



Fachgebiet 3-4 – Aufbau einer AA-Infrastruktur
für das D-Grid

Use Cases for Authorization in Grid-Middleware

co-ordinators

Christian Grimm (grimm@rvs.uni-hannover.de)

Marcus Pattloch (pattloch@dfn.de)

D-Grid Integrationsprojekt (DGI)

Authors

Tobias Dussa (SSCK, RZ, Universität Karlsruhe)

Ursula Epting (Forschungszentrum Karlsruhe)

Bartol Filipovic (Fraunhofer-Institute SIT Darmstadt)

Gerti Foest (DFN-Verein)

Joachim Götze (ICSY, University of Kaiserslautern)

Christian Grimm (RRZN, Leibniz Universität Hannover)

Markus Hillenbrand (ICSY, University of Kaiserslautern)

Christian Kohlschütter (L3S Research Center, Leibniz Universität Hannover)

Rudolf Lohner (SSCK, RZ, Universität Karlsruhe)

Paul Müller (RHRK/ICSY, University of Kaiserslautern)

Marcus Pattloch (DFN-Verein)

Stefan Piger (RRZN, Leibniz Universität Hannover)

Tobias Straub (Fraunhofer-Institute SIT Darmstadt)

Jan Wiebelitz (RRZN, Leibniz Universität Hannover)

Das diesem Bericht zugrunde liegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01AK800B gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei den Autoren.

1.	Summary	2
2.	Authorization Measures by Users	3
2.1.	Authorization Measures on Credentials.....	3
2.1.1.	Authorizing Access to User Credentials.....	3
2.1.2.	Authorizing Proxy Renewal.....	4
2.1.3.	Delegation of Restricted Rights	4
2.2.	Authorization Measures in Data Services	6
2.2.1.	Authorizing Access to unencrypted Data.....	6
2.2.2.	Authorizing Access to encrypted Data	8
2.2.3.	Authorizing File Transfer Service	8
2.3.	Authorization Measures on Compute Services.....	9
2.3.1.	Authorizing Access to Jobs.....	9
2.3.2.	Authorizing Access to dedicated Resources	9
2.4.	Authorization Measures on Information Services	10
2.4.1.	Authorizing Access to Information about the Job Status	10
2.4.2.	Authorizing Access to Information about Executables, Input/Output Files.....	10
2.4.3.	Authorizing Access to Accounting Information	11
3.	Authorization Measures by Administrators	12
3.1.	Authorization Measures on Credentials.....	12
3.1.1.	Authorizing Proxy Renewal.....	12
3.1.2.	Delegation of Restricted Rights	12
3.2.	Authorization Measures on Data Services	13
3.2.1.	Authorization Measures to Files and Directories.....	13
3.2.2.	Authorizing Access to Storage Space (Quota).....	14
3.2.3.	Authorizing Access to Resources for Data Services.....	14
3.2.4.	Authorizing File Transfer Service	15
3.3.	Authorization Measures on Compute Services.....	16
3.3.1.	Restricting Access on Compute Resources	16
3.3.2.	Restricting Access to (available) Executables.....	16
3.3.3.	Restricting Execution of imported Executables	17
3.4.	Authorization Measures on Information Services	18
3.4.1.	Restricting Access to Information about Resources and Jobs.....	18
3.4.2.	Authorizing Access to Accounting Information	18
3.5.	Authorization Measures on VO Management.....	20
3.5.1.	Administer the VO Membership of Users and Resources.....	20
3.5.2.	Definition of Groups/Roles, etc. within the VO	20
3.5.3.	Authorizing Access to Sites/Resources for VOs/Groups/Users	21
4.	Overview.....	22
4.1.	Authorization Measures by Users	22
4.2.	Authorization Measures by Administrators.....	23

1. Summary

Security services in Grid environments, especially the authentication but also parts of the user authorization are largely based on PKI and X.509 certificates. In contrast to the well-established authentication infrastructures, the harmonization of authorization mechanisms as implemented in the examined Grid middlewares – Globus Toolkit 4, gLite and UNICORE – turns out to be a complex challenge.

As a preliminary work towards a common D-Grid wide AAI, a study of the different authorization approaches for common use cases was conducted. The aim of this work was to examine the feasibility of a common authorization infrastructure of these Grid middlewares. The use cases were defined from two different perspectives: authorization measures by users and by administrators. The regarded application areas were computing and information services as well as data and VO management. All results are based on practical experiences. Only the means supported by a standard deployment of the respective middleware have been considered, i.e. without additional (3rd party) software

The report confirms the predominant notion of the inhomogeneous and coarse grained authorization mechanisms in current Grid middlewares. Furthermore, we found that authorization functionality differs greatly especially between Globus Toolkit 4 and gLite on one side and UNICORE on the other side. This most prominently results from the current lack of the VO paradigm in UNICORE. Thus, a harmonization of the authorization infrastructure appears feasible on a high-level between the first two middlewares. The inclusion of UNICORE in such a concept requires substantial modifications of the middleware itself.

2. Authorization Measures by Users

2.1. Authorization Measures on Credentials

2.1.1. Authorizing Access to User Credentials

A user wants to authorize other users to access her user credentials. Third party's access to user credentials is necessary whenever a third party should interact with the users jobs (abort, retrieve results, retransmit).

2.1.1.1. Implementation in GT4

A user transmits a proxy credential to the delegation service (running within the same container as the service) via the command `globus-credential-delegate`. Corresponding services are now able to use the credential via the returned EPR (Endpoint Reference) (see Figure 1). Therefore not the Web service interface of the delegation service is used, ensuring that the EPR is only usable by local services.

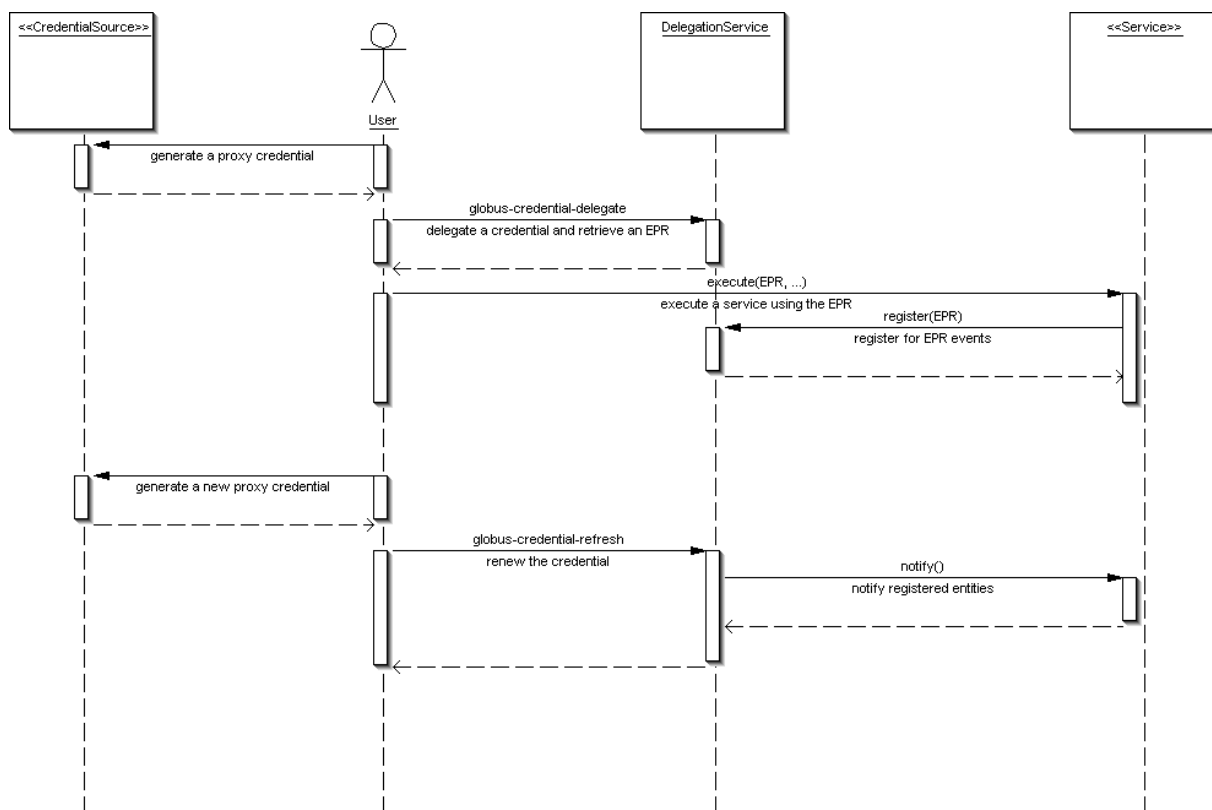


Figure 1: Submission and renewal of a proxy credential

Using the parameter “-d false” can restrict the delegation, i.e. the proxy credential cannot be used to start a new job.

If the environment allows the use of a MyProxy service a user can authorize access to her proxy credentials by authorizing access to credentials stored on the MyProxy credential service. By default access to the proxy credentials on the MyProxy service is not restricted, only protected by a passphrase. A user can restrict access to her proxy credentials to users specified by their distinguished name (DN) while storing the proxy credentials on the MyProxy server. This is done with the command:

```
myproxy-init -s <myproxy server> -xr <DN the of authorized third party>
```

To retrieve the proxy credentials authorized users have to know the passphrase protecting the private key on the MyProxy server.

2.1.1.2. Implementation in gLite

In gLite a user can authorize access to her proxy credentials by authorizing access to credentials stored on the MyProxy credential service. By default access to the proxy credentials on the MyProxy service is not restricted, only protected by a passphrase. A user can restrict access to her proxy credentials to users specified by their distinguished name (DN) while storing the proxy credentials on the MyProxy server. This is done with the command:

```
myproxy-init -s <myproxy server> -xr <DN the of authorized third party>
```

To retrieve the proxy credentials authorized users have to know the passphrase protecting the private key on the MyProxy server.

2.1.1.3. Implementation in UNICORE

Not supported by UNICORE. Job details are only available to the job owner and the administrator.

2.1.2. Authorizing Proxy Renewal

A user wants to authorize Grid services for renewal of proxy credentials. Proxy renewal is necessary if processing of the user's computing or data transfer jobs lasts longer than the validity of the submitted proxy credentials.

2.1.2.1. Implementation in GT4

Proxy Renewal can only be manually initiated by the user in order to extend the validity of the EPR at the delegation service.

If the environment allows the use of a MyProxy service the user has to deposit long-term proxy credentials on the MyProxy server. Proxy credentials derived from this long-term proxy credentials enable Grid services to act on behalf of the user. To allow proxy renewal, the user has to authorize Grid services to access proxy credentials stored on the MyProxy server. Authorized entities are nominated by entering their distinguished name with the command:

```
myproxy-init -d -s <myproxy server> -xR <DN of the authorized Grid service>
```

2.1.2.2. Implementation in gLite

Credential management in gLite is realized by the MyProxy service. To enable proxy renewal in gLite, the user has to deposit long-term proxy credentials on the MyProxy server. Proxy credentials derived from this long-term proxy credentials enable Grid services to act on behalf of the user. To allow proxy renewal, the user has to authorize Grid services to access proxy credentials stored on the MyProxy server. Authorized entities are nominated by entering their distinguished name with the command:

```
myproxy-init -d -s <myproxy server> -xR <DN of the authorized Grid service>
```

2.1.2.3. Implementation in UNICORE

Not explicitly supported by UNICORE.

2.1.3. Delegation of Restricted Rights

To prevent abuse a user wants to restrict the rights delegated with her proxy credentials to Grid services. By delegating only a subset of her rights the user limits the proxy credentials to the intended purpose of the job.

2.1.3.1. Implementation in GT4

The basic components of GT4 do not allow the delegation of restricted rights. However, if the GT4 VOMS Interceptor is used, a subset of VOMS attributes can be requested from the VOMS server and thus limit the credentials (cf. 2.1.3.2)

2.1.3.2. Implementation in gLite

gLite allows the user to delegate restricted rights to Grid services by submitting a defined subset of VO attributes within the proxy certificate. This subset of VO attributes is requested from the VOMS server while the user generates her proxy certificate. The user specifies the subset of VO attributes with the command:

```
voms-proxy-init -voms <myVO:/myGroup/Role=myRole>
```

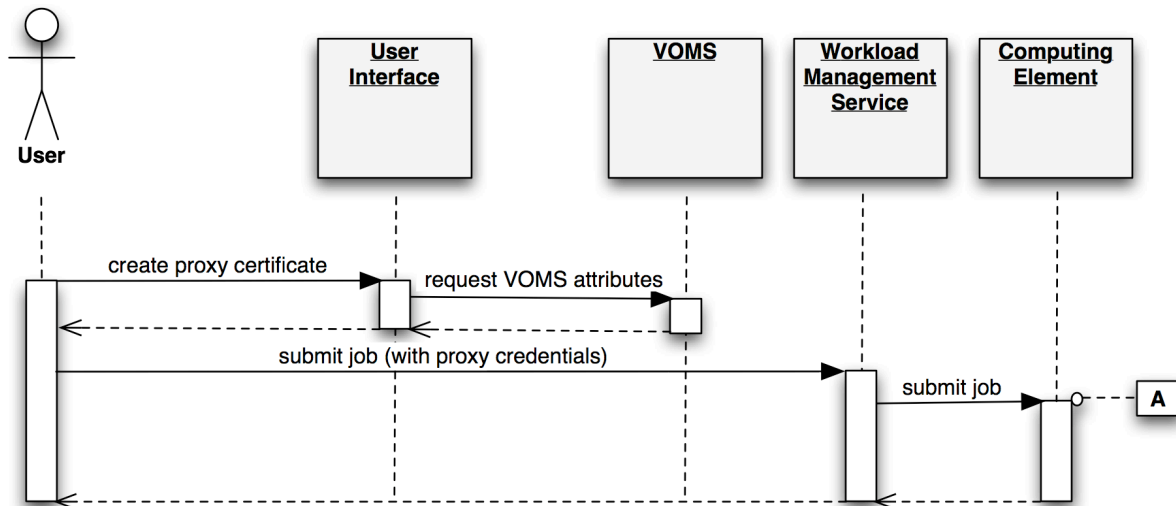


Figure 2: Sequence diagram for delegating proxy credentials with restricted rights

As a result of the authorization decision, the user is mapped to a pool account on the Grid resources (see Figure 2, step A). These pool accounts can have different authorizations on operating system level. This is out-of-scope of the gLite middleware.

To delegate more tightly specified rights in proxy certificates the user has to be able to specify a policy that is integrated in the proxy certificate. RFC 3820 style proxy certificates include a certificate extension where such a policy can be integrated. Support for this kind of proxy certificates is not implemented in gLite.

2.1.3.3. Implementation in UNICORE

The delegation of rights is conveyed explicitly by the job owner at the time when the job is being defined (Consignor/Endorser Model). A dynamic delegation of rights and the accompanied requirement to restrict the delegation is only planned for the successor of UNICORE, UniGridS.

Moreover, in the GRIP project, a plug-in has been developed to issue Globus-conforming proxy certificates, in order to delegate rights to Globus components. However, still there is no specification how to explicitly restrict rights within these certificates.

2.2. Authorization Measures in Data Services

2.2.1. Authorizing Access to unencrypted Data

A user wants to authorize access to her unencrypted directories and files stored on Grid resources. The decision applies to the following groups:

1. Other users in the same VO
2. Users from other VOs
3. Grid services

2.2.1.1. Implementation in GT4

A user wants to store data at a data service. In order to authorize access to this data, CAS (Community Authorization Service) can be used. This allows the authorization of access to the corresponding GridFTP-Server. This does not include direct access via the Unix file system.

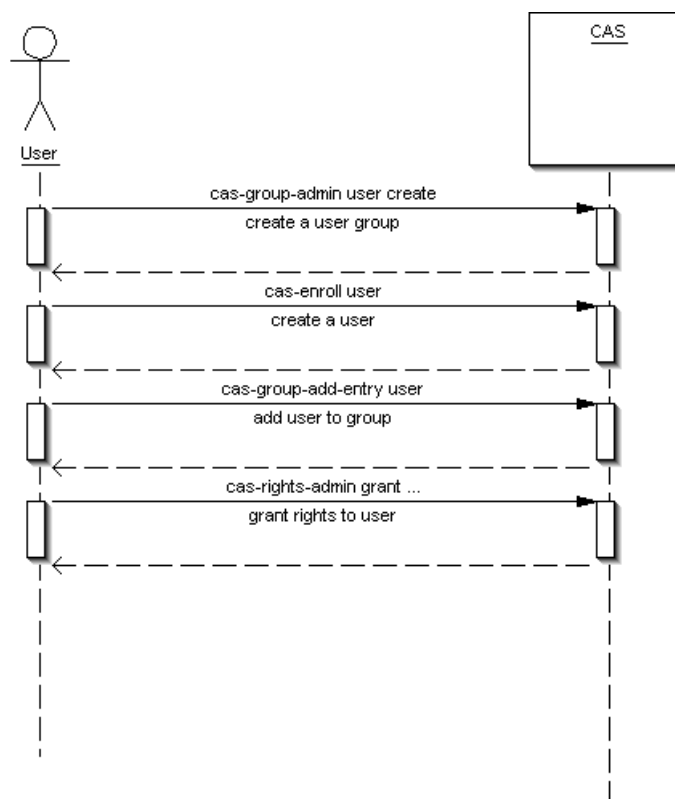


Figure 3: Authorizing data access with CAS

The procedure of using CAS with the precondition that the resource including the service type is already established (see Figure 3):

1. Creating a new user group:
cas-group-admin user create *user_group user_group*
2. Creating a new user:
cas-enroll user *user_group user_name user_subjectDN trust_anchor*
trust_anchor = symbolic identifier for CA
3. Adding the new user to the user group:
cas-group-add-entry user *user_group user*
4. Assign authorization for an action to a user group.
cas-rights-admin grant *user_group objectGroup object serviceAction serviceType_name action*
action = read/write – authorization

serviceType_name = typically a distinction between file, directory, etc. is made
object = symbolic identifier for a path on a GridFTP-Server

2.2.1.2. Implementation in gLite

In gLite the File Catalog Service handles the administration of directories and files. The File Catalog Service maps the logical file name (LFN/GUID) to physical files (SURL) and administers the access rights to these objects. A user gets access to files stored on the gLite Storage Element (SE) by using the gLite I/O service. Authorization is done through the *File Authorization Service*.

In gLite several Catalog Services are in use. In this section data access considering the *File and Replica Manager (Fireman)* is described exemplarily.

Access rights to directories or files in Fireman can be set by the commands „glite-catalog-chmod“ and „glite-catalog-setacl“.

1. The command „glite-catalog-chmod“ changes the permissions of a file or a directory. The gLite implementation provides the following permissions for users, groups or others:
 - p: allow to change permissions
 - d: delete the entry
 - r: read the file
 - w: write the file
 - l: list contents
 - x: execute
 - g: get the metadata of the file
 - s: set the metadata of the file
2. The command „glite-catalog-setacl“ manages Access Control Lists (ACL) for directories and files. Within these ACLs the user can grant access for users and services specified by the DN in the subject of the certificate or groups specified by VOMS attributes (VO membership, groups) included in the certificate.

2.2.1.3. Implementation in UNICORE

UNICORE allows storing data into three different storage „spaces“:

Uspace: Temporary storage on a Usite/Vsite which is being cleared after job execution

Xspace : Permanent storage on a Usite/Viste (e.g., archive), usually an arbitrary directory on the system, relative from the file system root.

Nspace: Storage at the client's side (the client must be online during transmission)

In order to access these different storage spaces, the user must possess proper access rights (e.g., a valid certificate for the corresponding Vsite). The attribution of access rights on file-level is performed via the filing system (e.g., by *chown/chmod*), and thus requires detailed knowledge about the affected Usite and its mapping from certificates (persons, roles, groups/VOs) to UNIX accounts/groups (Xlogin; via the UNICORE user database, UUDB) as well as the exact directory paths to the storage spaces (e.g., archive mount point). It is therefore difficult to draw a line between the “Grid platform” and the underlying system environment (Unix/Linux).

Before a job is terminated, all relevant data has to be explicitly copied by the user from the job's Uspace to another space – otherwise it will automatically be deleted. For a collaborative use, the Xspace is recommended, since the Nspace is volatile (it is not guaranteed that the client is reachable all the time). Here, for instance, the files can be stored in the home directory of the corresponding local UNIX user.

Per default (via *umask*), all data in the Xspace is readable by everyone, but only writable by the owner's UNIX account (Xlogin). If another user is allowed to *chdir* into the same Xspace, he might be able to at least read the data. Further, if more than one certificate is mapped to the UNIX account, all certificate owners may also have write access to that file. In this case, the system cannot make a distinction between them.

2.2.2. Authorizing Access to encrypted Data

Access to decrypted data requires authorization to access the encrypted file and authorization to access the symmetric key used for encrypting the data. Authorizing access to files is described in section 2.2.1. A user wants to authorize access to the unencrypted content of encrypted files.

2.2.2.1. Implementation in GT4

GT4 does not include an encryption service.

2.2.2.2. Implementation in gLite

In gLite the *Hydra Keystore* serves as a repository for symmetric keys to enable file decryption on the storage element.

Changing access rights for symmetric keys stored on the Hydra Keystore is done via the commandline interface is implemented with the commands „`glite-eds-chmod`“ and „`glite-eds-setacl`“.

2.2.2.3. Implementation in UNICORE

Not explicitly supported by UNICORE, OS/File-level authorization mechanisms apply.

2.2.3. Authorizing File Transfer Service

A user wants to authorize other users and services to initiate file transfers between Storage Elements or between GridFTP-Servers.

2.2.3.1. Implementation in GT4

Storage elements are not available in GT4. As for GridFTP, the initiator of the file transfer is acting like a normal Grid-FTP client and therefore only the actions mentioned in section 1.1.2.1 are at the disposal of client and server. Normally, a user would use the RFT service or the GridFTP client in order to transmit data between two GridFTP servers. As the service acts with the user's permissions, the suitable authorization entries have to be made in CAS.

2.2.3.2. Implementation in gLite

The File Transfer (FTS) and File Placement Service (FPS) differentiate between three user roles, the Submitter User, Regular and Vetoed User. These types of users offer the following capabilities:

Vetoed User:

- User is banned from using the FTS/FPS

Regular User (owning job) is allowed to

- cancel the job
- get the status of file transfers in the job
- get the status of the job
- get the summary of the job

Submitter User has all capabilities of the Regular User and is additionally allowed to

- submit a new job

The gLite FTS/FPS allows users only to access channels they are configured to use by the FTS/FPS administrator. There are no user configurable authorization mechanisms in this service.

2.2.3.3. Implementation in UNICORE

Not explicitly supported by UNICORE. If a user can access a file, he may transfer it on his behalf.

2.3. Authorization Measures on Compute Services

2.3.1. Authorizing Access to Jobs

A user wants to authorize access to her jobs for other users. This is necessary whenever this third party should interact with the users jobs (abort, retrieve results, retransmit).

2.3.1.1. Implementation in GT4

Only the user can access his/her jobs. The basic components of GT4 do not allow the authorization of other users to access a job that is not his/her own.

2.3.1.2. Implementation in gLite

gLite implements no such functionality.

2.3.1.3. Implementation in UNICORE

Job details are only available to the job owner and the administrator.

2.3.2. Authorizing Access to dedicated Resources

A user wants to authorize access to dedicated compute resources. Typically authorization decisions on Grid services or resources are in the responsibility of the resource administrators. Therefore regarding compute services the authorization decision a user can make affects only job submission.

2.3.2.1. Implementation in GT4

A user cannot manipulate the authorization to access dedicated resources. If the GT4 VOMS Interceptor is in use, the user can restrict his proxy credentials according to 2.1.3.

2.3.2.2. Implementation in gLite

In glite a user can restrict his proxy credentials (see 2.1.3) to the membership attributes provided by VOMS. This can implicitly limit the usable resources. An explicit way to restrict proxy credentials to specific resources is not implemented.

2.3.2.3. Implementation in UNICORE

See Section 2.2.1.3, as UNICORE does not make a distinction between data and compute services (both are Vsites).

2.4. Authorization Measures on Information Services

A user wants to authorize other users or services to access information about her job. This information may contain:

1. State of the job, (*submitted, running, done, failed*, etc.)
2. Content of the job, e.g. used executables, input/output-files
3. Accounting Information

Access to the following entities can be authorized:

1. Grid services, involved in the job execution
2. Grid services, not involved in the job execution
3. Other users

2.4.1. Authorizing Access to Information about the Job Status

A user wants to authorize other users or services to access Information about the status of her job.

2.4.1.1. Implementation in GT4

The status of a job can only be access by the owner of the job.

2.4.1.2. Implementation in gLite

The *Logging & Bookkeeping* (L&B) service is the central service in gLite for keeping track of all states of a job. In the default configuration only the user or services involved in the job's execution can access this information. The user can authorize other users to access information about the job by installing an ACL that contains the DN of the authorized user certificates. The DN is submitted to the logging and bookkeeping service with the command:

```
glite-lb-logevent -e ChangeACL -s UserInterface -p --permission 1 -j <job ID>  
--user_id <DN of the authorized user> --user_id_type 0 --permission_type 0 --  
operation 0
```

Access to information about the job status can be authorized for VOs and subsets of them by VOMS attributes in the ACL. Read access is the only implemented permission in gLite so far.

In gLite the Relational Grid Monitoring Architecture (R-GMA) service provides information about available resources in a VO. Access to this information is granted to everybody who has a valid certificate issued CA which is present in */etc/grid-security/certificates*.

2.4.1.3. Implementation in UNICORE

Status information is only available to the job owner and the administrator, via the Network Job Supervisor (NJS).

2.4.2. Authorizing Access to Information about Executables, Input/Output Files

A user wants to authorize other users or services not directly involved in job processing to access information about executables and/or input/output files of his job.

2.4.2.1. Implementation in GT4

This information is not provided by GT4, OS authorization mechanisms apply.

2.4.2.2. Implementation in gLite

Fine-grained authorization of access to information about executables and/or input/output files is not implemented in gLite. But as described in chapter 2.4.1 access to the complete set of information about a job can be authorized.

2.4.2.3. Implementation in UNICORE

Not explicitly supported by UNICORE, OS authorization mechanisms apply.

2.4.3. Authorizing Access to Accounting Information

A user wants to authorize other users and services to access his accounting information.

2.4.3.1. Implementation in GT4

Accounting information is not provided by GT4.

2.4.3.2. Implementation in gLite

DGAS is the accounting component in gLite. The DGAS accounting service accumulates information about the usage of Grid resources by the users and by groups of users, including Virtual Organizations as groups of users. Grid resources send accounting information to the so-called Home Location Register (HLR).

Authorization to access the accounting data is checked by the HLR server. Client access requires a valid user proxy certificate. Regular Grid users can retrieve only information related to their own HLR account. Users that are mapped as HLR administrators can access information about all user and resource accounts.

A user can authorize other users or services access to his accounting information only by conceding access to his credentials as described in 2.1.

2.4.3.3. Implementation in UNICORE

Not explicitly supported by UNICORE, OS authorization mechanisms apply.

3. Authorization Measures by Administrators

3.1. Authorization Measures on Credentials

3.1.1. Authorizing Proxy Renewal

An administrator wants to authorize dedicated Grid services to renew proxy credentials. Proxy renewal is necessary if processing of the users job, retransmitting the job or data transfer lasts longer than the validity of the submitted proxy credentials. Therefore the user has to deposit long-term proxy credentials on a credential server to enable proxy renewal. Proxy credentials derived from this long-term proxy credentials enable Grid services to act on behalf of the user. To allow proxy renewal, the user has to authorize Grid services access to the proxy credentials stored on the credential server.

3.1.1.1. Implementation in GT4

MyProxy is built and installed as part of a default GT 4.0 installation and it can be used for proxy renewal. The MyProxy administrator controls, who can store, modify, retrieve, and remove credentials by configuring the myproxy-server.config file accordingly. The configuration file can have amongst others the *authorized_renewers "DN regex"* type of line (regex is a limited regular expression for matching the distinguished names (DNs) of classes of users).

Users can place additional access controls on credentials when they store them with the command myproxy-init. The command has options to support credential renewal. The *-R* argument to myproxy-init configures the credential for renewal by the specified service. Renewal requires two authentications. The renewing service must authenticate with its own credentials, matching the distinguished name specified by the *-R* argument, and must also authenticate with an existing credential that matches the distinguished name of the stored credential, to retrieve a new credential.

3.1.1.2. Implementation in gLite

An administrator can set default entities during the configuration of gLite that are allowed to renew proxy credentials.

3.1.1.3. Implementation with UNICORE

The concept of proxy certificates is unknown in UNICORE. Therefore, there is no way to authorize the renewal of proxy certificates.

3.1.2. Delegation of Restricted Rights

To prevent abuse an administrator wants to restrict the rights of proxy credentials delegated to Grid services.

3.1.2.1. Implementation in GT4

Not explicitly supported.

3.1.2.2. Implementation in gLite

Administratively measures to restrict rights in proxy credentials are not implemented in the gLite middleware.

3.1.2.3. Implementation with UNICORE

The concept of privilege delegation is unknown in UNICORE. Therefore, there is no way to control the delegation of privileges.

3.2. Authorization Measures on Data Services

3.2.1. Authorization Measures to Files and Directories

An administrator wants to authorize access to dedicated files on the data services.

3.2.1.1. Implementation in GT4

Analogue to the description in section 2.2.1.1 the Community Authorization Service (CAS) could be used. This allows the authorization of access to the corresponding GridFTP-Server. This does not include direct access via the UNIX file system.

The procedure of using CAS with the precondition that the resource including the service type is already established (see Figure 1):

1. Creating a new user group:
cas-group-admin user create user_group user_group
2. Creating a new user:
cas-enroll user user_group user_name user_subjectDN trust_anchor
trust_anchor = symbolic identifier for CA
3. Adding the new user to the user group:
cas-group-add-entry user user_group user
4. Assign authorization for an action to a user group.
cas-rights-admin grant user_group objectGroup object serviceAction
serviceType_name action
action = read/write – authorization
serviceType_name = typically a distinction between file, directory, etc. is made
object = symbolic identifier for a path on a GridFTP-Server

The Administrator of the CAS server adds in step 4 information about the kinds of rights that can be granted for these objects. These are stored as service types and relevant actions are mapped to these service types. The kinds of service types that the Administrator can add are file, directory and so on. To do so the cas-enroll client with the serviceType option may be used. To add a service type called *file* and give members of *user_group* all rights on this service type the Administrator uses the command `cas-enroll serviceType user_group file`.

In some cases required authorization mechanisms can be described with security descriptors:

Security descriptors contain various security properties like credentials, the grid-mapfile location, required authentication and authorization mechanisms and so on. There are four types of security descriptors in the code base for setting container, service, resource and client security properties:

- Container security descriptor: determines the container level security requirement that needs to be enforced.
- Service security descriptor: determines the service level security requirement that needs to be enforced.
- Resource security descriptor: determines the resource level security requirement that needs to be enforced.
- Client security descriptor: determines the security properties that need to be used for a particular invocation.

Each of these is represented as an object and can be altered at runtime. Service and container security descriptors can be configured as XML files in the deployment descriptor. Resource security descriptors can only be created dynamically, either at runtime or from a descriptor file. Client security descriptor can be configured as a XML file and set as property on Stub. If the security descriptor file is altered at runtime, it will not be reloaded. Details on how to configure authorization mechanisms are described in the GT4 documentation¹.

¹ “3.5. Configuring authorization mechanisms”,
http://www.globus.org/toolkit/docs/4.0/security/authzframe/security_descriptor.html

Services using security descriptors are for example:

RFT (Reliable File Transfer): Required authorization mechanisms are described with security descriptors (type: resource security descriptors). RFT uses GridFTP for the file transfer and therefore CAS and the Unix resp. file system access rights for authorization.

RLS (Replica Location Service): Required authorization mechanisms are described with security descriptors (type: resource security descriptors).

3.2.1.2. Implementation in gLite

In gLite administrators can define authorization rules to access data services during the installation and configuration of the gLite middleware. Access rights can be set for each object in the file catalog for user, groups and others.

3.2.1.3. Implementation with UNICORE

With UNICORE, there are no dedicated data services. Therefore, there are also no explicit means to control access rights. Access rights are implemented by the underlying file system, usually a UNIX file system.

Case 1: Setting access rights of VOs

The concept of VOs is unknown to UNICORE. Thus, it is not possible to set access rights of VOs.

Case 2: Setting access rights of groups

Groups' rights to access certain data are controlled with the help of the UNIX file system access control. The administrator has full power to specify access rights to files/directories with the chmod command.

Case 3: Setting access rights of users

This case is analogous to case 2.

3.2.2. Authorizing Access to Storage Space (Quota)

An administrator wants to limit the storage space for users and groups.

3.2.2.1. Implementation in GT4

Not explicitly supported, OS authorization mechanisms apply..

3.2.2.2. Implementation in gLite

Currently dealing with quotas on storage elements is not implemented in the gLite middleware. It can only be controlled via file-system level functions.

3.2.2.3. Implementation with UNICORE

User-/group-specific access rights are not controlled by UNICORE. This is done by the underlying file system, usually a UNIX file system.

Case 1:

If the file system that UNICORE runs on has support for quotas, the administrator is able to authorize access with the usual means for user/group quota control.

Case 2:

The administrator can define the maximum storage usage granted to every job with regard to storage space defined within UNICORE at configuration time. To do so, the appropriate variables within the storage space definition must be configured accordingly. This scheme, however, does not provide a means to actually enforce the limits specified. It is expected that the actual resource usage is specified correctly in the UNICORE jobs and that resource usage checking is done by the UNICORE client. If any one of these conditions fails to be met, the specified limits are not enforced.

3.2.3. Authorizing Access to Resources for Data Services

An administrator wants to authorize access to data services on his site.

3.2.3.1. Implementation in GT4

Resource security descriptor may be used.

3.2.3.2. Implementation in gLite

In gLite an administrator can grant VOs access to resources on his site during the configuration. Access can be granted to whole VOs but not to separate groups or users within a VO.

3.2.3.3. Implementation with UNICORE

In UNICORE, data services, file transfer services and compute services are not distinct concepts; Vsites are used for all. Therefore, granting access to a data service is exactly the same as granting access to a file transfer service, a compute service, or, more generally speaking, a Vsite.

3.2.4. Authorizing File Transfer Service

The File Transfer Service uses dedicated, unidirectional channels to transfer files from one Storage Elements to another. An administrator wants to authorize users and service to use the data transfer service.

3.2.4.1. Implementation in GT4

Refer to section 3.2.1.1. Additionally, if the GT4 VOMS Interceptor is in use, the RFT service administrator can use VOMS attributes to control service access.

3.2.4.2. Implementation in gLite

The gLite FTS/FPS differentiates between three kinds of administrators, the administrator of the service, the Channel Manager and the VO Manager.

The administrator of the FTS/FPS creates and removes channels between site's SEs. She adds the DN of users that are allowed to submit jobs (Submitter Users) to the FTS/FPS (transferring files) to the submit-mapfile or specifies VOMS attributes that are given to users that shall have the status of a Submitter User. Furthermore, she can exclude users from access to the FTS/FPS by adding the respective DNs. to a veto-mapfile.

The Channel Manager configures the parameters of the channel itself, like data bandwidth or state (up, down) and adds/removes other users as Channel Administrators.

The last type of administrator, the VO administrator creates and removes administrators and users by giving them the respective VOMS attributes. Additionally, the VO manager has access to the jobs of the users of the same VO and can thus list jobs, get information about jobs and cancel jobs.

3.2.4.3. Implementation with UNICORE

In UNICORE, file transfer services, data services and compute services are not distinct concepts; Vsites are used for all. Therefore, granting access to a file transfer service is exactly the same as granting access to a data service, a compute service, or, more generally speaking, a Vsite.

3.3. Authorization Measures on Compute Services

3.3.1. Restricting Access on Compute Resources

An administrator wants to authorize users to access compute resources. Furthermore, she wants to limit the amount of a resource (CPU/RAM) these users can access.

3.3.1.1. Implementation in GT4

The limitation of the amount of resource usage is not explicitly supported. Note: GT4 uses the Grid Resource Allocation and Management (GRAM) interface as a basic mechanism for the purposes of coordination of remote computations. The GT4 GRAM server is typically deployed in conjunction with the Delegation and RFT service to address data staging, delegation of proxy credentials, and computation monitoring and management in an integrated manner. GRAM can cooperate with different schedulers which may have different capabilities. Because of this multipurpose approach there is no general mechanism with GRAM to limit the amount of a resource an user can use.

As for the access to the resource itself, the normal security considerations apply (a grid-mapfile, a Security Descriptor, and – if the GT4 VOMS Interceptor is in use – VOMS attributes can be used to authorize access to compute resources.

3.3.1.2. Implementation in gLite

In gLite access to resources can be configured administratively by binding compute resources to specific queues of the batch system on the Compute Element (CE). Access to these queues is granted to specific UNIX groups to which pool accounts belong. The mapping to these pool accounts on the CE is done by LCMAPS. LCMAPS allows the mapping to different pool accounts in dependence of VOMS groups, roles, and capabilities.

3.3.1.3. Implementation with UNICORE

User-/group-specific access rights are not controlled by UNICORE. This is done by the underlying batch/operating systems.

Case 1:

If the underlying batch/operating systems have support for limiting access to CPUs and/or RAM, the administrator can authorize resource access with the means provided by these mechanisms.

Case 2:

The administrator can specify the maximum resource usage of every job at Vsite configuration time. To do so, the appropriate variables must be set accordingly in the Vsite configuration. To do so, the appropriate variables within the Vsite definition must be configured accordingly. This scheme, however, does not provide a means to actually enforce the limits specified. It is expected that the actual resource usage is specified correctly in the UNICORE jobs and that resource usage checking is done by the UNICORE client. If any one of these conditions fails to be met, the specified limits are not enforced.

3.3.2. Restricting Access to (available) Executables

An administrator wants to restrict access to and usage of executables installed on a resource.

3.3.2.1. Implementation in GT4

Not explicitly supported, OS authorization mechanisms apply.

3.3.2.2. Implementation in gLite

The restriction of the access and execution of installed applications on resources is not handled by glite. In gLite the responsibility for job processing ends on the Compute Element that enqueues the job in a batch system.

3.3.2.3. Implementation with UNICORE

User-/group-specific access rights are not controlled by UNICORE. This is done by the underlying batch/operating systems.

Case 1:

If the underlying batch/operating systems have support for limiting access to installed applications, the administrator can authorize application access with the means provided by these mechanisms. Also consider the possibility to use license managers to provide access control.

Case 2:

The administrator needs advertise the installed applications at Vsite configuration time. If an application is not defined within the Vsite definition, then it is not accessible through the standard UNICORE interface.

Consider, however, that in general it is possible and permitted for the users to execute arbitrary applications either via shell scripts or direct

3.3.3. Restricting Execution of imported Executables

An administrator wants to authorize the execution of applications contained in the job.

3.3.3.1. Implementation in GT4

Restricting the execution of applications on resources is not in the scope of GT4, but OS authorization mechanisms apply (like noexec mount)..

3.3.3.2. Implementation in gLite

Restricting the execution of applications on resources is not in the scope of gLite. In gLite the responsibility for job processing ends on the Computing Element that enqueues the job in a batch system.

3.3.3.3. Implementation with UNICORE

User-/group-specific execution privileges are not controlled by UNICORE. This is done by the underlying file/batch/operating systems.

Case 1:

If it is possible to limit the execution of self-installed software with the help of the underlying file/batch/operating systems, then the administrator can make use of this possibility. As an example, it may be feasible to have the users' home directories mounted in noexec mode, so that no software that resides in any such home directory may be executed.

Case 2:

The administrator can directly define the applications that are available to users in the Vsite configuration. Setting the appropriate configuration parameters accordingly prohibits the users from running arbitrary software.

3.4. Authorization Measures on Information Services

3.4.1. Restricting Access to Information about Resources and Jobs

An administrator wants to authorize users or services to access information about resources or jobs.

3.4.1.1. Implementation in GT4

The Grid Resource Allocation and Management (GRAM) mechanism in GT4 can provide information about the status of jobs (execution management and monitoring of remote computations). GRAM is not a resource scheduler, but rather a protocol engine for communicating with a range of different local resource schedulers using a standard message format. WS GRAM (Web Services GRAM) requires that the `sudo` command is installed and functioning on the service host where WS GRAM software will execute. Authorization rules will need to be added to the `sudoers` file to allow the WS GRAM service account to execute (without a password) the scheduler adapter in the accounts of authorized GRAM users. To protect users from each other, jobs are executed in appropriate local security contexts, e.g. under specific UNIX user IDs based on details of the job request and authorization policies. In the default configuration only the user or services involved in the job's execution can access this information.

In GT4 the Monitoring and Discovery System (known as MDS4 or WS MDS) provides information about resources in the Grid environment. It includes the WSRF-based *Index Service*, which collects data from various sources and provides a query/subscription interface to that data. In addition to the Index Service, MDS4 includes an Aggregator Framework which can be used to build services that collect and aggregate data from an aggregator source and sends that data to an aggregator sink for processing. This information can be accessed with the `wsrf-query` command or with WebMDS using a standard web browser. GT4 does not provide authorization mechanisms to regulate access to the information.

3.4.1.2. Implementation in gLite

The *Logging & Bookkeeping* (L&B) service is the central service in gLite for keeping track of all states of a job execution. In the default configuration only the user or services involved in the job's execution can access this information. There are no means to extend the ACLs for other users or services for administrators.

In gLite the Relational Grid Monitoring Architecture (R-GMA) Service provides information about available resources in Grid environment. Access to this information is granted to everybody who has a valid certificate issued by a trusted CA. For this the CA's root certificate has to be present in `/etc/grid-security/certificates`. There are no means to restrict access to information about available resources for administrators.

3.4.1.3. Implementation with UNICORE

Access rights are not controlled by UNICORE.

Case 1:

All users who log in with the same certificate have access to all information about all their jobs. Furthermore, all users whose certificates are mapped to the same Xlogin by the UUDB can find out plenty of information about jobs belonging to their Xlogin.

Case 2:

In general, information about a Vsite can only be granted to all users that have access to the Vsite.

3.4.2. Authorizing Access to Accounting Information

An administrator wants to authorize users or services to access accounting information.

3.4.2.1. Implementation in GT4

No particular accounting information is provided by standard GT4 installation. (Note: The SweGrid Accounting System (SGAS) is a new tech preview since GT 4.0.1 that may provide accounting information).

3.4.2.2. Implementation in gLite

DGAS is the accounting component in gLite. The DGAS accounting service accumulates information about the usage of Grid resources by the users and by groups of users, including Virtual Organizations as groups of users. Grid resources send accounting information to the so-called Home Location Register (HLR)

Authorization to access the accounting data is checked by the HLR server. Client access requires a valid user proxy certificate. Regular Grid users can retrieve only information related to their own HLR account. Users that are mapped as HLR administrators can access information about all user and resource accounts.

An administrator can empower users to administrator level with the following command:

```
glite_dgas_hlrAddHlrAdmin -a <DN of the user>
```

3.4.2.3. Implementation with UNICORE:

No particular accounting information is provided by UNICORE. Thus, there is no access to be authorized. In any case, if underlying mechanisms like batch systems or operating systems provide accounting information, it is up to those layers to authorize access to this information.

3.5. Authorization Measures on VO Management

A VO administrator is responsible for the implementation of policies for a VO. The responsibilities are:

1. Create/delete VO
2. Create/delete groups, roles and capabilities within the VO
3. Administer VO Membership

An administrator may delegate tasks for the VO management to other administrators.

3.5.1. Administer the VO Membership of Users and Resources

A VO administrator wants to authorize other administrators to administer the VO membership.

3.5.1.1. Implementation in GT4

There is no support for the management of a VO using the standard GT4 installation. However, a GT4 VOMS Interceptor is available that can be configured for every service in GT4. Then the VOMS management policy applies also to GT4.

3.5.1.2. Implementation in gLite

In gLite it is possible to delegate administrative responsibilities to other users. An administrator can set the following permissions to administer the VO membership:

1. add
2. delete
3. delete-any-request
4. list
5. list-any request
6. remove

These permissions can be granted to the following entities:

- A. The local database administrator
- B. A specific user (not necessarily a member of this VO) which is represented by his DN and the CA that has signed his certificate
- C. Anyone who has a specific VOMS attribute
- D. Anyone who presents a certificate issued by a known CA (Including host and service certificates)
- E. Absolutely anyone, even unauthenticated clients

An administrator can set these permissions using a web interface.

3.5.1.3. Implementation with UNICORE

The VO concept is unknown to UNICORE. Therefore, such a grouping of users and resources can only be emulated by other means, if at all.

Case 1:

Specific users may, for instance, be grouped together within a POSIX group so as to emulate a VO structure, if this is supported by the underlying operating system.

Case 2:

Resources cannot be grouped trivially.

3.5.2. Definition of Groups/Roles, etc. within the VO

An administrator wants to authorize other users to define groups and roles within a VO.

3.5.2.1. Implementation in GT4

There is no support for the management of a VO using the standard GT4 installation.

3.5.2.2. Implementation in gLite

In gLite it is possible to delegate administrative responsibilities to other users. An administrator can set the following permissions to define groups and roles within a VO:

1. create
2. get-acl
3. get-default-acl
4. set-acl
5. set-default-acl

These permissions can be granted to the following entities:

1. The local database administrator
2. A specific user (not necessarily a member of this VO) which is represented by his DN and the CA that has signed his certificate
3. Anyone who has a specific VOMS attribute
4. Anyone who presents a certificate issued by a known CA (Including host and service certificates)
5. Absolutely anyone, even unauthenticated clients

An administrator can set these permissions using a web interface.

3.5.2.3. Implementation with UNICORE

Neither groups nor roles are concepts known to UNICORE.

Case 1:

Groups may be emulated with POSIX groups if the underlying operating system supports this.

Case 2:

Roles cannot be emulated trivially.

3.5.3. Authorizing Access to Sites/Resources for VOs/Groups/Users

An administrator wants to restrict dedicated VOs or groups respectively users from these VOs to access his site or resources.

3.5.3.1. Implementation in GT4

In GT4 there are no means to restrict access to sites or resources within the VO management.

3.5.3.2. Implementation in gLite

In gLite there are no means to restrict access to sites or resources within the VO management.

3.5.3.3. Implementation with UNICORE

Neither VOs nor groups are concepts known to UNICORE. Fine-grained control of the access to resources is also not supported.

Case 1:

Every member of the VO must be authorized individually.

Case 2:

Every member of the group must be authorized individually.

Case 3:

Every user must be authorized individually.

4. Overview

4.1. Authorization Measures by Users

	Globus Toolkit 4	gLite 3.0	UNICORE
Authorization Measures on Credentials			
Authorizing Access to User Credentials	Handled by Middleware; a) Delegation Service: only minimal adjustments of delegated authorization possible b) MyProxy: granularity based on DN	Handled by Middleware; granularity based on DN	Not supported by Middleware
Authorizing Proxy Renewal	Handled by Middleware; granularity based on DN	Handled by Middleware; granularity based on DN	Not supported by Middleware
Delegation of Restricted Rights	Not supported by Middleware. If VOMS Interceptor in use, then granularity based on VOMS attributes.	Handled by Middleware; granularity based on VOMS attributes; no user-defined policies	Handled by Middleware, following the Consignor/Endorser Model, but no Restricted Proxy Certificates.
Authorization Measures on Data Services			
Authorizing Access to unencrypted Data	Handled by Middleware; Community Authorization Service (CAS)	Handled by Middleware; Fireman: NTFS like permissions for files and directories regarding owner, group and others; permissions for additional entities configurable by ACLs	Yes, but requires knowledge about the target system (file paths, user mappings, file system access rights)
Authorizing Access to encrypted Data	Not supported by Middleware	Handled by Middleware; Fireman/Hydra Keystore: NTFS like permissions for files and directories regarding owner, group and others; permissions for additional entities configurable by ACLs	Not supported by Middleware
Authorizing File Transfer Service	Handled by Middleware; Community Authorization Service (CAS)	Not supported by Middleware	Not supported by Middleware
Authorization Measures on Compute Services			
Authorizing Access to Jobs	Not supported by Middleware	Not supported by Middleware	Not supported by Middleware
Authorizing Access to specific Resources	Not supported by Middleware (May be handled by the batch/operating system.)	Not supported by Middleware (May be handled by the batch/operating system.)	Not supported by Middleware (May be handled by the batch/operating system.)
Authorization Measures on Information Services			
Authorizing Access to Information about the Job Status	Not supported by Middleware	Handled by Middleware; currently limited to read access	Not supported by Middleware
Authorizing Access to Information about Executables, Input/Output Files	Not supported by Middleware	Handled by Middleware; access can be granted only to whole information about job and its status	Not supported by Middleware
Authorizing Access to Accounting Information	Not supported by Middleware	Not supported by Middleware	Not supported by Middleware

4.2. Authorization Measures by Administrators

	Globus Toolkit 4	gLite 3.0	UNICORE
Authorization Measures on Data Services			
Authorizing Access to Files and Directories	Handled by Middleware; Community Authorization Service (CAS) and Security Descriptors.	Handled by Middleware; Fireman: administrator can modify permissions and ownership; administrator role defined by VOMS attribute	Not supported by Middleware (May be handled by the operating system.)
Authorizing Access to Storage Space (Quota)	Not supported by Middleware (May be handled by the operating system.)	Not supported by Middleware (May be handled by the operating system.)	Not supported by Middleware (May be handled by the operating system.)
Authorizing Access to Resources for Data Services	Handled by Middleware; Service/Resource security descriptors may be used. If VOMS Interceptor in use, then granularity based on VOMS attributes.	Handled by Middleware; grant access to whole VOs only	Not supported by Middleware
Authorizing File Transfer Service	Handled by Middleware; Community Authorization Service (CAS) and Security Descriptors. If VOMS Interceptor in use, then granularity based on VOMS attributes.	Handled by Middleware; access to FTS/FPS is granted depending on VOMS attributes and on entries in <i>submit_mapfile</i> , <i>admin_mapfile</i>	Not supported by Middleware (May be handled by the operating system.)
Authorization Measures on Compute Services			
Restricting Access to Compute Resources	Not supported by Middleware (May be handled by the batch/operating system.)	Handled by Middleware and Batch System; Middleware: depending on their VOMS attributes users are mapped to pool accounts; Batch System: access to queues can be restricted to specific UNIX groups	Not supported by Middleware (May be handled by the batch/operating system.)
Restricting Access to (available) Executables	Not supported by Middleware (May be handled by the batch/operating system.)	Not supported by Middleware (May be handled by the batch/operating system.)	Not supported by Middleware (May be handled by the batch/operating system.)
Restricting execution of imported Executables	Not supported by Middleware (May be handled by the batch/operating system.)	Not supported by Middleware (May be handled by the batch/operating system.)	Not supported by Middleware (May be handled by the batch/operating system.)
Authorization Measures on Information Services			
Restricting Access to Information about Resources and Jobs	Configuring the sudo'ers file with authorization rules allows the WS GRAM service account to execute the scheduler adapter in the accounts of authorized GRAM users.	Not supported by Middleware.	Not supported by Middleware
Authorizing Access to Accounting Information	Not supported by Middleware	Handled by Middleware; DGAS: administrator is allowed to access all information; administrator grants users administrator status	Not supported by Middleware
Authorization Measures on VO Management			
Administer the VO Membership of Users and Resources	Not supported by standard installation of Middleware; possible if using VOMS Interceptor additionally.	Handled by Middleware; VOMS: administrator grants users roles and capabilities	Not supported by Middleware
Definition of Groups/Roles, etc. within the VO	Not supported by standard installation of Middleware; possible if using VOMS Interceptor additionally.	Handled by Middleware; VOMS: administrator grant users administrator status with certain permissions only	Not supported by Middleware
Authorizing Access to Sites/Resources for VOs/Groups/Users	Not supported by Middleware	Not supported by Middleware	Not supported by Middleware