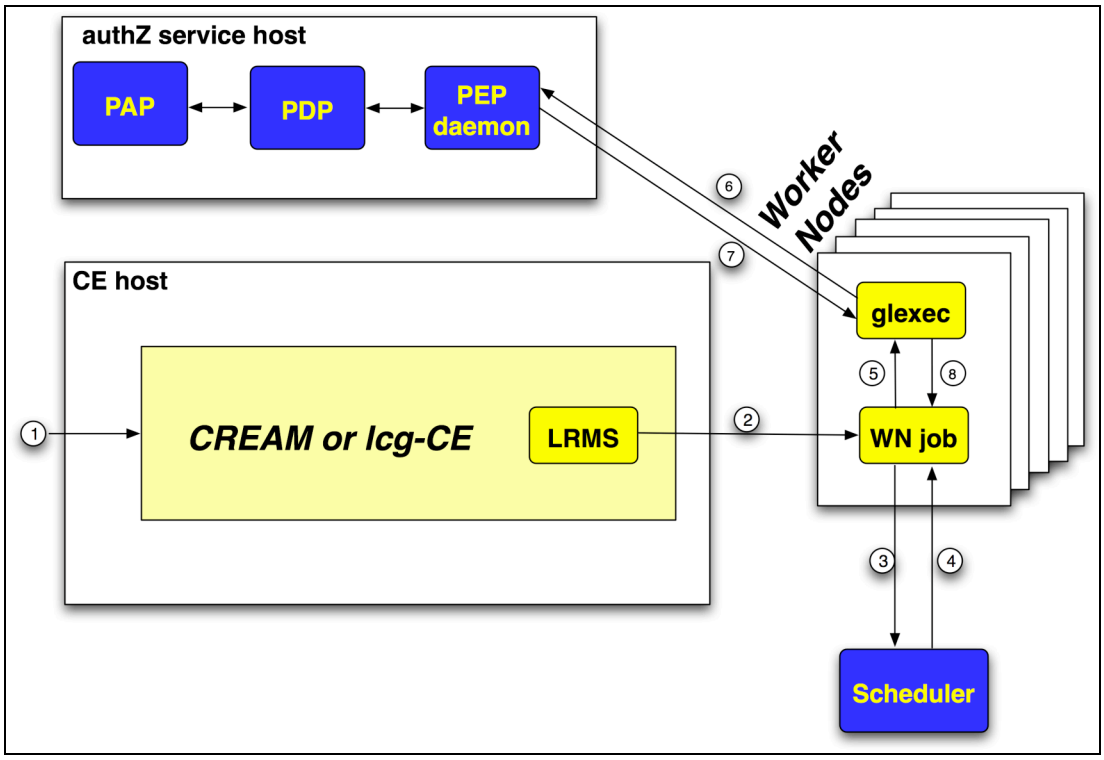

Figure 2 Step1: Adding glexec on the WN: The numbers correspond to a typical sequence of invocations during the job submission

Comments:

1. The only change on the WN is the installation of the new version of glexec. It calls out to the authorization service in order to obtain the authorization decision for the job in question. glexec receives as part of the authorization response the UID and list of GIDs, which it needs in order to perform the user switching.
2. There are two ways how glexec can be modified to call out to the authorization service:
    a. Direct integration into glexec: In this option, the client code (in the C programming language) is added directly to glexec. Its advantages are:
        i. Faster performance

ii. As the client code has been optimised in order to reduce dependencies on other software packages, this deployment option has the benefit that less code is installed and no dependencies are added on the WN.

b. Invocation of the PEP client through an LCMAPS plug-in. Its advantages are:

i. This option represents an incremental change of glexec, as LCMAPS is already integrated into it.

ii. The plug-in can be re-used in other services, where LCMAPS is installed.

Decision: Option b shall be used in the initial deployment. This means that the use of the authorization service is a configuration time option.
In the long term it is preferable to reduce the dependencies on the WN. The authorization service development team will provide performance numbers on these two options.

3. It is proposed that all three components of the service are installed on one host. This is the setup that will initially be supported by YAIM. However, it should be pointed out that this is only one way of installing the new authorization service, which has been designed to support several different deployment options in order to provide a high availability and fail-safe service with *no* single point of failure. Appendix A shows several other deployment options, which may suit larger sites. These setups will be supported from the beginning, and may require custom installations (i.e. not through YAIM).
4. The authorization service does not import remote policies at this stage. This means that ALL policies will be local and maintained by the site administrator.

Prerequisite: The pre-requisite for Step 1 is the successful certification of the service by SA3. The project team plans to support the following platforms: debian4, red hat enterprise 4/5, SL4/5 in 32 and 64 bit. The supported Java version is 1.5 and higher.

The **benefits** of this step are:

1. Centralized UID/GID mapping for WN through a service that is scalable, supports high-availability and does not require a shared file system (see appendix A for other deployment options).
2. Support for pilot jobs with an authorization with
   a. Proper Authorization
   b. Possible policy of matching submitter and invoker attributes

3 STEP 2: GRID-WIDE BANNING BY OSCT:

OSCT has raised the issue of a grid-wide, centralized banning list to JRA1. This requirement can be fulfilled by the authorization service as follows: OSCT operates one of the service components (called Policy Administration Point, PAP, [R2]), which contains a list of policies that describe, for example, the banned users, FQANs, VOs and CAs. This service will only be operated by trusted OSCT personnel.

Once this service is installed, every site can decide whether it wants to trust this banning service offered by OSCT. Initially, we propose that remote policies will be disabled by default. Site

administrators have to enable them if they choose to do so. In addition, local policies shall always overrule remote policies in case of a mismatch between them.

The only **prerequisite** of this banning service is that the authorization service is installed on sites and is used by the WN, i.e. step 1. Note that this step is independent of the work on the CE and/or WMS (see Figure 1).

The **advantages** of this step are:

1. OSCT offers a centralized banning list to the sites. The propagation time of banning thus reduces from weeks to hours. It is proposed to initially ban DNs, FQANs, VOs and CAs. It should be stressed that this requirement is based on past incidents that did take place in the EGEE/LCG infrastructure.
2. Every site is free to choose whether it wants to use this banning list or not.
3. It supports a migration scenario, where sites with and without banning are in use and experiences can be gained.
4. This step can be seen as a test-case[1] for supporting other remote policies based on
    a. National boundaries (e.g. NGIs)
    b. Regional boundaries (e.g. federations)
    c. Any group of small sites that co-operate closely and want to have a common policy.
    d. VOs can operate their own banning list, which sites may want to include
    e. VOs can include the banning list into their VO schedulers as well as other services

A site administrator can investigate the policy either through the command line tools or by importing and exporting them and thus verify their implications on the authorization on the site.

4 STEP 3: INTEGRATION INTO CREAM

Figure 3 shows the default setup of the service with CREAM.

[1] See Appendix B for a description of the use-case of hierarchical policies.

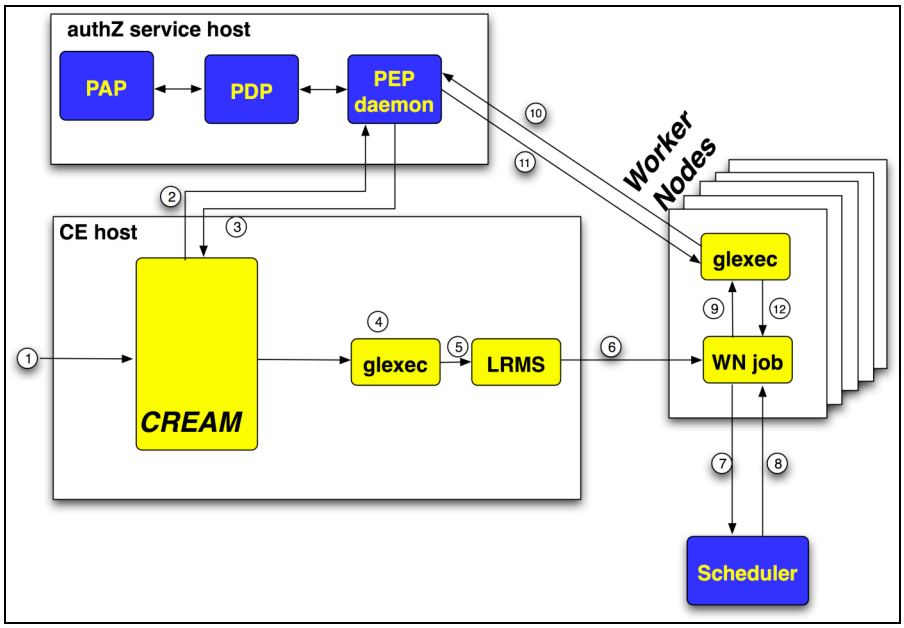

Figure 3 Step 2: Integration into CREAM

CREAM has various authorization steps that will need to be investigated in detail with the CREAM developers:

1. Delegation of proxy: Is there an authorization step?
2. Job registration: Currently call-out to gJAF
3. Create directory for the user job: Currently call-out to gLexec (?)
4. gridFTP: call-out to LCAS/LCMAPS through the gridftp plug-ins
5. Job start: call-out to gJAF
6. Blah invoking gLexec, which in turn invokes LCAS/LCMAPS
7. Callout to gJAF for the other operations on the job besides register and start, namely purge, status, suspend, resume.

Note:

1. It is not planned to integrate the authorization service into the lcg-CE.
2. There should[2] be only one call-out from CREAM to the service.
3. gLexec on the CE should not call the service directly, but should receive sufficient information from CREAM in order to authorize the user. This must be done in a secure manner. The details of this mechanism should be determined jointly with the CREAM and blah developers.
4. It is not foreseen to switch all CREAM CEs from the old to the new authorization scheme. This means that two node types will be supported: one with and one without the new authorization service (e.g. “CREAM” and “CREAM_AUTHZ”).

[2] It is preferred to have only one call-out. This has to be studied with the CREAM developers in detail.

Prerequisite: The service is installed on the site (i.e. phase 1).

The **benefits** of this stage are:

1. All the benefits of stage 1 and 2
2. Single authorization point for the CE.
   This includes easy centralized banning of user at CE through command line interface and easy evaluation of authorization policies for the CE through command line interface [R3].

5 STEP 4: INTEGRATION INTO WMS FOR AUTHORIZATION:

In this step the WMS uses the service in order to authorize the user at job submission time. This involves modifying the authorization decision within the WMS and has no effect on authorization decisions taken outside the WMS. Therefore this step can be done independent of the work on CREAM (see Figure 1).

Prerequisites: The first version of the service is available.

The **benefits** of this step are:

1. The same authorization mechanism is used in the WMS and the CE
2. Easy centralized banning of user at the WMS through command line interface [R3]. Unauthorized users are banned at an early stage, i.e. at submission time of the job.
3. Easy evaluation of authorization policies for the WMS through command line interface [R3]
4. Several WMS can share an authorization service.
5. Jobs from banned users can be stopped at submission time. This saves resources that would be used during the matchmaking process. It also stops the job from getting to the CE, which would just reject it. Furthermore it also prevents the CE from taking up resources to determine the job wouldn't run and gives more immediate feedback to the user.

6 STEP 5: INTEGRATION INTO WMS FOR MATCH-MAKING:

In this step, the service is being used by the WMS in order to evaluate whether the user is authorized on the CE. This step involves an interaction between WMS and CE. It is proposed that the WMS performs initially the match-making as it does today and then evaluates the policies of the chosen CEs as the very last step of the match-making. The advantage of this proposal is that there are only minimal changes in the match-making. In the longer term, i.e. outside the scope of EGEE-III, other matching-making algorithms can be developed and deployed as needed.

There are two possibilities how the WMS can evaluate the policies of the chosen CEs:

1. The WMS imports the policy of the CE. This requires that the sites agree to publish their policies[3].
2. The WMS calls-out to the authorization service at the site and receives the authorization decision. This requires that the sites agree access to the site authorization service from the outside.

Further input is needed from the developers of CREAM and WMS as well as other stakeholders to decide on the details of this step.

Comments:

1. JSPG and the sites should be involved in deciding between the two options a) and b). Technically, it would also be possible to support both simultaneously and let the sites choose individually which one they prefer.
2. The scaling properties of the two solutions may be very different. This may dictate the choice.
3. The BDII must contain the endpoint of the authorization service of every site. It is not planned to change any other information in the BDII (in particular the ACBRs) within the lifetime of EGEE-III.

Prerequisites: The service is installed on a subset of sites.

The **benefits** of this step is the consistent authorization between WMS and CE.

7 STEP 6: INTEGRATION INTO DATA MANAGEMENT

So far we have received conflicting statements with respect to the usefulness of the authorization service for the data management. Therefore this step requires further work.

However, it is stated that this service is not suited for supporting authorization decisions on large collections of individual items with immediate response times (e.g. ls command where the user's authorisation must be checked for each file). This is better done locally at the SE (e.g. through the mechanism of virtual UIDs and GIDs). But in our view this service is suited for authorizing command level access. There are several options implementing this such as

1. Call-out to the authorization service
2. Importing policies from the site's authorization service using a local PAP
3. Importing policies from the site's authorization service into an authorization mechanism specific for the SE.

The only **prerequisite** of this step is the installation of the authorization service at the local site, e.g. through step 1 if the site operates a CE.

[3] Note that sites only publish their public policies. A site can still choose to have a (typically) small set of private policies that are not published towards services outside the site.

The **advantages** of this step are:

1. A site administrator can control access to all SEs of his site through the same service with which the local CE(s) are controlled. Thus, there is one central authorization service per site which
    a. Reduces the administration effort and
    b. Guarantees consistent authorization at the site across all services (CE, SE).
2. Optionally, the SE can also benefit from centralized banning lists (see section 5).

APPENDIX A ALTERNATE DEPLOYMENT OPTIONS

It is planned to deploy the three components (PAP, PDP and PEP daemon) of the authorization service by default on one host. This is the configuration that will be supported by YAIM.

However, large sites may need to use another deployment model in order to take advantage of the scalability and high-availability features of the authorization service. In this appendix we describe the following possible options:

- Installing several independent instances of the basic setup of PEP daemon, PDP and PAP. The policies are synchronized between the different PAPs (see Figure 4) and the gridmap dir is synchronized between the different PEPs. The advantage of this setup is that the basic YAIM installation procedure can be used.
- Multiple instances of the PAP, PDP and PEP daemon running on multiple hosts (see figure 5): This setup allows dividing the load between the components on several hosts in a fail-safe manner. There is no single point of failure in this setup.

Note on the mapping between DN and UID/GID in a multi-host environment:
A PEP daemon plugin is responsible for maintaining the mapping of a user to a UID and set of GIDs. In order to avoid single-point-of-failure (SPoF) cases each Authorization Service, including the PEP daemon, may be deployed in a clustered fashion (i.e. multiple instances may be deployed and treated as a single logical instance). The current mechanism for persisting user to UID/GID tuples is to create filesystem hard links, treating the filesystem as a poor-man's database. This approach suffers from a couple problems, first it's slow since the filesystem must be accessed for each mapping operation. Second, in order to avoid SPoFs either a single filesystem (which must also be protected from SPoFs) must be shared between each PEP daemon deployment or the links must be replicated to each filesystem in some manner.

Therefore the PEP daemon uses a hybrid model for representing mapping data. The mapping table queried when a mapping operation is performed is stored in memory. This table is replicated across all nodes in the logical PEP daemon. One, or more, nodes also write out this information to the filesystem in the same hard-linked based mechanism (thus allowing any existing tools that expect this setup to continue to function). The library used to do this is very stable, serving as the clustering mechanism for hundreds of projects (e.g. JBoss, Tomcat, Shibboleth) for many years.

This mechanism has the following failure modes:

- Single node failure: When the node restarts it simply queries the cluster and pulls down the current mapping table.
- Total failure of all nodes. In this case when the nodes are restarted they reform the logical PEP daemon. When the node(s) that carry the on-disk mapping are started they restore this state.
- Network split (i.e. all nodes are still operational but a network issue prevents on group of nodes from seeing the others). In this case the isolated group of nodes begins to operate as its own logical PEP daemon. When the network issue is resolved the two groups of nodes will detect that their mapping tables are out of synch and will, first, try to merge the tables (this is possible if different users have not been mapped to the same UID). If there is an actual conflict there are two options. Either allow the conflict to continue and issue warnings or

remove one group of nodes from the logical PEP daemon. In either case manual intervention would be required to merge the data sets. It should also be noted that if any current mapping mechanism attempted to provide failover capability they would face the exact same problems as these problems are inherent in any multi-master replication system.

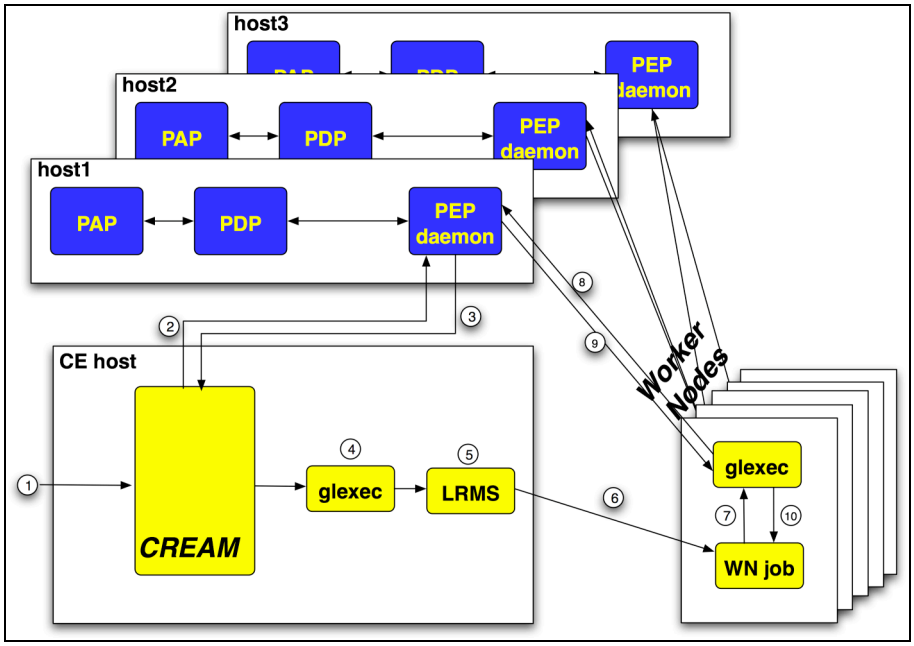

Figure 4 Multiple instances of the basic PAP-PDP-PEP daemon setup at a site

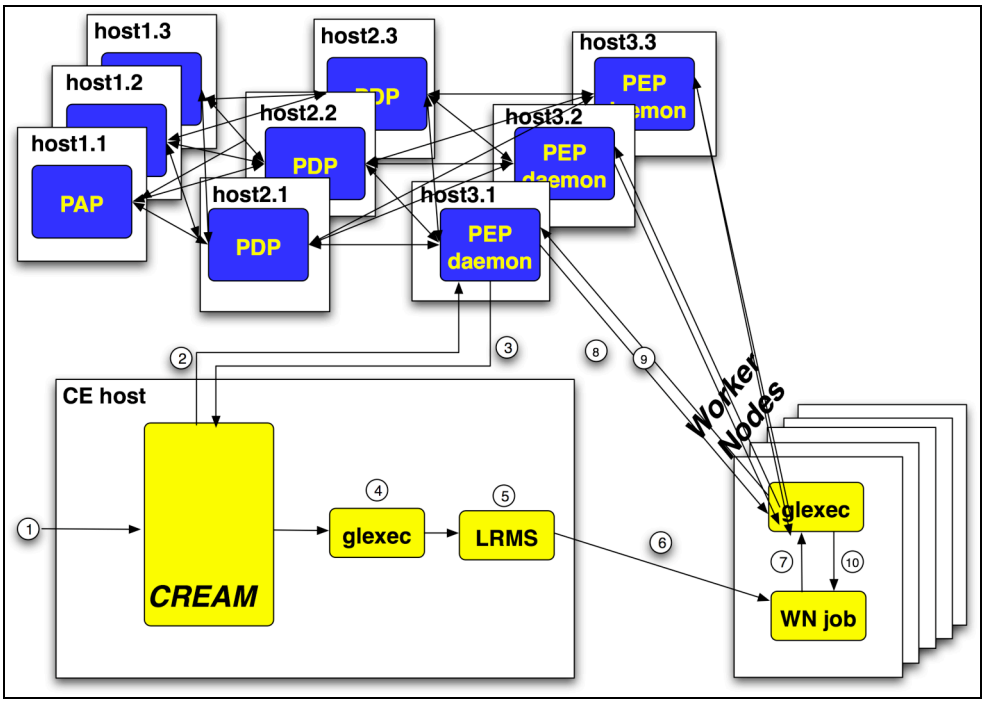

Figure 5 All components installed over several hosts in a fail-safe configuration

APPENDIX B HIERARCHICAL POLCIES AT A SITE

The authorization service can also be used to implement a hierarchical set of policies at a site. This use-case may be of interest to large sites operating several resources. Note that it does not involve importing policies from outside the site.

Figure 6 shows such a setup where several resources (CEs or SEs) are managed through two levels of policies:

1. A global site policy, where the site for example maintains a site wide ban list as well as a few policies on which users are authorized to submit which applications.
2. Special policies for groups of resources at the site, which may restrict or widen the global site authorization policies. For example, one group of resources could be part of the production infrastructure, the other part of a test-bed.

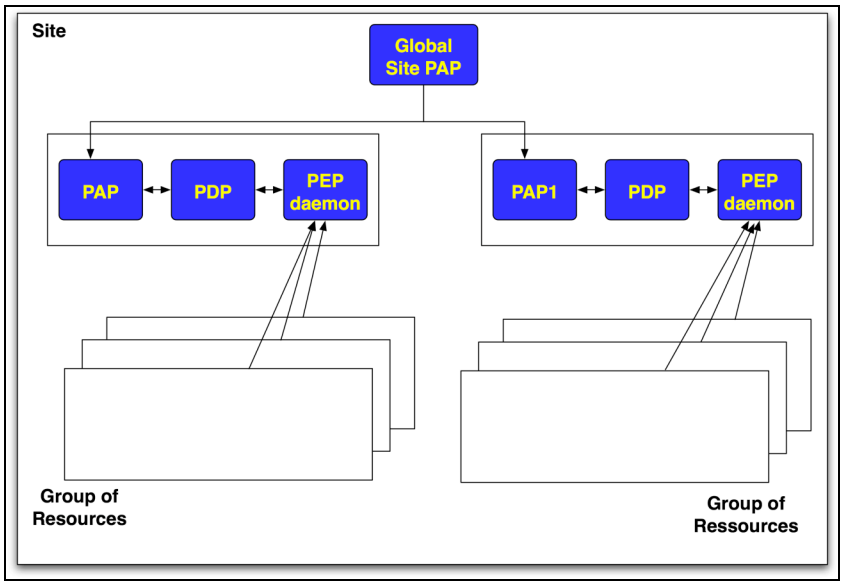

Figure 6 Administration of groups of resources

We see other uses of this deployment option within

- A group of sites, which trust each other and want to share authorization policies
- Policies formulated on a federation or NGI level