Grid User Management System v.1.0.1

Project Documentation

BNL siteAAA **07 April 2005 14:37 EDT**

Table of Contents

1.1 Is GUMS for you

Is GUMS the right tool for your grid site?

This article is intended to provide enough information about GUMS that the reader can determine whether GUMS will fit his or her site's needs. We describe the functions GUMS performs and discuss how it fits in the GRID architecture. We also give a brief history of GUMS.

What does GUMS do?

The GUMS service performs one and only one function: it maps users' grid certificates/credentials to site-specific identities/credentials (e.g., UNIX accounts or Kerberos principals) in accordance with the site's grid resource usage policy. GUMS can be configured to generate static grid-mapfiles or to map users dynamically as each job is submitted. If configured to generate a grid-mapfile, GUMS downloads the file to each gatekeeper as scheduled or requested by an administrator via the GUMS Admin tool. If configured to map users dynamically and individually, GUMS is called by the gatekeeper upon each job submission.

Scenario: A job arrives at a site and gets routed to a particular gatekeeper. The job comes with a grid credential (the proxy certificate), but it will need to run under a site-specific credential. Before the gatekeeper can forward the job to the job manager, it (the gatekeeper) must obtain a site-specific credential for the job. Where does it get the site credential? Depending on the configuration, it either consults the GUMS-generated grid-mapfile or it calls the site's GUMS server and requests a site credential. In the latter case, GUMS maps the the grid credential dynamically to the appropriate site identity and credential, and sends the mapping information back to the gatekeeper. The gatekeeper now forwards the job along with its site credential to the job manager.

Notice that GUMS doesn't perform authentication: it doesn't 'su', it doesn't retrieve Kerberos credentials. It just tells the gatekeeper which site credentials the job should get. The gatekeeper is still in charge of enforcing the site mapping established by GUMS. Technically speaking, GUMS is a Policy Decision Point (PDP) not a Policy Enforcement Point (PEP).

Implementing Site Policies

GUMS runs at a grid site under the control of site administrators; it is a "site tool" as opposed to a "VO tool". The users it maps may be affiliated with numerous VOs. The mappings in a site's GUMS installation are defined in a single XML policy file. This file may contain multiple policies, and the administrator can assign different policies to different groups of users. The administrator can also specify different mappings on different hosts or different sets of hosts.

For example, say that there are two groups of users: all the users known to the ATLAS VOMS server, and other ATLAS users who are already mapped to accounts taken from a site pool of accounts. For all hosts 'atlas*.bnl.org', an administrator could map the first set of users to a group account named 'usatlas01', and let all the other users use the accounts to which they're already mapped. On another host or set of hosts, there could be a different mapping.

GUMS is designed to be extensible so that it can address specific site requirements. All the GUMS policy components (i.e., user group definition, mapping policies, and so on) are implementated via a few simple interfaces which exist outside of GUMS. Thus a site administrator with very little knowledge of GUMS itself can add external code to manipulate GUMS functionality or data, e.g, to tell GUMS how to map credentials, or to pull GUMS data into a local storage system.

Components for the following functionalities are already implemented in GUMS:

- Retrieve membership information from a VO server such as LDAP or VOMS.
- Maintain a manual group definition (this is useful to handle special cases).
- Map groups of users to the same account (a group account).
- Map groups of users to an account pool, in which one account will be assigned to each user.
- Map groups of users according to the information present in NIS.
- Map groups of users according to a manual mapping definition.

The GUMS callout interface

The GUMS interface for the callout is being implemented according to standards discussed at GGF, using the OSGA AuthZ interface with the SAML message format. The existence of this interface means that any kind of service is able to contact GUMS: even though we are working on a GT2 gatekeeper callout, the same service can be used for a GT4 service, or a different type of service altogether.

GUMS and other tools

It's possible to get confused between GUMS, VOMS, SAZ, LCMAPS and other siteAAA tools. See our FAQ for a brief explanation of the differences between GUMS and these other tools.

The development of GUMS

GUMS was first designed by Rich Baker and Dantong Yu at BNL in the first half of 2003. A first implementation was provided by Tomasz Wlodek and Dantong Yu. Gabriele Carcassi took over the project in March 2004 and brought the system into full production at BNL in May 2004. Between June and July 2004 the code was refactored to allow the business logic to be called either from the command line or from a web application, opening the door to a web service implementation.

Current work is focused on a web application that would provide both a web interface for the administrator and a web service that implements the OGSA AuthZ interface. This is being developed within the VO Privilege Project , a joint project between USCMS and USATLAS. This interface implements fine-grained authorization based on an extended grid proxy certificate.

GUMS in the future

The grid community, and especially the grid security groups, are moving toward a computing model in which (1) a grid job would be able to access a service only through grid credentials and (2) a job would not be allowed to leave any trace of its passage (or any traces would subsequently be erased). These two requirements would lessen the importance of site-specific credential mapping. The need to map to local identities therefore might go away in the future. Currently, though, many site-specific authorization decisions are still made using the username and uid, and GUMS provides a necessary function.

The following is an incomplete list of issues that will need to be resolved before local account mapping becomes irrelevant:

- File access control. If file access for all inputs and outputs of a job were to go through an interface with grid authorization, say an SRM, then whether a job was mapped to a particular account wouldn't affect file access privileges. As long as we use UNIX file systems directly for storage, local account mappings are crucial.
- Priority on a batch system. Many batch systems use uids and gids to determine submission rights to queues and/or to determine priorities for users. Batch system interfaces that make decisions based-only on grid credentials would be needed.
- Identification of running processes. The easiest way to track a process running on a host is to look at the username under which it is running. We'd need a mechanism that associates a process id with a grid identity.

1.2 GUMS Architecture

GUMS Architecture and Workflow

This article is intended to describe GUMS in somewhat more detail than (ref other page). Here we provide information about the components that make up GUMS, what it interfaces to and how, the workflow, and how it fits in the GRID architecture for a number of scenarios. For further reading, see Transition to Role-based Assignment of Local User-Ids .

Mapping concepts

GUMS can generate mapping that gets transferred periodically to a gatekeeper with gums-host, which generates a suitable grid-mapfile. Or GUMS can respond dynamically to individual requests for mapping from a gatekeeper. In either case, users can be mapped to accounts in a variety of ways:

- map groups of users to shared group accounts,
- map users to pool accounts, one user per account, or
- map users to group accounts manually

The accounts themselves are created outside of GUMS. Particular accounts or ranges of accounts (e.g., xyz0000 to xyz9999) are then specified in the GUMS configuration file. The mapping policy as implemented in this file may take into account the grid identity of the user and his/her VO only, or it can accommodate extended attributes (implemented via voms-proxy-init) such as the user's role and group within the VO. Furthermore, different mapping policies may be implemented depending on the host chosen to process the job.

Components used for mapping

GUMS stores information about users (e.g., their DN and VO) and (if pool accounts are implemented) the user-to-account mapping. GUMS can perform mapping in a variety of ways depending on the requests it receives and the configuration file (also called a policy file) that governs it. The configuration file addresses three aspects of mapping:

- Where to store user and mapping data (the storage is referred to as "persistence" in GUMS; the interface to it is called a "persistence factory")
- Definitions of user groups and how they get mapped to accounts
- Definitions of user group mappings on various hosts or groups of hosts

A VO service (e.g., VOMS Admin or LDAP) must be available to provide user and VO information to GUMS. GUMS downloads this information and stores it locally according to the persistence factory that's configured. The persistent storage may be the GUMS-native MySQL database or a site-specific storage service.

Mapping requests come from one or more gatekeepers.

The GUMS client tools are installed on the gatekeepers, and provide functionality for

- mapping users to manually-managed group accounts
- retrieving maps from GUMS server and creating a grid-mapfile
- forcing GUMS server to update the user lists from the VO servers

Workflow

For configuration:

- Admin creates accounts (the accounts must be available on the gatekeeper and the worker nodes)
- Admin creates GUMS configuration file specifying which VO servers and which persistence (e.g., mysql) will be used
- Admin configures the gatekeeper to either retrieve the grid-mapfile from GUMS or to use the PRIMA module to connect to GUMS
- GUMS reads the configuration file (the "rules") into memory.
- GUMS consults the rules to find persistence factory information, and sets up connection to persistent storage
- GUMS contacts VO service and downloads grid id information for all users in VO; stores it in persistence

For processing an individual mapping request (assuming only group accounts are to be used, no pool accounts):

- Gatekeeper sends request for mapping to GUMS
- GUMS receives request from gatekeeper
- GUMS looks in user table to verify that user is there
- GUMS consults configuration file rules (in memory) to find out how to map user (in this case, it uses only group accounts).
- GUMS maps user to the first group account for which he or she is eligible
- GUMS sends message out via web service to gatekeeper; message contains one of three responses: Permission granted (send with "obligations"), Indeterminate (permission is not granted, but is not denied either), or Denied (user is not valid).

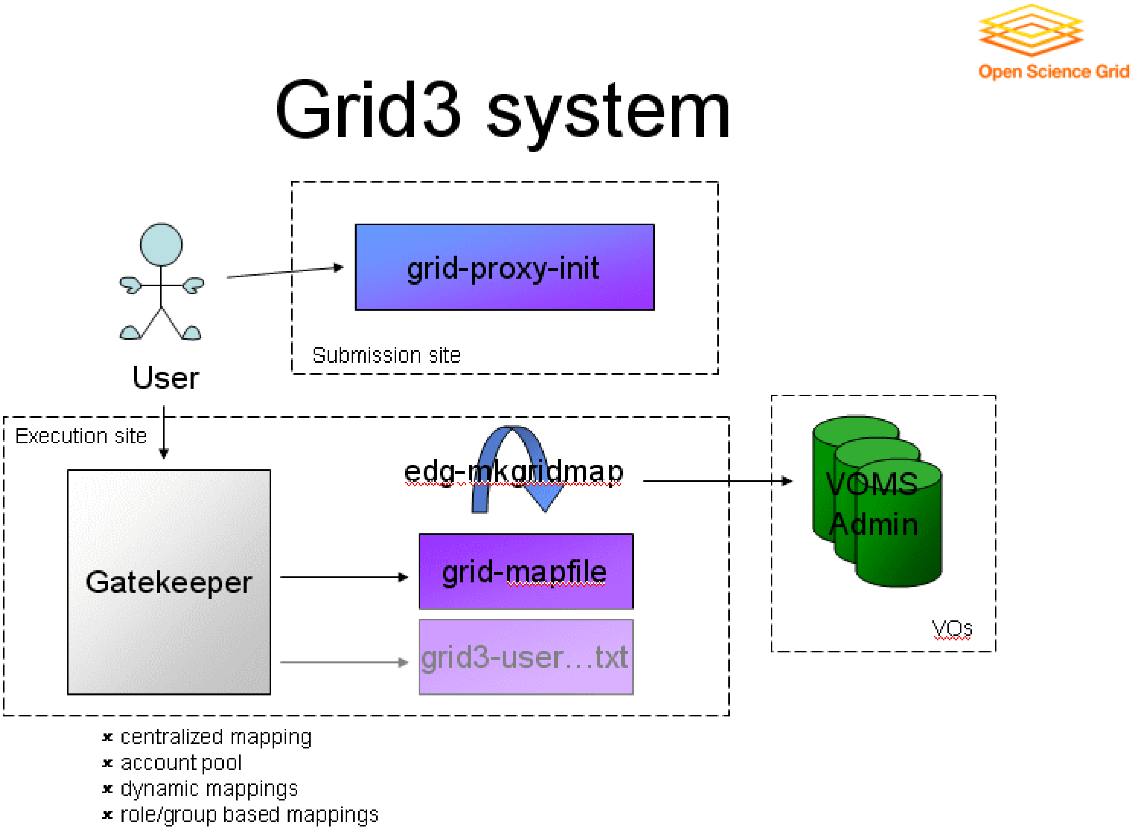
For processing an individual mapping request (assuming pool accounts are to be used):

- Gatekeeper sends request for mapping to GUMS
- GUMS receives request from gatekeeper
- GUMS looks in user table to verify that user is there
- GUMS consults configuration file rules (in memory) to find out how to map user.
- If user should be mapped to a pool account, GUMS checks the userGroupMapping table to see if user is already mapped. If so, GUMS uses that mapping. If not, it creates a mapping and stores it in the table.
- GUMS sends message out via web service to gatekeeper: Permission granted (send with

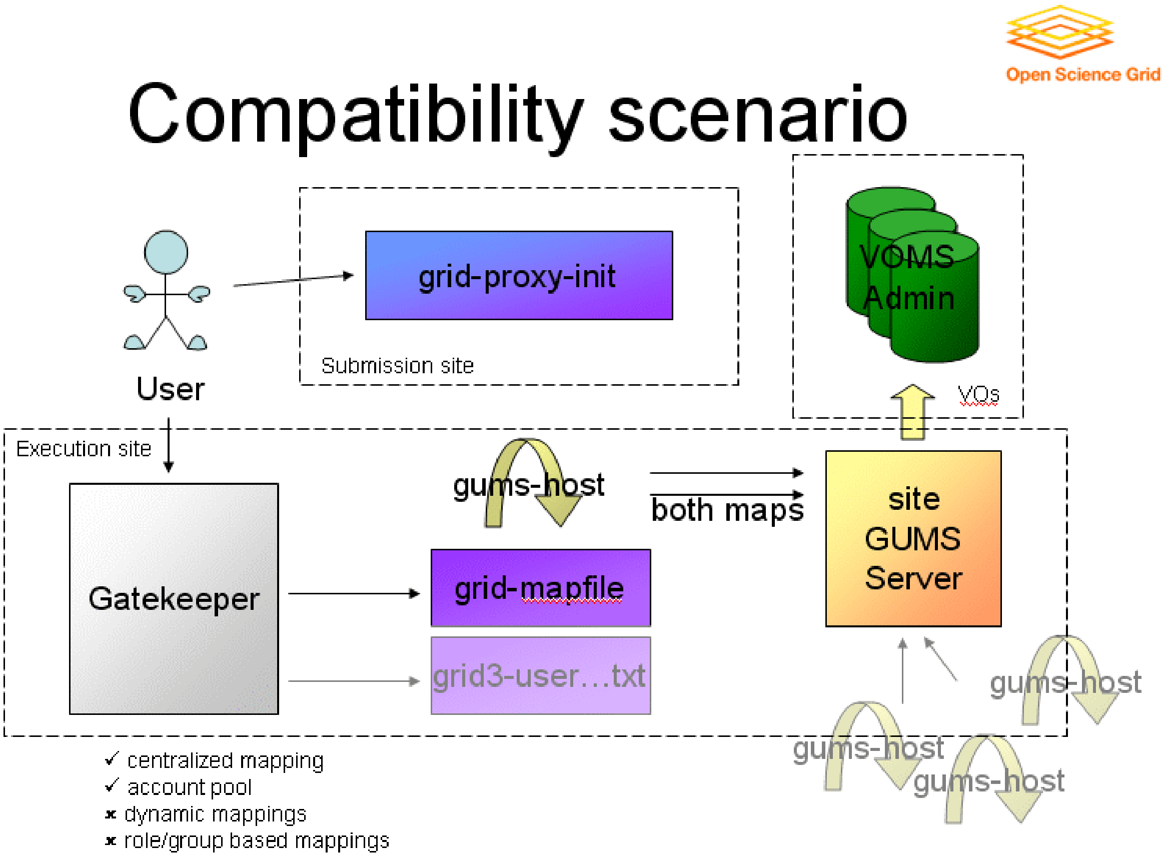
"obligations"), Indeterminate (permission is not granted, but is not denied either), or Denied (user is not valid).

GUMS in the grid architecture

First, let's look at the Grid3 system which did not use GUMS. Users run grid-proxy init to get credentials, without contacting the VOMS server. Each gatekeeper at each site has to connect to VOMS independently (e.g., via edg-mkgridmap) in order to create its static grid-mapfile. This is done periodically. Note, as shown at bottom of image, that there is no support for centralized mapping at a site, account pools, dynamic mappings, or role/group-based mappings. The inverse map for accounting (i.e., user-to-VO mapping) is created manually.



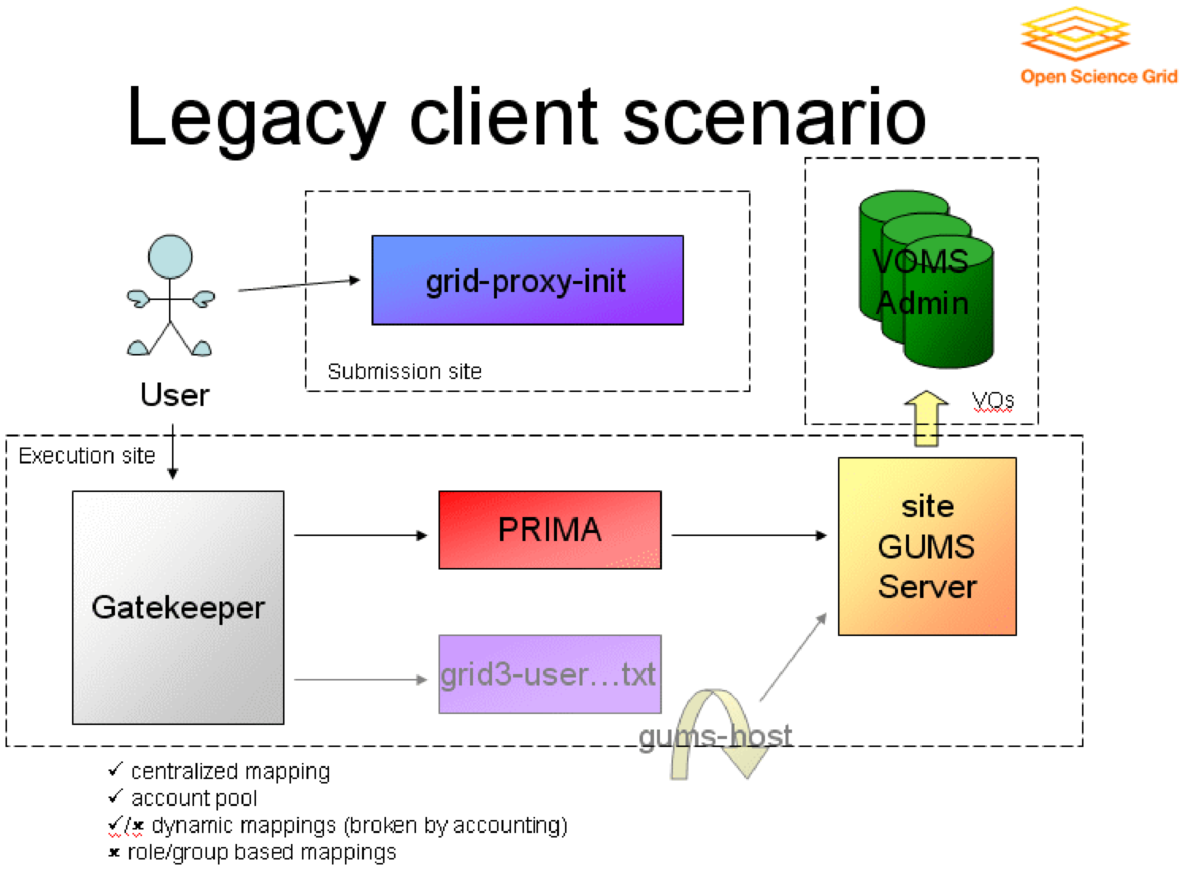
The next scenario shows GUMS implemented such that it sits between VOMS Admin and the client(s). The gatekeeper does not implement a callout to GUMS. GUMS polls VOMS Admin periodically to update its local list of users. The site may have only one GUMS server, in which case there would be only one communication point to VOMS, thus enabling centralized mapping. All clients would access the same information locally. (A site could deploy multiple GUMS servers, but would lose the centralized mapping capability.) GUMS downloads mapping information to each gatekeeper: the GUMS host tool, gums-host, replaces edg-mkgridmap in this diagram, although edg-mkgridmap could still be used to create the grid-mapfile; see the FAQ . The grid-mapfiles on all the gatekeeper can be made identical to each other if they all use the same GUMS server (you can still have different maps for different gatekeepers of group of gatekeepers). Gums-host also creates the inverse map for accounting. Mapping to account pools is now available.



Compatibility scenario

✓ centralized mapping
✓ account pool
✗ dynamic mappings
✗ role/group based mappings

The next image shows GUMS implemented in a legacy client scenario. Here we deploy the PRIMA module (see the VO Privilege Project) on the gatekeepers in order to enable dynamic mapping. There are no more grid-mapfiles. We continue to use the gums-host tool to generate the accounting map (if needed), but this breaks dynamic mapping when account pools are being used. (Whenever GUMS is asked to generate a map, it has to go through the whole policy, and assign an account for all the different possibilities. Therefore, accounts will be assigned to everybody beforehand (when generating the map) instead of when the individual request comes in.)

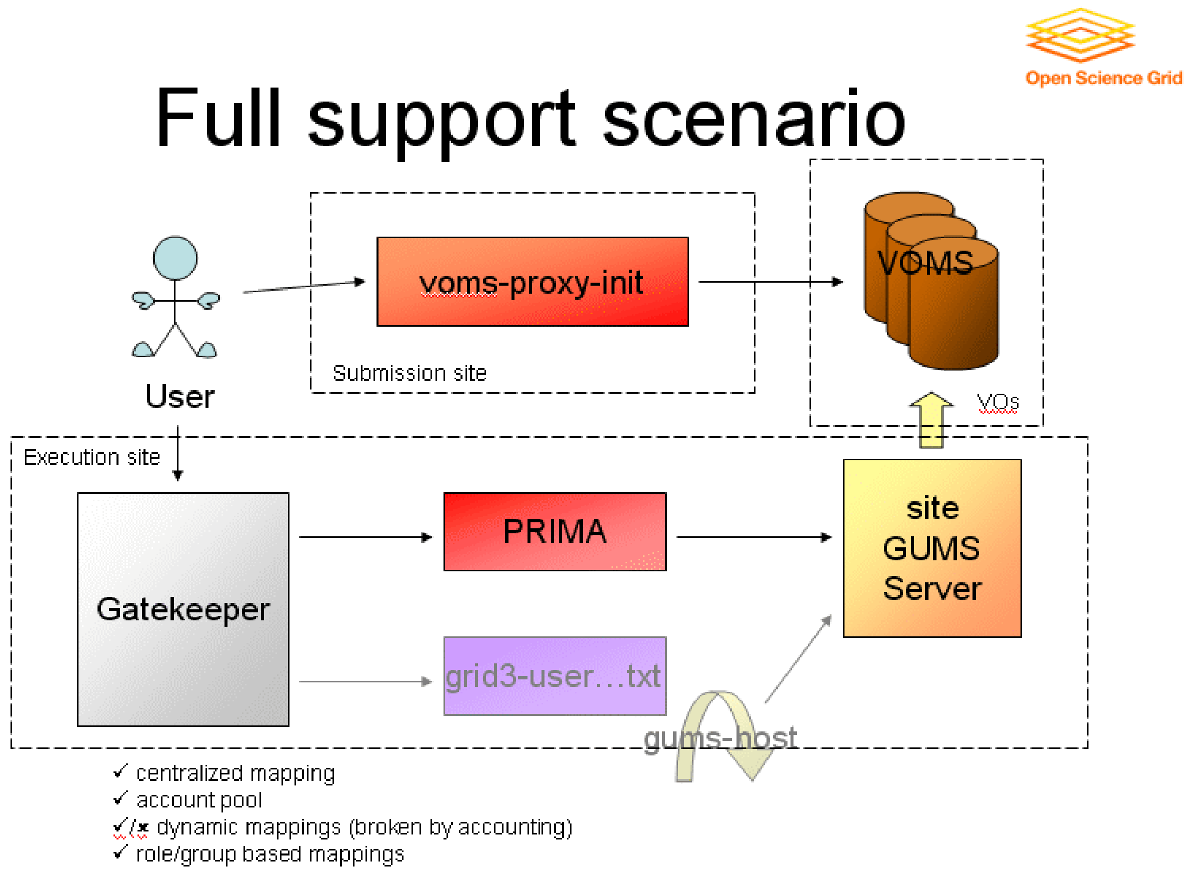


Legacy client scenario

- ✓ centralized mapping
- ✓ account pool
- ✓/✗ dynamic mappings (broken by accounting)
- ✗ role/group based mappings

The next image shows GUMS implemented in a full support scenario. This is similar to the previous scenario except that the user now runs voms-proxy-init which VOMS uses to produce an extended proxy with role/group information. This enables GUMS to produce role/group-based mappings.

We continue to use the gums-host tool to generate the accounting map if needed, but this breaks dynamic mapping when account pools are being used, as discussed above.



Open Science Grid

Full support scenario

User

voms-proxy-init

Submission site

VOMS

VOs

Execution site

Gatekeeper

PRIMA

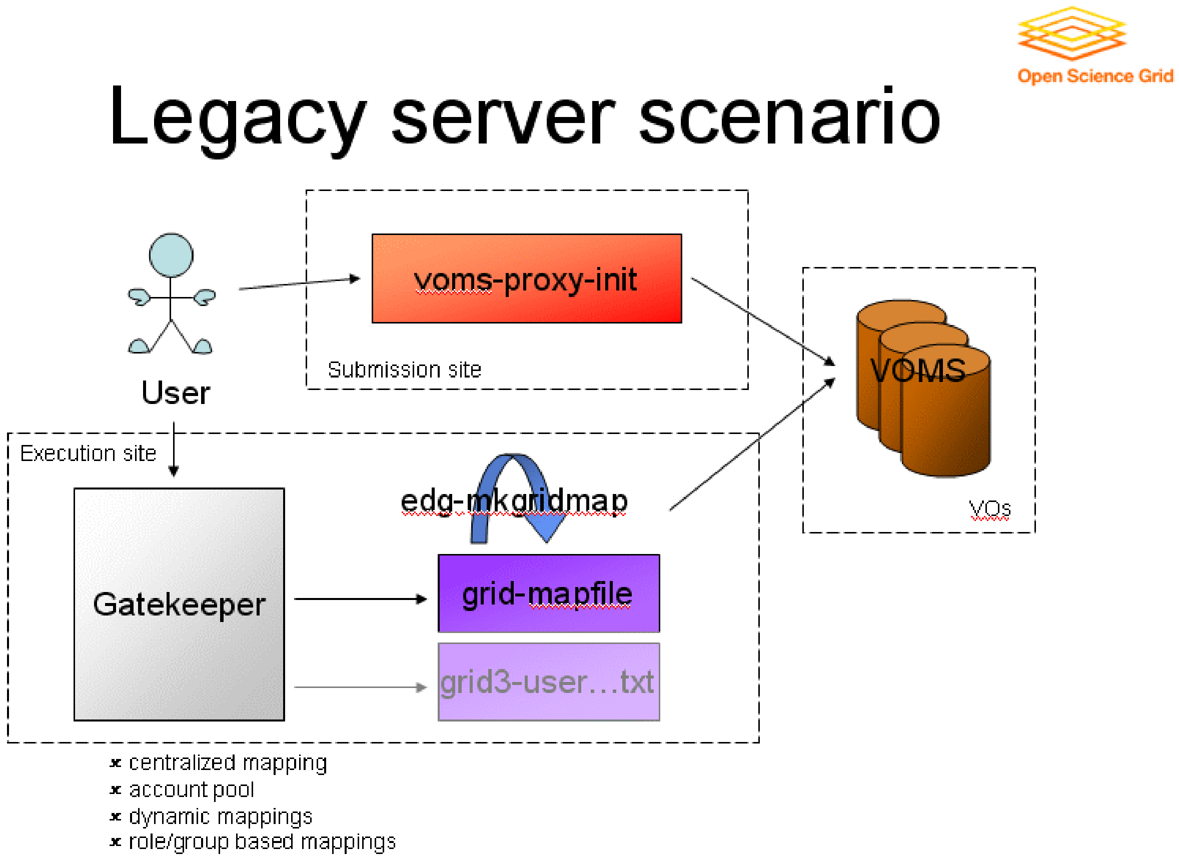
grid3-user…txt

site GUMS Server

gums-host

- ✓ centralized mapping
- ✓ account pool
- ✓/✗ dynamic mappings (broken by accounting)
- ✓ role/group based mappings

The next image shows a legacy server scenario. You can use voms-proxy-init, but you can't take advantage of its extended proxy features. You're back to having no support for any of the four items listed at bottom.



Open Science Grid

Legacy server scenario

User

voms-proxy-init

Submission site

VOMS

VOs

Execution site

Gatekeeper

edg-mkgridmap

grid-mapfile

grid3-user…txt

- centralized mapping
- account pool
- dynamic mappings
- role/group based mappings

1.3 How to: quick installation

HOW TO: GUMS Service quick-start installation

This article describes how to quickly install a GUMS service, without going through many of the details. As of GUMS 1.0.0, the preferred way to install GUMS will be VDT. You should use this guide only if you need to install GUMS without using VDT (i.e. when you need the really latest version).

Preparing java

GUMS is written in java, and requires java to be installed to run. Be sure it is installed in your $PATH. Try running:

```
[root@gums root]# java -version
java version "1.4.2_04"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2_04-b05)
Java HotSpot(TM) Client VM (build 1.4.2_04-b05, mixed mode)
```

If you do not have java installed, go to http://java.sun.com and follow the instructions to get the latest version. Then add java to $PATH.

Preparing the certificate directory

GUMS will need the GRID certificate for the host and the certificate directories in place. Easiest way: use VDT; but you are not, so you are on your own here.

Prepare the database

You will need a mysql server, with version 4.0.18 or greater installed. You can either install one from scratch (follow the instruction on mysql's site) or you can use an installation you have ready. You'll need root password.

Preparing Tomcat + EGEE security

The GUMS service will require a web server container, configured to use SSL with Globus proxy certificates. You will also need Xerces 2.5.0 in the common/endorsed directory. If you do not know what that means, just grab the already packaged Tomcat from the download page and install it.

- Grab the tarball from the download section [root@gums root]# cd /opt/ [root@gums opt]# wget http://grid.racf.bnl.gov/GUMS/dist/tomcat-5.0.28-egeesec.tar.gz .
- Untar it [root@gums opt]# tar -xzvf tomcat-5.0.28-egeesec.tar.gz

- Review the configuration of the server [root@gums opt]# vi tomcat-5.0.28-egeesec/conf/server.xml

  NOTE: if you need to change the configuration for the service certificate, or the port on which the service runs, you can edit the $TOMCAT_HOME/conf/server.xml tomcat configuration file. Find the section:

  <Connector port="**8443**" maxThreads="150" minSpareThreads="25" maxSpareThreads="75" enableLookups="false" disableUploadTimeout="true" acceptCount="100" debug="0" scheme="https" secure="true" sSLImplementation="org.glite.security.trustmanager.tomcat.TMSSLImplementation" sslCAFiles="/etc/grid-security/certificates/*.0" crlFiles="/etc/grid-security/certificates/*.r0" sslCertFile="**/etc/grid-security/hostcert.pem**" sslKey="**/etc/grid-security/hostkey.pem**" log4jConfFile="**/opt/tomcat-5.0.28-egeesec/**conf/log4j-trustmanager.properties" clientAuth="true" sslProtocol="TLS" />

  If you installed in a different directory than /opt/tomcat-5.0.28-egeesec, change the location of log4jConfFile. To change the location of the service certificate or the CAs, simply change the sslXxx and crlFiles properties. To change the port, change the port property.

- Start the server [root@gums opt]# tomcat-5.0.28-egeesec/bin/catalina.sh start
- Connect to the server through a web browser with a Grid certificate installed, to check that is indeed running.

Install the service

The service itself is a standard java web application. Grab the latest gums-service tarball file from the dist directory , and unpack it.

- Grab the latest build and install [root@gums root]# cd /opt/ [root@gums opt]# wget http://grid.racf.bnl.gov/GUMS/dist/gums-service-1.0.0dev-latest.tar.gz . [root@gums opt]# tar -xzvf gums-service-1.0.0dev-latest.tar.gz
- You will need to create the database. You can do by running the setupDatabase script giving[root@gums opt]# cd gums-service/sbin/ [root@gums sbin]# ./setupDatabase Usage: ./setupDatabase [mysql user for GUMS] [GUMS server host] [GUMS mysql password] Examples: ./setupDatabase gums gums.mysite.com secret This will make mysql authenticate as root with a password (-p), create a 'gums' user with password 'secret' authorized to connect from 'gums.mysite.com'. [root@gums sbin]# ./setupDatabase gums gums.mysite.com secret

  The script will only run on localhost. If you need to create db on another server, or with different account then root, edit setupDatabase:

  [root@gums sbin]# cat setupDatabase #!/bin/sh ... MYSQLPARAM="-p" ..

- Now add yourself to admins:[root@gums sbin]# ./addAdmin Adds an admin in the GUMS database on localhost Usage: ./addAdmin [DN for administrator] Example: ./addAdmin "/DC=org/DC=doegrids/OU=People/CN=Gabriele Carcassi 12345" [root@gums sbin]# ./addAdmin "/DC=org/DC=doegrids/OU=People/CN=Your Self 83753"
- Last thing, we need to tell tomcat to run GUMS: [root@gums sbin]# cd ../.. [root@gums opt]# ln -s ../../gums-service/var/war tomcat-5.0.28-egeesec/webapps/gums
- Get a browser in which you have your grid certificate, go to: https://<machine>:<port>/gums : you should see the GUMS web interface. You might need to wait a bit for tomcat to realize the gums

application was installed.

- Try generating the mapfile for the host "testing.site.com", and it should give you some response.
- [Optional] Another thing you can do is activate the e-mail forwarding of the log in case of error. Edit /opt/gums-service/webapps/gums/WEB-INF/classes/log4j.properties, following the instructions within the file. Fill in the appropriate information, restart the service, and whenever GUMS will encounter an error, an e-mail will be sent to the address you specified.
- [Optional] GUMS provides a log suitable for cybersecurity in the /opt/gums-service/logs/gums-site-admin.log. The same log can be configured to use syslogd. Please refer to the logging documentation for more details.
- [Optional] One of the things GUMS does, is downloading the information from the VO servers every 12 hours. The clock will start at the first access to GUMS functionalities after each restart (i.e. first time you actually generate a mapfile or map a user). The time between updates can be set. The easiest way to do it, is to edit the /opt/gums-service/webapps/gums/WEB-INF/web.xml file. <env-entry> <env-entry-name>updateGroupsMinutes</env-entry-name> <env-entry-type>java.lang.Integer</env-entry-type> <env-entry-value>720</env-entry-value> </env-entry>Change the entry value to the number of mintues you prefer.

Congratulations: the server is up and running.

Installing the client

To generate maps (grid-mapfile, grid3-user-vo.txt) or to administer GUMS (i.e. create account pools, manage manual groups, ...) you will need to install GUMS Client which gives includes all the command line tools.

Grab the latest rpm file from the dist directory : it will install by default in /opt/gums, but the package is relocatable, so you can install it wherever you want.

- Grab the latest build and install [root@gums root]# wget http://grid.racf.bnl.gov/GUMS/dist/gums-admin-1.0.0dev-latest.noarch.rpm . [root@gums root]# rpm -Uvh gums-admin-1.0.0dev-latest.noarch.rpm
- Check that GUMS was installed correctly[root@gums root]# cd /opt/gums/bin/ [root@gums bin]# ./gums usage: gums command [command-options] Commands: generateGrid3UserVoMap - Generate grid3-user-vo-map.txt for a given host. generateGridMapfile - Generate grid-mapfile for a given host. manualGroup-add - Includes a DN in a group. manualGroup-remove - Removes a DN from a group. manualMapping-add - Adds a DN-to-username in a mapping. manualMapping-remove - Removes a DN from a mapping. mapUser - Local credential used for a particular user. mapfileCache-refresh - Regerates mapfiles in the cache. updateGroups - Contact VO servers and retrieve user lists. For help on any command: gums command --help
- Now, you need to tell gums-admin where is your GUMS server.

  ```
  [root@gums bin]# cat ../etc/gums-client.properties
  gums.location=https://localhost:8443/gums/services/GUMSAdmin
  gums.authz=https://localhost:8443/gums/services/GUMSAuthorizationServicePort
  ```

  Replace the localhost with the full machine name (even if you installed GUMS Admin on the same machine).

```
[root@gums bin]# vi ../etc/gums-client.properties [root@gums bin]# cat
../etc/gums-client.properties
gums.location=https://gums.mysite.com:8443/gums/services/GUMSAdmin
gums.authz=https://gums.mysite.com:8443/gums/services/GUMSAuthorizationServicePort
```

You might need to set the permission correctly: GUMS can be run only by admins and, due to the file privileges on the logs and configuration files, you will need to set the username and group accordingly. You will need to change permission on the directory:

```
[root@gums bin]# chown username:groupname .. -R
```

where groupname is a group that contain all the admins.

- Test the service by generating a mapfile at the command line

  ```
  [root@gums bin]# su - username [username@gums bin]# grid-proxy-init [username@gums bin]#
  ./gums generateGridMapfile testing.test.gov
  ```

  You should get the same response you got from the web server.

- You will have noticed that there are two command in the bin directory, plus a cron script: gums is meant to be run by an admin, and uses your proxy; gums-host and gums-host-cron are meant to be run using the host credentials, and they have limited functionalities (retrieve maps and mappings only). To make GUMS generate the maps periodically, just link the cron script in /etc/cron.hourly.

Congratulations you installed GUMS Admin. To make it actually useful, now you need to go back on the server and write an XML policy file. Please refer to the rest of the GUMS documentation. As an example of configuration, this is a configuration file suitable for OSG:

```
<gums>
  <persistenceFactories>
    <persistenceFactory name='mysql'
className='gov.bnl.gums.MySQLPersistenceFactory' jdbcDriver='com.mysql.jdbc.Driver'
jdbcUrl='jdbc:mysql://*your_server*/GUMS_0_7' user='gums' password='*password*'
autoReconnect='true' />
  </persistenceFactories>
  <adminUserGroup className='gov.bnl.gums.ManualUserGroup'
persistenceFactory='mysql' name='admins' />
  <groupMappings>
    <groupMapping name='grase' accountingVo='grase' accountingDesc='GRASE'>
      <userGroup className='gov.bnl.gums.VOMSGroup'
url='https://acdc.ccr.buffalo.edu:8443/edg-voms-admin-GRASE/services/VOMSAdmin'
persistenceFactory='mysql'
      name='grase' voGroup="/GRASE/grid3"
sslCertfile='/etc/grid-security/hostcert.pem'
sslKey='/etc/grid-security/hostkey.pem' ignoreFQAN="true"/>
      <accountMapping className='gov.bnl.gums.GroupAccountMapper' groupName='grase'
/>
    </groupMapping>
    <groupMapping name='gadu' accountingVo='gadu' accountingDesc='GADU'>
      <userGroup className='gov.bnl.gums.VOMSGroup'
url='https://cmsvo.fnal.gov:8443/edg-voms-admin/gadu/services/VOMSAdmin'
persistenceFactory='mysql'
      name='gadu' voGroup="/gadu/grid3users"
```

```
sslCertfile='/etc/grid-security/hostcert.pem'
sslKey='/etc/grid-security/hostkey.pem' ignoreFQAN="true"/>
      <accountMapping className='gov.bnl.gums.GroupAccountMapper' groupName='gadu'
/>
    </groupMapping>
    <groupMapping name='usatlas' accountingVo='usatlas' accountingDesc='ATLAS'>
      <userGroup className='gov.bnl.gums.VOMSGroup'
url='https://cmssrv08.fnal.gov:8443/edg-voms-admin/atlas/services/VOMSAdmin'
persistenceFactory='mysql'
      name='usatlas' voGroup="/atlas/usatlas"
sslCertfile='/etc/grid-security/hostcert.pem'
sslKey='/etc/grid-security/hostkey.pem' ignoreFQAN="true"/>
      <accountMapping className='gov.bnl.gums.GroupAccountMapper'
groupName='usatlas1' />
    </groupMapping>
    <groupMapping name='uscms' accountingVo='uscms' accountingDesc='CMS'>
      <userGroup className='gov.bnl.gums.VOMSGroup'
url='https://cmsvo.fnal.gov:8443/edg-voms-admin/uscms/services/VOMSAdmin'
persistenceFactory='mysql'
      name='uscms' voGroup="/uscms/grid3_users"
sslCertfile='/etc/grid-security/hostcert.pem'
sslKey='/etc/grid-security/hostkey.pem' ignoreFQAN="true"/>
      <accountMapping className='gov.bnl.gums.GroupAccountMapper'
groupName='uscms01' />
    </groupMapping>
    <groupMapping name='ivdgl' accountingVo='ivdgl' accountingDesc='iVDgL'>
      <userGroup className='gov.bnl.gums.VOMSGroup'
url='https://pki0.indiana.edu:8443/edg-voms-admin/g3dev/services/VOMSAdmin'
persistenceFactory='mysql'
      name='ivdgl' voGroup="/g3dev/group1"
sslCertfile='/etc/grid-security/hostcert.pem'
sslKey='/etc/grid-security/hostkey.pem' ignoreFQAN="true"/>
      <accountMapping className='gov.bnl.gums.GroupAccountMapper' groupName='ivdgl'
/>
    </groupMapping>
    <groupMapping name='sdss' accountingVo='sdss' accountingDesc='SDSS'>
      <userGroup className='gov.bnl.gums.VOMSGroup'
url='https://hotdog63.fnal.gov:8443/edg-voms-admin-sdss/services/VOMSAdmin'
persistenceFactory='mysql'
      name='sdss' voGroup="/sdss/grid3"
sslCertfile='/etc/grid-security/hostcert.pem'
sslKey='/etc/grid-security/hostkey.pem' ignoreFQAN="true"/>
      <accountMapping className='gov.bnl.gums.GroupAccountMapper' groupName='sdss'
/>
    </groupMapping>
    <groupMapping name='cdf' accountingVo='cdf' accountingDesc='CDF'>
      <userGroup className='gov.bnl.gums.VOMSGroup'
url='https://testbed008.cnaf.infn.it:8443/edg-voms-admin/cdf/services/VOMSAdmin'
persistenceFactory='mysql'
      name='cdf' sslCertfile='/etc/grid-security/hostcert.pem'
sslKey='/etc/grid-security/hostkey.pem' ignoreFQAN="true"/>
      <accountMapping className='gov.bnl.gums.GroupAccountMapper' groupName='cdf' />
    </groupMapping>
    </groupMappings>
    <hostGroups>
    <hostGroup className='gov.bnl.gums.WildcardHostGroup'
wildcard='*,*.*,*.*.*,*.*.*.*' groups='grase,gadu,usatlas,uscms,ivdgl,sdss,cdf' />
  </hostGroups>
</gums>
```

Problems?

If you have any problem, feel free to contact GUMS developers.

1.4 Installation

1.4.1 Quick installation

HOW TO: GUMS Service quick-start installation

This article describes how to quickly install a GUMS service, without going through many of the details. As of GUMS 1.0.0, the preferred way to install GUMS will be VDT. You should use this guide only if you need to install GUMS without using VDT (i.e. when you need the really latest version).

Preparing java

GUMS is written in java, and requires java to be installed to run. Be sure it is installed in your $PATH. Try running:

```
[root@gums root]# java -version
java version "1.4.2_04"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2_04-b05)
Java HotSpot(TM) Client VM (build 1.4.2_04-b05, mixed mode)
```

If you do not have java installed, go to http://java.sun.com and follow the instructions to get the latest version. Then add java to $PATH.

Preparing the certificate directory

GUMS will need the GRID certificate for the host and the certificate directories in place. Easiest way: use VDT; but you are not, so you are on your own here.

Prepare the database

You will need a mysql server, with version 4.0.18 or greater installed. You can either install one from scratch (follow the instruction on mysql's site) or you can use an installation you have ready. You'll need root password.

Preparing Tomcat + EGEE security

The GUMS service will require a web server container, configured to use SSL with Globus proxy certificates. You will also need Xerces 2.5.0 in the common/endorsed directory. If you do not know what that means, just grab the already packaged Tomcat from the download page and install it.

- Grab the tarball from the download section [root@gums root]# cd /opt/ [root@gums opt]# wget http://grid.racf.bnl.gov/GUMS/dist/tomcat-5.0.28-egeesec.tar.gz .
- Untar it [root@gums opt]# tar -xzvf tomcat-5.0.28-egeesec.tar.gz

- Review the configuration of the server [root@gums opt]# vi tomcat-5.0.28-egeesec/conf/server.xml

  NOTE: if you need to change the configuration for the service certificate, or the port on which the service runs, you can edit the $TOMCAT_HOME/conf/server.xml tomcat configuration file. Find the section:

  <Connector port="**8443**" maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
  enableLookups="false" disableUploadTimeout="true" acceptCount="100" debug="0"
  scheme="https" secure="true"
  sSLImplementation="org.glite.security.trustmanager.tomcat.TMSSLImplementation"
  sslCAFiles="/etc/grid-security/certificates/*.0" crlFiles="/etc/grid-security/certificates/*.r0"
  sslCertFile="**/etc/grid-security/hostcert.pem**" sslKey="**/etc/grid-security/hostkey.pem**"
  log4jConfFile="**/opt/tomcat-5.0.28-egeesec/**conf/log4j-trustmanager.properties"
  clientAuth="true" sslProtocol="TLS" />

  If you installed in a different directory than /opt/tomcat-5.0.28-egeesec, change the location of log4jConfFile. To change the location of the service certificate or the CAs, simply change the sslXxx and crlFiles properties. To change the port, change the port property.

- Start the server [root@gums opt]# tomcat-5.0.28-egeesec/bin/catalina.sh start
- Connect to the server through a web browser with a Grid certificate installed, to check that is indeed running.

Install the service

The service itself is a standard java web application. Grab the latest gums-service tarball file from the dist directory , and unpack it.

- Grab the latest build and install [root@gums root]# cd /opt/ [root@gums opt]# wget http://grid.racf.bnl.gov/GUMS/dist/gums-service-1.0.0dev-latest.tar.gz . [root@gums opt]# tar -xzvf gums-service-1.0.0dev-latest.tar.gz
- You will need to create the database. You can do by running the setupDatabase script giving[root@gums opt]# cd gums-service/sbin/ [root@gums sbin]# ./setupDatabase Usage: ./setupDatabase [mysql user for GUMS] [GUMS server host] [GUMS mysql password] Examples: ./setupDatabase gums gums.mysite.com secret This will make mysql authenticate as root with a password (-p), create a 'gums' user with password 'secret' authorized to connect from 'gums.mysite.com'. [root@gums sbin]# ./setupDatabase gums gums.mysite.com secret

  The script will only run on localhost. If you need to create db on another server, or with different account then root, edit setupDatabase:

  [root@gums sbin]# cat setupDatabase #!/bin/sh ... MYSQLPARAM="-p" ..

- Now add yourself to admins:[root@gums sbin]# ./addAdmin Adds an admin in the GUMS database on localhost Usage: ./addAdmin [DN for administrator] Example: ./addAdmin "/DC=org/DC=doegrids/OU=People/CN=Gabriele Carcassi 12345" [root@gums sbin]# ./addAdmin "/DC=org/DC=doegrids/OU=People/CN=Your Self 83753"
- Last thing, we need to tell tomcat to run GUMS: [root@gums sbin]# cd ../.. [root@gums opt]# ln -s ../../gums-service/var/war tomcat-5.0.28-egeesec/webapps/gums
- Get a browser in which you have your grid certificate, go to: https://<machine>:<port>/gums : you should see the GUMS web interface. You might need to wait a bit for tomcat to realize the gums

application was installed.

- Try generating the mapfile for the host "testing.site.com", and it should give you some response.
- [Optional] Another thing you can do is activate the e-mail forwarding of the log in case of error. Edit /opt/gums-service/webapps/gums/WEB-INF/classes/log4j.properties, following the instructions within the file. Fill in the appropriate information, restart the service, and whenever GUMS will encounter an error, an e-mail will be sent to the address you specified.
- [Optional] GUMS provides a log suitable for cybersecurity in the /opt/gums-service/logs/gums-site-admin.log. The same log can be configured to use syslogd. Please refer to the logging documentation for more details.
- [Optional] One of the things GUMS does, is downloading the information from the VO servers every 12 hours. The clock will start at the first access to GUMS functionalities after each restart (i.e. first time you actually generate a mapfile or map a user). The time between updates can be set. The easiest way to do it, is to edit the /opt/gums-service/webapps/gums/WEB-INF/web.xml file. <env-entry> <env-entry-name>updateGroupsMinutes</env-entry-name> <env-entry-type>java.lang.Integer</env-entry-type> <env-entry-value>720</env-entry-value> </env-entry>Change the entry value to the number of mintues you prefer.

Congratulations: the server is up and running.

Installing the client

To generate maps (grid-mapfile, grid3-user-vo.txt) or to administer GUMS (i.e. create account pools, manage manual groups, ...) you will need to install GUMS Client which gives includes all the command line tools.

Grab the latest rpm file from the dist directory : it will install by default in /opt/gums, but the package is relocatable, so you can install it wherever you want.

- Grab the latest build and install [root@gums root]# wget http://grid.racf.bnl.gov/GUMS/dist/gums-admin-1.0.0dev-latest.noarch.rpm . [root@gums root]# rpm -Uvh gums-admin-1.0.0dev-latest.noarch.rpm
- Check that GUMS was installed correctly[root@gums root]# cd /opt/gums/bin/ [root@gums bin]# ./gums usage: gums command [command-options] Commands: generateGrid3UserVoMap - Generate grid3-user-vo-map.txt for a given host. generateGridMapfile - Generate grid-mapfile for a given host. manualGroup-add - Includes a DN in a group. manualGroup-remove - Removes a DN from a group. manualMapping-add - Adds a DN-to-username in a mapping. manualMapping-remove - Removes a DN from a mapping. mapUser - Local credential used for a particular user. mapfileCache-refresh - Regerates mapfiles in the cache. updateGroups - Contact VO servers and retrieve user lists. For help on any command: gums command --help
- Now, you need to tell gums-admin where is your GUMS server.

  [root@gums bin]# cat ../etc/gums-client.properties
  gums.location=https://localhost:8443/gums/services/GUMSAdmin
  gums.authz=https://localhost:8443/gums/services/GUMSAuthorizationServicePort

  Replace the localhost with the full machine name (even if you installed GUMS Admin on the same machine).

```
[root@gums bin]# vi ../etc/gums-client.properties [root@gums bin]# cat
../etc/gums-client.properties
gums.location=https://gums.mysite.com:8443/gums/services/GUMSAdmin
gums.authz=https://gums.mysite.com:8443/gums/services/GUMSAuthorizationServicePort
```

You might need to set the permission correctly: GUMS can be run only by admins and, due to the file privileges on the logs and configuration files, you will need to set the username and group accordingly. You will need to change permission on the directory:

```
[root@gums bin]# chown username:groupname .. -R
```

where groupname is a group that contain all the admins.

- Test the service by generating a mapfile at the command line

  ```
  [root@gums bin]# su - username [username@gums bin]# grid-proxy-init [username@gums bin]#
  ./gums generateGridMapfile testing.test.gov
  ```

  You should get the same response you got from the web server.

- You will have noticed that there are two command in the bin directory, plus a cron script: gums is meant to be run by an admin, and uses your proxy; gums-host and gums-host-cron are meant to be run using the host credentials, and they have limited functionalities (retrieve maps and mappings only). To make GUMS generate the maps periodically, just link the cron script in /etc/cron.hourly.

Congratulations you installed GUMS Admin. To make it actually useful, now you need to go back on the server and write an XML policy file. Please refer to the rest of the GUMS documentation. As an example of configuration, this is a configuration file suitable for OSG:

```
<gums>
  <persistenceFactories>
    <persistenceFactory name='mysql'
className='gov.bnl.gums.MySQLPersistenceFactory' jdbcDriver='com.mysql.jdbc.Driver'
jdbcUrl='jdbc:mysql://*your_server*/GUMS_0_7' user='gums' password='*password*'
autoReconnect='true' />
  </persistenceFactories>
  <adminUserGroup className='gov.bnl.gums.ManualUserGroup'
persistenceFactory='mysql' name='admins' />
  <groupMappings>
    <groupMapping name='grase' accountingVo='grase' accountingDesc='GRASE'>
      <userGroup className='gov.bnl.gums.VOMSGroup'
url='https://acdc.ccr.buffalo.edu:8443/edg-voms-admin-GRASE/services/VOMSAdmin'
persistenceFactory='mysql'
      name='grase' voGroup="/GRASE/grid3"
sslCertfile='/etc/grid-security/hostcert.pem'
sslKey='/etc/grid-security/hostkey.pem' ignoreFQAN="true"/>
      <accountMapping className='gov.bnl.gums.GroupAccountMapper' groupName='grase'
/>
    </groupMapping>
    <groupMapping name='gadu' accountingVo='gadu' accountingDesc='GADU'>
      <userGroup className='gov.bnl.gums.VOMSGroup'
url='https://cmsvo.fnal.gov:8443/edg-voms-admin/gadu/services/VOMSAdmin'
persistenceFactory='mysql'
      name='gadu' voGroup="/gadu/grid3users"
```

```
sslCertfile='/etc/grid-security/hostcert.pem'
sslKey='/etc/grid-security/hostkey.pem' ignoreFQAN="true"/>
      <accountMapping className='gov.bnl.gums.GroupAccountMapper' groupName='gadu'
/>
    </groupMapping>
    <groupMapping name='usatlas' accountingVo='usatlas' accountingDesc='ATLAS'>
      <userGroup className='gov.bnl.gums.VOMSGroup'
url='https://cmssrv08.fnal.gov:8443/edg-voms-admin/atlas/services/VOMSAdmin'
persistenceFactory='mysql'
      name='usatlas' voGroup="/atlas/usatlas"
sslCertfile='/etc/grid-security/hostcert.pem'
sslKey='/etc/grid-security/hostkey.pem' ignoreFQAN="true"/>
      <accountMapping className='gov.bnl.gums.GroupAccountMapper'
groupName='usatlas1' />
    </groupMapping>
    <groupMapping name='uscms' accountingVo='uscms' accountingDesc='CMS'>
      <userGroup className='gov.bnl.gums.VOMSGroup'
url='https://cmsvo.fnal.gov:8443/edg-voms-admin/uscms/services/VOMSAdmin'
persistenceFactory='mysql'
      name='uscms' voGroup="/uscms/grid3_users"
sslCertfile='/etc/grid-security/hostcert.pem'
sslKey='/etc/grid-security/hostkey.pem' ignoreFQAN="true"/>
      <accountMapping className='gov.bnl.gums.GroupAccountMapper'
groupName='uscms01' />
    </groupMapping>
    <groupMapping name='ivdgl' accountingVo='ivdgl' accountingDesc='iVDgL'>
      <userGroup className='gov.bnl.gums.VOMSGroup'
url='https://pki0.indiana.edu:8443/edg-voms-admin/g3dev/services/VOMSAdmin'
persistenceFactory='mysql'
      name='ivdgl' voGroup="/g3dev/group1"
sslCertfile='/etc/grid-security/hostcert.pem'
sslKey='/etc/grid-security/hostkey.pem' ignoreFQAN="true"/>
      <accountMapping className='gov.bnl.gums.GroupAccountMapper' groupName='ivdgl'
/>
    </groupMapping>
    <groupMapping name='sdss' accountingVo='sdss' accountingDesc='SDSS'>
      <userGroup className='gov.bnl.gums.VOMSGroup'
url='https://hotdog63.fnal.gov:8443/edg-voms-admin-sdss/services/VOMSAdmin'
persistenceFactory='mysql'
      name='sdss' voGroup="/sdss/grid3"
sslCertfile='/etc/grid-security/hostcert.pem'
sslKey='/etc/grid-security/hostkey.pem' ignoreFQAN="true"/>
      <accountMapping className='gov.bnl.gums.GroupAccountMapper' groupName='sdss'
/>
    </groupMapping>
    <groupMapping name='cdf' accountingVo='cdf' accountingDesc='CDF'>
      <userGroup className='gov.bnl.gums.VOMSGroup'
url='https://testbed008.cnaf.infn.it:8443/edg-voms-admin/cdf/services/VOMSAdmin'
persistenceFactory='mysql'
      name='cdf' sslCertfile='/etc/grid-security/hostcert.pem'
sslKey='/etc/grid-security/hostkey.pem' ignoreFQAN="true"/>
      <accountMapping className='gov.bnl.gums.GroupAccountMapper' groupName='cdf' />
    </groupMapping>
    </groupMappings>
    <hostGroups>
    <hostGroup className='gov.bnl.gums.WildcardHostGroup'
wildcard='*,*.*,*.*.*,*.*.*.*' groups='grase,gadu,usatlas,uscms,ivdgl,sdss,cdf' />
  </hostGroups>
</gums>
```

Problems?

If you have any problem, feel free to contact GUMS developers.

1.4.2 Installation guide

GUMS installation and walkthrough

This article describes all the components of GUMS and all the installation details: it is not meant as a quick installation guide. ***Please, refer to the quick installation guide if you prefer to get up and running and need the command line.*** *This guide assumes you are familiar with SQL, a bit of Java, Tomcat, ..., and that you are not interested in the exact command lines to write. It's meant for giving all the pieces of the puzzle so that an admin is comfortable with what GUMS does, and has an idea where to put his hands in case of problems.*

The pieces

Before beginning, we describe what components are there in GUMS.

- The service: the main component is a web application that consists of a Web Service interface (WS) and a Web Uuser Interface (WebUI). The same web application contains web pages for the admin to use through a browser, and a Web Service door that allows command line tools and other services to use GUMS functionalities. The Web Service part is a SOAP service built on Apache Axis. The web application is secured through SSL, that is also both WS and WebUI have the same transport level security. If GUMS is installed on a server without SSL, though part of the WebUI will be available, it won't proceed with any operations. The authorization is internal to GUMS: in the configuration you define a group of admins, who have full access; Services will have only read access to their mappings (i.e. map user or generate maps).
  The service was developed and tested on a Tomcat 5.0.28 + EGEE security installation, though it should work on any J2EE compliant web servers with SSL.
- The persistence layer: GUMS will need a place to store some temporary data, for example it caches the information it reads from the VO server, and some critical data, such as manual mappings the admin might want to define. At this time, the whole persistence layer is based on a MySQL server, though we plan to allow any RDBMS (the persistence layer needs some refining before it can be used 24/7, as part of that refining we will be able to use almost any JDBC compliant database). The idea, though, is that a site might want to integrate this part within their infrastructure. For example, at BNL we want to store all the critical data on our LDAP servers, which already contain most of the user accounts information. To integrate one needs only to implement a couple of classes and change the configuration file, all of which can be done at runtime. The suggestion is to evaluate GUMS on a MySQL back-end, and then, if desired, plan the integration.
- The admin tools: this is a set of command line tools to administer GUMS. They connect to the WS to perform the different tasks. All operation will be carried with the admin credentials, who will use the GRID proxy beforehand. All this activity will be logged on the server. Being the admin tools are a client to GUMS, they can be installed on the same host where GUMS is running or on a different host.
- The host tools: this is a set of command line tools to be installed on a generic host to retrieve the maps (i.e. grid-mapfile et al) , or to test the connectivity for the callout. All operation must be carried

with the host/service credentials, that is the certificate and private key. The host will be allowed only to retrieve information about its mappings. The CN of the host certificate and the hostname will be used, and must match to the GUMS setup. That is, for a host gatekeeper.mysite.com, with a CN=gatekeeper.mysite.com, GUMS will have to have a map for the host with the same name.

GUMS can also run in different modes, described in the GUMS modes of operation section. They are essentially variations in the user of GUMS. We won't talk extensively about those, as they are getting less relevant now, as we get closer and closer to GUMS 1.0. We will mention a few things, though, that affect the setup. It will be indicated by a "*Different mode note:*" warning.

The installation steps are:

- Preparing the persistance layer backend (i.e. MySQL for the standard installation)
- Installing the service and setting up a policy
- Installing the admin tools
- Installing the host tools on the target gatekeepers

Root vs non-root

GUMS can be installed as both root and non-root. The only issue is the host certificate: GUMS must be able to access a host/service certificate with its private key for authentication. Generally, it is located in /etc/grid-mapfile/hostcert.pem with root permissions. One could either set those permissions to a different user, or create another copy for gums. Such as /etc/grid-security/gumscert.pem.

Firewall and security considerations

GUMS doesn't require any port to be installed outside the firewall. The only requirement is to have an inbound TCP port opened on the GUMS server (default 8443), and an outbound port from all gatekeeper to that GUMS port.

All GUMS requests are over SSL. Grid certificates are used for authentication and authorization.

All access to GUMS is logged. Logs can be configured to use syslogd, which can be used to forward the logs to the cybersecurity department of the site.

Prepare the database

You will need a mysql server, with version 4.0.18 or greater installed. You can either install one from scratch (follow the instruction on mysql's site) or you can use an installation you have ready.

The gums service comes with a ./sbin/setupDatabase script which will create the database, and modify the policy file with the relevant information. The script will make you log in as root in mysql, will create a user and a database. You can see the script in ./var/sql/setupDatabase.mysql.

```
[root@www gums-service]# cat var/sql/setupDatabase.mysql
CREATE DATABASE GUMS_1_0;

GRANT ALL
  ON GUMS_1_0.*
  TO @USER@@'@SERVER@' IDENTIFIED BY '@PASSWORD@';
```

```
USE GUMS_1_0;

CREATE TABLE User (
  userID INTEGER AUTO_INCREMENT PRIMARY KEY,
  userDN VARCHAR(255) NOT NULL,
  userFQAN VARCHAR(255) DEFAULT NULL,
  registrationDate DATETIME NOT NULL,
  removalDate DATETIME DEFAULT NULL,
  userGroup VARCHAR(255) NOT NULL
);

CREATE TABLE UserAccountMapping (
  mappingID INTEGER AUTO_INCREMENT PRIMARY KEY,
  userDN VARCHAR(255) DEFAULT NULL,
  account VARCHAR(31) NOT NULL,
  startDate DATETIME DEFAULT NULL,
  endDate DATETIME DEFAULT NULL,
  userGroup VARCHAR(255) NOT NULL
);
```

The database structure is very simple: it's just a place to store lists of users (in the User table) and a list of mappings (UserAccountMapping). It's meant to be simple: it's direct consequence of the requirement that GUMS should be easily ported to other persistence mechanisms (i.e. LDAP, other site internal DB, ...). The first table will be mainly used to cache the values from the VO servers. For example, in the policy you will specify you want to map all the users from the ATLAS VO to the 'usatlas1' account: from time to time GUMS will query the VO server and store the list of users in the User table. For each mapping request, GUMS will look at its local copy instead of the remote VO server.

The User and UserAccountMapping tables are also critical for manual user groups and manual mappings. This means an admin is free to create a group of certificates, or a certificate to user mapping, to handle special cases. GUMS will have commands to add entries to these mapping, so you do not need to use the DB directly. You can if you want, though, for integration purposes. In any case, the use of these manual groups and mappings has a big effect: this is critical information that cannot be lost. Which means you will need to backup the server. This is a good candidate for integration: you might want to keep this information in the same information system you use for user account, as the information is connected.

To sum up: if you do not use manual groups and mappings, the information in the database can be regenerated at any time. If you do, GUMS has critical information, and you might want to make it safer through some kind of backup.

Once you created the database, you probably want to insert your DN in the User table within a group you will later use in GUMS as the admin group. You can use the ./sbin/addAdmin script, which will use the ./var/sql/addAdmin.mysql template:

```
[root@www sbin]# cat ../var/sql/addAdmin.mysql
USE GUMS_1_0;

INSERT INTO User SET userDN="@ADMINDN@", userGroup="admins";
```

This inserts a DN in a manual user group named "admins".

Once the database is prepared: you can now proceed to install the Service

Installing the service

GUMS is written in java, and requires java to be installed to run. It was developed against Sun JDK 1.4.2, but it should run on any 1.4.x and 1.5.x compliant JVM. If you need to install java, or learn more about it, refer to the documentation at http://java.sun.com .

The GUMS service is a standard J2EE application, which means it is a file (.war) that can be installed in any compliant engine. We provide a tarball containing a Tomcat 5.0.28 + EGEE security preconfigured, which is the configuration GUMS was developed against. The EGEE provides an SSL Socket Factory that Tomcat uses to create the SSL connections. The EGEE SSL is essentially standard SSL with the addition of Grid proxies. You can find more information about it if you look for "glite trustmanager". The bundled tomcat is modified in the following way:

- 4 more jars where put in the $CATALINA_HOME/server/lib directory:
  bcprov-jdk14-125.jar
  glite-security-trustmanager.jar
  glite-security-util-java.jar
  log4j-1.2.8.jar
  These contain the EGEE SSL socket factory and dependency
- In $CATALINA_HOME/conf the following file is added:
  log4j-trustmanager.properties
  Which is the logging configuration for the EGEE security
- The $CATALINA_HOME/conf/server.xml was modified. The following section is added:
  <Connector port="8443" maxThreads="150" minSpareThreads="25" maxSpareThreads="75" enableLookups="false" disableUploadTimeout="true" acceptCount="100" debug="0" scheme="https" secure="true"
  sSLImplementation="org.glite.security.trustmanager.tomcat.TMSSLImplementation"
  sslCAFiles="/etc/grid-security/certificates/*.0" crlFiles="/etc/grid-security/certificates/*.r0"
  sslCertFile="/etc/grid-security/hostcert.pem" sslKey="/etc/tomcat/hostkey.pem"
  log4jConfFile="/opt/tomcat-5.0.28-egeesec/conf/log4j-trustmanager.properties" clientAuth="true"
  sslProtocol="TLS" />

  This makes Tomcat listen on port 8443 for https, using the EGEE security. You will notice all the parameters for a Grid connection, and the configuration file that was added for EGEE security logging. The part declaring the port 8080 on http was closed.

Essentially, this is what is needed to setup EGEE security.

The service is available on the web site as a tar, though you should use it unpacked. It contains the web application in ./var/war, and the configuration files are for kept in the ./var/war/WEB-INF/classes directory, and loaded through classloading. Tomcat will need to refer to this war directory: the easiest way is to make a link to it from the Tomcat webapps directory. The service tar also contain a couple of scripts to setup the database, which you have seen before.

The configuration files are 2:

- log4j.properties. This is a standard log4j configuration, which will determine how the logging is implemented. This is loaded once when the server is started. To change it, the service will need to be restarted.

- gums.config. This is the policy file for GUMS, which determines how all users will be mapped to their local account. Also the access to the database is defined here. This can be changed at any point, and the service will pick it up. At any operation that needs the configuration, a check is performed to see if the file has been changed. If it is, this will trigger a configuration reload.

You can refer to the full documentation of the configuration files for the details.

Once you have setup the server and the web application, you can use your browser with your grid credential to connect to the WebUI and run a couple of commands.

Installing the client tools

The client tools come packaged in an rpm. The default location is /opt/gums, but the destination can be changed. Once you install it, you will see 4 directories:

- bin: contains the gums executables. They are shell script that prepares the java environment for GUMS. One critical part is the creation of the variable to handle the grid security. It is a single executable that accept different commands, like cvs does (i.e. "./gums mapUser ...", "./gums generateGridMapfile ...". The script has --help options that explains what are the current features.
- etc: contains the configuration file for admin, which is one. The only thing it contains the URL to the servlet that runs the web service. You have to point to your server before being able to run any commands
- lib: contains all the java libraries needed by GUMS
- var/log: contains the log files for the admin. There won't be much there, as all the functionalities are implemented on the service.

There are two main tools: gums and gums-host. The first one runs with the user credentials (proxy) and allows an admin to perform all the operations. The second runs with the host credentials (/etc/grid-security/host*.pem), and allows the host to retrieve the maps (grid-mapfiles and the OSG accounting map).

You can use ./bin/gums to:

- View the generations of the map
- Change the manual group and mapping
- Trigger a refresh of the groups (i.e. make GUMS contact the VO servers to refresh the local member lists)

You can use ./bin/gums-host to:

- Generate maps for the host: these will be based on the hostname
- Test the connectivity of the callout door.

Sample configuration

In the quick installation, you can find a sample configuration for OSG.

1.5 Configuration

Configuring GUMS

All the configuration of GUMS is in the following files:

- ./var/war/WEB-INF/classes/gums.config - holds all the configuration information for GUMS, including the database parameters, the policy, which VOs GUMS is going to contact and which gatekeeper will use which map.
- ./var/war/WEB-INF/classes/log4j.properties - this is the configuration for log4j, which is a well known library for logging in java. We will not discuss the details here: GUMS comes with this file preconfigured, and you can refer to the logging section for more details. If you want more information about how to modify this file, refer to the log4j documentation at http://logging.apache.org/log4j/docs/documentation.html .

1.5.1 gums.config

gums.config

This file contains the policy in an XML format. The syntax is meant to allow anybody to create his/her own components and integrate them just by dropping a jar in the lib directory. Therefore many components are defined by class names and bean properties. (If you are not a java programmer, a bean property is a getXxx/setXxx pattern, where xxx is the name of the property).

The XML file has this structure:

```
<gums>
  <persistenceFactories>
    <persistenceFactory/>
  </persistenceFactories>
  <adminUserGroup/>
  <groupMappings>
    <groupMapping>
      <userGroup/>
      <accountMapping/>
    </groupMapping>
    <groupMapping>
      <userGroup/>
      <compositeAccountMapping>
        <accountMapping/>
        <accountMapping/>
        <accountMapping/>
      </compositeAccountMapping>
    </groupMapping>
    ...
  </groupMappings>

  <hostGroups>
    <hostGroup/>
  </hostGroups>
</gums>
```

There are 3 sections:

- persistenceFactories - defines where the local data can be stored. For example, GUMS will keep a local copy of the VO listings: you can decide where to keep them. Each component that will need persistence will retrieve it through the factory. This allows to create a custom persistence layer for the facility.
- groupMappings - defines group of user and how they are mapped. A groupMapping is defined by two thins: a set of users (userGroup) and a policy for account mapping (accountMapping). Optionally, the policy can be composed by different policies (compositeAccountMapping)
- hostGroups - defines which groupMappings are used for the different hosts

persistenceFactories

This section just contains a list of persistenceFactory elements.

```
  <persistenceFactories>
    <persistenceFactory name='mysql'
className='gov.bnl.gums.MySQLPersistenceFactory' />
  </persistenceFactories>
```

persistenceFactory

The type of persitenceFactory is determined by the class which has to implement the PersistenceFactory interface. The basic attributes are:

Attribute	Description	Examples
name	The name that will be used by the other components to refer to this persistenceFactory.	mysql files ldap
className	The class that is going to provide the implementation for the persistence layer. It must implement gov.bnl.gums.Persistence Factory.	gov.bnl.gums.MySQLPersistence Factory org.mysite.HRDatabaseFactory

Other attributes are implementation specific.

All the elements that will be using the database, will need to set the 'persistence Factory' attribute to the name, and then provide a 'name' attribute that will identify which information to use. What that name means is implementation specific. For a database layer, for example, it could mean a table or a column value within a well known table; for a file layer it could mean the name of the file.

gov.bnl.gums.MySQLPersistence Factory

Currently, the only implementation provided is the MySQLPersistenceFactory. All the attributes are passed as properties to the database driver. For example:

```
 <persistenceFactory name='mysql' className='gov.bnl.gums.MySQLPersistenceFactory'
jdbcDriver='com.mysql.jdbc.Driver'
         jdbcUrl='jdbc:mysql://mydb.mysite.com/GUMS_0_7' user='gums'
password='mypass' autoReconnect='true' />
```

adminUserGroup

This defines the set of users that have admin privileges on GUMS. This entry has the same options as a userGroup entry. Refer to that part of the documentation.

groupMappings

This section will contain a list of groupMappings elements.

groupMapping

A group mapping is composed by two elements: a userGroup and a mapping, which can either be a compositeAccountMapping or an accountMapping.

Attribute	Description	Examples
name	The name that will be used by the other components to refer to this persistenceFactory.	atlas star phenix

userGroup

This element defines the list of people that will be part of this groupMapping. A userGroup is typically defined by a group on a VO server or on a database. This element corresponds to the UserGroup interface in the code, meaning you can provide your own logic. The basic attributes are:

Attribute	Description	Examples
className	The class that is going to provide the implementation for the user group. It must implement gov.bnl.gums.UserGroup.	gov.bnl.gums.LDAPGroup gov.bnl.gums.VOMSGroup gov.bnl.gums.ManualGroup

gov.bnl.gums.LDAPGroup

This class retrieves the list of members from an LDAP VO, as it is defined within LCG. The attributes available are:

Attribute	Description	Examples
server	The LDAP server from which to retrieve the information	grid-vo.nikhef.nl

query	The query to be used on the server.	ou=usatlas,o=atlas,dc=eu-datagrid,dc=org
		ou=People,o=atlas,dc=eu-datagrid,dc=org

persistence Factory	The persistence layer to be used to store locally the list of users. The string must be one of the names defined within the persistenceFactories section. GUMS doesn't contact the server at every request, but it keeps a local cache, which is refreshed from time to time	mysql

Attribute	Description	Examples
name	The name of the cache within the persistence factory. Refer to the specifics of the persistence factory itself.	atlas usatlas

For example:

```
    <userGroup className='gov.bnl.gums.LDAPGroup' server='grid-vo.nikhef.nl'
query='ou=People,o=atlas,dc=eu-datagrid,dc=org' persistanceFactory='mysql'
name='atlas' />
```

Retrieves all the user in the ATLAS VO LDAP server.

gov.bnl.gums.VOMSGroup

This class retrieves the list of members from an VOMS Server. The attributes available are:

Attribute	Description	Examples
url	The url of the web services for VOMS. Notice that it needs the full url of the service: it won't be constructed from the server name or vo.	https://vo.racf.bnl.gov:8443/edg-voms-admin/atlas/services/VOMSAdm
voGroup	The group defined within the VO.	/atlas/test /atlas/group/subgroup
persistence Factory	The persistence layer to be used to store locally the list of users. The string must be one of the names defined within the persistenceFactories section. GUMS doesn't contact the server at every request, but it keeps a local cache, which is refreshed from time to time	mysql
name	The name of the cache within the persistence factory. Refer to the specifics of the persistence factory itself.	atlasTest atlasGroupSubgroup

For example:

```
      <userGroup className='gov.bnl.gums.VOMSGroup'
url='https://vo.racf.bnl.gov:8443/edg-voms-admin/atlas/services/VOMSAdmin'
persistanceFactory='mysql'
                 name='atlas' voGroup="/atlas/test"
sslCertfile='/etc/grid-security/hostcert.pem'
sslKey='/etc/grid-security/hostkey.pem'/>
```

Retrieves all the user in the VOMS server at the specified url from the /atlas/test group. It also specifies which credentials should be used to contact the server.

gov.bnl.gums.ManualGroup

This class manages a group of identities stored in the persistence. Useful to handle special cases, for development testbed or for the list of admins. GUMS The attributes available are:

Attribute	Description	Examples
persistence Factory	The persistence layer to be used to store the list of users. The string must be one of the names defined within the persistenceFactories section.	mysql
name	The name of the group within the persistence factory. Refer to the specifics of the persistence factory itself.	test testbedA admins

For example:

```
      <userGroup className='gov.bnl.gums.ManualUserGroup' persistanceFactory='mysql'
name='testGroup' />
```

Selects the users stored manually in the testGroup group.

compositeAccountMapping

A compositeAccountMapping is a mapping policy made up by a list of policies. When a request to map a user comes, the composite mapper will forward the request to the first mapper in the list. If this fails, the request is forwarded to the second, and so on. This allows you to create a policy that has a default (the last element on the list) but allows special cases (the top element in the list).

This element simply contains a list of accountMapping elements.

accountMapping

An account mapping defines the logic with which the user credentials are mapped to the local account. The logic will be provided by a class implementing the goc.bnl.gums.AccountMapping interface.

Attribute	Description	Examples
className	The class that is going to provide the implementation for the mapping. It must implement gov.bnl.gums.AccountMapping.	gov.bnl.gums.ManualAccountMapper gov.bnl.gums.NISAccountMapper gov.bnl.gums.AccountPoolMapper gov.bnl.gums.GroupAccountMapper

gov.bnl.gums.NISAccountMapper

This class retrieves the NIS maps and tries to match the name from a certificate. Please, read the full documentation on the javadoc about this class before using it. The attributes available are:

Attribute	Description	Examples
jndiNisUrl	The url as defined in the Java JNDI driver, that allows to specify the NIS server and the domain.	nis://nis.bnl.gov/atlas

For example:

```
    <accountMapping className='gov.bnl.gums.NISAccountMapper2'
jndiNisUrl='nis://nis.mysite.org/domain' />
```

Uses the NIS map taken from the nis.mysite.org server for domain.

gov.bnl.gums.AccountPoolMapper

This class implements account pooling. Please refer to the account pool documentation for the full detailed description. The attributes available are:

Attribute	Description	Examples
TODO		

For example:

```
TODO
```

todo

gov.bnl.gums.GroupAccountMapper

This class maps all users to the same account. The attributes available are:

Attribute	Description	Examples
groupName	The name of the account	atlas testAccount

For example:

```
    <accountMapping className='gov.bnl.gums.GroupAccountMapper' groupName='atlas' />
```

Maps everyone to the atlas account.

hostGroups

This section contains a list of host groups. To determine to which group a particular host is part, GUMS start from the first one in the list and stops at the first match.

hostGroup

A hostGroup defines a group of hosts and which groupMappings will be used. This element corresponds to the HostGroup interface. This allows to retrieve hosts lists from other components of the facility, for example an information service.

Attribute	Description	Examples
className	The class that is going to provide the implementation for the hostGroup. It must implement gov.bnl.gums.HostGroup.	gov.bnl.gums.WildcardHostGroup
groups	A list of groupMappings, in the order of preference. To determine which group should be used for a particular user, GUMS will start from the beginning of the list until it finds a match. Therefore, if there would be more than one match (i.e. a user is part of more groups) the first one in the list is used.	group1,group2

gov.bnl.gums.WildcardHostGroup

This class represent a set of hosts defined by a hostname wildcard. For example, *.mysite.org would include all the hosts which end in mysite.org. The attributes that can be set for this class are:

Attribute	Description	Examples
wildcard	The wildcard for the set of hosts to be included. The wildcard is a string where * can be substituted with any character, except '.'. That is, *.bnl.gov wouldn't match myhost.usatlas.bnl.gov.	myhost.mysite.org atlas*.mysite.org *.atlas.mysite.org

For example:

```
    <hostGroup className="gov.bnl.gums.WildcardHostGroup"
wildcard='*.usatlas.bnl.gov' groups='gridex,sdss,uscms,usatlasGroup,btev,ligo,ivdgl'
/>
```

Maps the hosts in the usatlas subdomain at BNL to the list of groups, which will have been defined in the groupMappings section.

1.5.2 Examples

How to Configure GUMS

This article goes through different scenarios of GUMS configuration. See the gums.config file documentation before reading this.

Mapping people from a VOMS server

Use VOMSGroup. For example, here we get all people from the ATLAS server in the USATLAS (/atlas/usatlas) group, and we map them to a default account 'usatlas'.

```
<groupMapping name='usatlas' accountingVo='usatlas' accountingDesc='ATLAS'>
    <userGroup className='gov.bnl.gums.VOMSGroup'
    url='https://voms.cern.ch:8443/edg-voms-admin/atlas/services/VOMSAdmin'
    persistenceFactory='mysql' name='usatlas'
    voGroup="/atlas/usatlas" sslCertfile='/etc/grid-security/hostcert.pem'
    sslKey='/etc/grid-security/hostkey.pem' ignoreFQAN="true"/>
    <accountMapping className='gov.bnl.gums.GroupAccountMapper' groupName='usatlas1'
/>
</groupMapping>
```

The userGroup.persistenceFactory and userGroup.name tell us to use mysql to store the information. The VOMS server is contacted only when updateGroup is done. We can make all hosts at our site use this mapping by adding in hostGroup.groups the value declared in groupMapping.name, as shown below:

<hostGroup className='gov.bnl.gums.WildcardHostGroup' wildcard='*.mysite.com' groups='...,usatlas,...' />

Allowing ad-hoc list of people

You might need to allow some people without adding a VO, e.g., for testing. You do that by using a ManualGroup. For example:

```
<groupMapping name='testers'>
    <userGroup className='gov.bnl.gums.ManualUserGroup'
    persistenceFactory='mysql' name='testers' />
    <accountMapping className='gov.bnl.gums.GroupAccountMapper' groupName='test' />
</groupMapping>
```

The userGroup.persistenceFactory and userGroup.name?? tells us the list will be stored in mysql. In this example, we add the group name to the beginning of the hostGroup.groups list, so it will override all the other group, as shown below:.

```
<hostGroup className='gov.bnl.gums.WildcardHostGroup' wildcard='*.mysite.com'
groups='testers,...' />
```

You can then add and/or remove people using the commands './bin/gums manualGroup-add' and/or './bin/gums manualGroup-remove'.

Using account pools

GUMS does not create accounts. First you'll have to create accounts and make them known to the gatekeeper and worker nodes. Here's an example of configuring a pool of accounts for the ATLAS VO:

```
<groupMapping name='usatlas' accountingVo='usatlas' accountingDesc='ATLAS'>
    <userGroup className='gov.bnl.gums.VOMSGroup'
    url='https://voms.cern.ch:8443/edg-voms-admin/atlas/services/VOMSAdmin'
    persistenceFactory='mysql' name='usatlas'
    voGroup="/atlas/usatlas" sslCertfile='/etc/grid-security/hostcert.pem'
    sslKey='/etc/grid-security/hostkey.pem' ignoreFQAN="true"/>
    <accountMapping className='gov.bnl.gums.AccountPoolMapper'
persistenceFactory='mysql' name='bnlPool' />
</groupMapping>
```

This will tell GUMS to look for the list of available accounts in the bnlPool stored in mysql. You can set different pools for different groups, or the same pool for some or all groups. Here we configure the pool to be used by a group of hosts:

```
<hostGroup className='gov.bnl.gums.WildcardHostGroup' wildcard='*.mysite.com'
groups='...,usatlas,...' />
```

Let's leave the GUMS configuration file for a moment and put some accounts in the pool (the accounts must be known to the gatekeeper and the worker nodes):

```
> ./bin/gums pool-addRange mysql bnlPool grid0000-199
```

GUMS will now assign accounts to DNs as requests come in from the gatekeeper. (If you generate a user-to-VO map for accounting, though, all DNs will be assigned an account immediately.)

Composite mapping

It can be useful to assign an account to a DN by hand. For example, we may generally want to assign accounts to usatlas from a pool, but there are a few special cases in which we need more control. We would use <compositeAccountMapping> for these cases, as shown here:

```
<groupMapping name='usatlas' accountingVo='usatlas' accountingDesc='ATLAS'>
    <userGroup className='gov.bnl.gums.VOMSGroup'
    url='https://voms.cern.ch:8443/edg-voms-admin/atlas/services/VOMSAdmin'
    persistenceFactory='mysql' name='usatlas'
    voGroup="/atlas/usatlas" sslCertfile='/etc/grid-security/hostcert.pem'
    sslKey='/etc/grid-security/hostkey.pem' ignoreFQAN="true"/>
    <compositeAccountMapping>
        <accountMapping className='gov.bnl.gums.ManualAccountMapper'
persistenceFactory='mysql' name='bnlMap' />
        <accountMapping className='gov.bnl.gums.AccountPoolMapper'
persistenceFactory='mysql' name='bnlPool' />
    </compositeAccountMapping>
</groupMapping>
```

The compositeAccount allows you specify a list of mappers (e.g., ManualAccountMapper, AccountPoolMapper). If the first doesn't return an account, the second is tried, and so on. In this example, we first use the ManualAccountMapper that takes the bnlMap map from mysql. If the DN in question isn't mapped there, we fall back on the pool.

You can add/remove entries in the map using the commands './bin/gums manualMapping-add' and './bin/gums manualMapping-remove'.

Mapping based on groups/roles

You'll need to create different group mappings for the different roles. This first example maps people with the VOMS attribute voGroup of "/atlas/usatlas" to the account usatlas1. Notice that userGroup.ignoreFQAN is missing; this means that if the VOMS attribute doesn't match, the next group is checked.

```
<groupMapping name='usatlas' accountingVo='usatlas' accountingDesc='ATLAS'>
    <userGroup className='gov.bnl.gums.VOMSGroup'
    url='https://voms.cern.ch:8443/edg-voms-admin/atlas/services/VOMSAdmin'
    persistenceFactory='mysql' name='usatlas'
    voGroup="/atlas/usatlas" sslCertfile='/etc/grid-security/hostcert.pem'
    sslKey='/etc/grid-security/hostkey.pem'/>
    <accountMapping className='gov.bnl.gums.GroupAccountMapper' groupName='usatlas1'
/>
</groupMapping>
```

The second maps people with the VOMS attributes voGroup of "/atlas" and voRole "production" to the account usatprod.

```
<groupMapping name='usatlasProd' accountingVo='usatlas' accountingDesc='ATLAS'>
    <userGroup className='gov.bnl.gums.VOMSGroup'
    url='https://voms.cern.ch:8443/edg-voms-admin/atlas/services/VOMSAdmin'
    persistenceFactory='mysql' name='usatlasProd'
    voGroup="/atlas" voRole="production"
    sslCertfile='/etc/grid-security/hostcert.pem'
    sslKey='/etc/grid-security/hostkey.pem'/>
    <accountMapping className='gov.bnl.gums.GroupAccountMapper' groupName='usatprod'
/>
</groupMapping>
```

And then we define the hostGroup and include both groups:

```
<hostGroup className='gov.bnl.gums.WildcardHostGroup' wildcard='*.mysite.com'
groups='usatlasProd,usatlas,...' />
```

1.6 GUMS Client Commands

GUMS Client command line tools

Here we describe all the commands available use and their intended use.

List of tools

All GUMS commands are located in the ./bin directory of your GUMS installation. There are three tools:

- ./bin/gums - this script will provide access to all GUMS functionalities, including adding and removing people to manually managed groups, generating maps for any hosts and forcing GUMS to refresh the user lists from the VO servers. This script will run using user credentials, and will need a valid proxy certificate. The DN for the user must be in the admin group, or GUMS will respond with an authorization denied.
- ./bin/gums-host - this script will provide access to only the maps for the current host. One can either retrieve the maps (mapfile and osg inverse map) or test the callout door requesting the mapping for a particular user credential (DN and FQAM). The script will use the host credentials, so it will be typically executed as root. The host credentials will need to match the name of the map requested, and no special authorization needs to be set in GUMS.
- ./bin/gums-host-cron - this is a pre-made cron job that retrieves the maps and installs them at the proper position. Will generate the mapfile and install it in /etc/grid-security/grid-mapfile, and if $VDT_LOCATION, the inverse map for accounting will be saved in $VDT_LOCATION/monitoring/grid3-user-vo-map.txt.

1.6.1 ./bin/gums

./bin/gums

We describe the ./bin/gums which provides all the administrative functions for GUMS. We'll describe the use of the commands, giving examples, but for the full options please refer to the --help output of the commands themselves.

./bin/gums consists of a set of command line tools which will be run under the user GRID credentials. The user must be part of the GUMS admins.

Authentication and authorization

./bin/gums runs using the user credentials, not the host credentials. This means that the user will need to run grid-proxy-init, or voms-proxy-init, to generate a valid proxy certificate. The certificate will also need to be in the admin group, or you will get an authorization denied. To add a person in the admin group, refer to the GUMS Service documentation.

./bin/gums commands

The script provides many commands, which you can list just by running the script with no arguments:

```
[carcassi@atestgk01 ~]$ /opt/gums/bin/gums
usage: gums command [command-options]
Commands:
  generateGrid3UserVoMap - Generate grid3-user-vo-map.txt for a given host.
  generateGridMapfile - Generate grid-mapfile for a given host.
  manualGroup-add - Includes a DN in a group.
  manualGroup-remove - Removes a DN from a group.
  manualMapping-add - Adds a DN-to-username in a mapping.
  manualMapping-remove - Removes a DN from a mapping.
  mapUser - Local credential used for a particular user.
  mapfileCache-refresh - Regerates mapfiles in the cache.
  pool-addRange - Adds accounts to an account pool.
  updateGroups - Contact VO servers and retrieve user lists.
For help on any command:
  gums command --help
```

You can then retrieve the full syntax of each command by calling it with the --help options. For example:

```
[carcassi@atestgk01 ~]$ /opt/gums/bin/gums mapUser --help
usage: gums mapUser [-s SERVICE] [-n TIMES] [-t NREQUESTS] [-b] [-f FQAN]
            [-i FQANISSUER] USERDN1 [USERDN2] ...
Maps the grid identity to the local user.
Options:
 -f,--fqan <arg>      Fully Qualified Attribute Name, as it would be
                      selected using voms-proxy-init; no extended information by
```

```
default
 -t,--timing <arg>    enables timing, grouping the requests. For example,
                      "-t 100" will give you timing information on 100 requests at a
time
 -b,--bypassCallout   connects directly to GUMS instead of using the
                      callout
 -i,--issuer <arg>    Fully Qualified Attribute Name Issuer, that is the
                      DN of the VOMS service that issued the attribute certificate
 -n,--ntimes <arg>    number of times the request will be repeated
   --help             print this message
 -s,--service <arg>   DN of the service
```

Please, refer to the help on the command line for the full syntax of the commands.

Service mapping generation commands

This set of commands can be used by the admin to check how the mapping across the services is maintained. One can look how the maps generated by GUMS look like, and check to which local user any Grid identity is mapped. These are the same as the ./bin/gums-host commands, but, since they run with admin credential, they can access mapping to all hosts.

gums mapUser

With this command an admin can check the mapping of a specific identity, including the VOMS extended proxy FQAN. This allows to check if the user is mapped to the correct account when using different VO roles. It issues a mapping request as the callout to the gatekeeper does, which is very helpful to diagnose problems.

Here are a couple of examples:

```
[carcassi@atestgk01 ~]$ /opt/gums/bin/gums mapUser \
  -s  "/DC=org/DC=doegrids/OU=Service/CN=mygk.mysite.com" \
  "/DC=org/DC=doegrids/OU=People/CN=Gabriele Carcassi 12345"
LocalId[userName: grid12345]
```

This examples asks the GUMS server what account would the certificate be mapped on the mygk.mysite.com gatekeeper.

One can also ask what account would be used if the user would come in with a particular role.

```
[carcassi@atestgk01 ~]$ /opt/gums/bin/gums mapUser \
  -s  "/DC=org/DC=doegrids/OU=Service/CN=mygk.mysite.com" \
  -f  "/myvo/Role=role1" \
  -i  "/DC=org/DC=doegrids/OU=Service/CN=voms.mysite.com" \
  "/DC=org/DC=doegrids/OU=People/CN=Gabriele Carcassi 12345"
LocalId[userName: special1]
```

Here we see the same user being assigned a different account.

gums generateGridMapfile

This commands allows to retrieve a grid-mapfile for a specific host. Be careful: generating a map will force all the policy to be explored, and it might have undesired side effects. For example, when using the account mapping, this will force assigning an account to each user, even if they are never going to come on site.

Here are a couple of examples:

Without arguments, it will generate the map for the hostname assigned to the machine where the command is run. Notice that, in special configuration, the name of the machine and the name specified in the host certificate might different.

```
[carcassi@atestgk01 ~]$ /opt/gums/bin/gums generateGridMapfile
#---- members of vo: usatlas ----#
"/C=CH/O=CERN/OU=GRID/CN=Frederik Orellana 5894" usatlas1
"/C=CH/O=CERN/OU=GRID/CN=Michela Biglietti 4798" usatlas1
"/C=CH/O=CERN/OU=GRID/CN=Miguel De Oliveira Branco 2423" usatlas1
"/C=CH/O=CERN/OU=GRID/CN=Shulamit Moed 9840" usatlas1
"/DC=org/DC=doegrids/OU=People/CN=Alden Stradling 409738" usatlas1
"/DC=org/DC=doegrids/OU=People/CN=Aldo Saavedra 942457" usatlas1
"/DC=org/DC=doegrids/OU=People/CN=Alexandre V Vaniachine 778117" usatlas1
...
```

As an admin, you can generate the map for any host by giving the name as an argument

```
[carcassi@atestgk01 ~]$ /opt/gums/bin/gums generateGridMapfile gatekeeper.mysite.com
#---- members of vo: usatlas ----#
"/C=CH/O=CERN/OU=GRID/CN=Frederik Orellana 5894" usatlas1
"/C=CH/O=CERN/OU=GRID/CN=Michela Biglietti 4798" usatlas1
"/C=CH/O=CERN/OU=GRID/CN=Miguel De Oliveira Branco 2423" usatlas1
"/C=CH/O=CERN/OU=GRID/CN=Shulamit Moed 9840" usatlas1
"/DC=org/DC=doegrids/OU=People/CN=Alden Stradling 409738" usatlas1
"/DC=org/DC=doegrids/OU=People/CN=Aldo Saavedra 942457" usatlas1
"/DC=org/DC=doegrids/OU=People/CN=Alexandre V Vaniachine 778117" usatlas1
...
```

gums generateGrid3UserVoMap

This command allows to retrieve the inverse map used by Grid3/OSG accounting. Be careful: generating a map will force all the policy to be explored, and it might have undesired side effects. For example, when using the account mapping, this will force assigning an account to each user, even if they are never going to come on site.

Here are a couple of examples:

Without arguments, it will generate the map for the hostname assigned to the machine where the command is run. Notice that, in special configuration, the name of the machine and the name specified in the host certificate might different.

```
[carcassi@atestgk01 ~]$ /opt/gums/bin/gums generateGrid3UserVoMap test.mysite.com
#User-VO map
# #comment line, format of each regular line line: account VO
# Next 2 lines with VO names, same order, all lowercase, with case (lines starting
wi
th #voi, #VOc)
#voi usatlas ivdgl ligo btev uscms sdss gridex grase
#VOc ATLAS iVDgL LIGO BTeV CMS SDSS GRIDEX GRASE
#---- accounts for vo: usatlas ----#
usatlas1 usatlas
#---- accounts for vo: ivdgl ----#
ivdgl ivdgl
...
```

As an admin, you can generate the map for any host by giving the name as an argument

```
[carcassi@atestgk01 ~]$ /opt/gums/bin/gums generateGrid3UserVoMap
gatekeeper.mysite.com
#User-VO map
# #comment line, format of each regular line line: account VO
# Next 2 lines with VO names, same order, all lowercase, with case (lines starting
wi
th #voi, #VOc)
#voi usatlas ivdgl ligo btev uscms sdss gridex grase
#VOc ATLAS iVDgL LIGO BTeV CMS SDSS GRIDEX GRASE
#---- accounts for vo: usatlas ----#
usatlas1 usatlas
#---- accounts for vo: ivdgl ----#
ivdgl ivdgl
...
```

Manual groups and mappings managements commands

This set of commands allows the admin to add and remove entries from a manual group or mapping. These are resident in your database, and are managed "by hand" by the admin to handle special cases or customizations. To have any effect, these groups/mappings must be set in the configuration file. For example, the following uses a manually define set of users to be mapped to the account "myacc".

```
    <groupMapping name='example1' accountingVo='myvo' accountingDesc='myVo'>
      <userGroup className='gov.bnl.gums.ManualUserGroup' persistenceFactory='mysql'
                 name='group1' />
      <accountMapping className='gov.bnl.gums.GroupAccountMapper' groupName='myacc'
/>
    </groupMapping>
```

The manual user group is identified by the persistence (mysql) and the name of the group (group1). These are the parameter that will also be used for the command line.

For a manual mapping, the following defines a manual mapping for the users taken from a VOMS server.

```
    <groupMapping name='example2' accountingVo='myvo' accountingDesc='myVo'>
      <userGroup className='gov.bnl.gums.VOMSGroup'
```

```
url='https://voms.mysite.com:8443/edg-voms-admin-myvo/services/VOMSAdmin'
persistenceFactory='mysql'
                 name='myvo' voGroup="/myvo"
sslCertfile='/etc/grid-security/hostcert.pem'
sslKey='/etc/grid-security/hostkey.pem' ignoreFQAN="true"/>
      <accountMapping className='gov.bnl.gums.ManualAccountMapper'
persistenceFactory='mysql' name='map1'/>
    </groupMapping>
```

Also there, the manual map is defined by the persistence (mysql) and a name (map1).

gums manualGroup-add

To add a member in a manual group:

```
[carcassi@atestgk01 ~]$ /opt/gums/bin/gums manualGroup-add mysql group1
"/DC=org/DC=doegrids/OU=People/CN=Gabriele Carcassi 12345"
```

gums manualGroup-remove

To remove a member from a manual group:

```
[carcassi@atestgk01 ~]$ /opt/gums/bin/gums manualGroup-remove mysql group1
"/DC=org/DC=doegrids/OU=People/CN=Gabriele Carcassi 12345"
```

gums manualMapping-add

To add an entry in a map:

```
[carcassi@atestgk01 ~]$ /opt/gums/bin/gums manualMapping-add mysql map1
"/DC=org/DC=doegrids/OU=People/CN=Gabriele Carcassi 12345" carcassi
```

gums manualMapping-remove

To remove an entry from a map:

```
[carcassi@atestgk01 ~]$ /opt/gums/bin/gums manualMapping-remove mysql map1
"/DC=org/DC=doegrids/OU=People/CN=Gabriele Carcassi 12345"
```

Other commands

gums updateGroups

This command will force GUMS to recontact all the VO servers and retrieve a new list of people.

```
[carcassi@atestgk01 ~]$ /opt/gums/bin/gums updateGroups
```

gums pool-addRange

This is a utility to add a range of accounts to a pool. This won't actually create accounts, which must be already created; it just makes GUMS aware that they exist and can be used. It assumes that accounts are something like grid0002, grid0003, ... That is with a fixed part at the beginning and an increasing number at the end.

This command will add the accounts grid0020, grid0021, ... until grid0040 to the pool

```
[carcassi@atestgk01 ~]$ /opt/gums/bin/gums pool-addRange mysql pool1 grid0020-40
```

1.6.2 ./bin/gums-host

./bin/gums-host

We describe the ./bin/gums-host which provides all the host tools for managing a gatekeeper with GUMS. We'll describe the use of the commands, giving examples, but for the full options please refer to the --help output of the commands themselves.

./bin/gums-host consists of a set of command line tools which will be run under the host GRID credentials.

Authentication and authorization

./bin/gums-host runs using the host credentials, not the user credentials. This means that the user running gums-host will need to be able to read the host credentials. A host will be able to only access the maps relative to the host. That is, the hostname included in the host certificate will need to match to the maps within GUMS.

./bin/gums-host commands

The script provides many commands, which you can list just by running the script with no arguments:

```
[root@mygk bin]$ ./gums-host
usage: gums-host command [command-options]
Commands:
  generateGrid3UserVoMap - Generate grid3-user-vo-map.txt for this host.
  generateGridMapfile - Generate grid-mapfile for this host.
  mapUser - Local credential used for a particular user.
For help on any command:
  gums-host command --help
```

You can then retrieve the full syntax of each command by calling it with the --help options. For example:

```
[root@mygk bin]# ./gums-host mapUser --help
usage: gums mapUser [-s SERVICE] [-n TIMES] [-t NREQUESTS] [-b] [-f FQAN]
            [-i FQANISSUER] USERDN1 [USERDN2] ...
Maps the grid identity to the local user.
Options:
 -f,--fqan <arg>      Fully Qualified Attribute Name, as it would be
                      selected using voms-proxy-init; no extended information by
default
 -t,--timing <arg>    enables timing, grouping the requests. For example,
                      "-t 100" will give you timing information on 100 requests at a
time
 -b,--bypassCallout   connects directly to GUMS instead of using the
                      callout
 -i,--issuer <arg>    Fully Qualified Attribute Name Issuer, that is the
                      DN of the VOMS service that issued the attribute certificate
 -n,--ntimes <arg>    number of times the request will be repeated
    --help            print this message
 -s,--service <arg>   DN of the service
```

Please, refer to the help on the command line for the full syntax of the commands.

Service mapping generation commands

The only set of commands available for a host are the ones to retrieve mapping information. One can retrieve the maps generated by GUMS, and check to which local user any Grid identity is mapped. These are the same as the ./bin/gums commands, but, since they run with host credential, they can access only that particular host mapping.\

gums-host mapUser

With this command an admin can check the mapping of a specific identity, including the VOMS extended proxy FQAN. This allows to check if the user is mapped to the correct account when using different VO roles. It issues a mapping request as the callout to the gatekeeper does, which is very helpful to diagnose problems. Remember that the service name has to match the name used in the credentials.

Here are a couple of examples:

```
[root@mygk bin]# ./gums-host mapUser \
  -s  "/DC=org/DC=doegrids/OU=Service/CN=mygk.mysite.com" \
  "/DC=org/DC=doegrids/OU=People/CN=Gabriele Carcassi 12345"
LocalId[userName: grid12345]
```

This examples asks the GUMS server what account would the certificate be mapped on the mygk.mysite.com gatekeeper.

One can also ask what account would be used if the user would come in with a particular role.

```
[root@mygk bin]# ./gums-host mapUser \
  -s  "/DC=org/DC=doegrids/OU=Service/CN=mygk.mysite.com" \
  -f  "/myvo/Role=role1" \
  -i  "/DC=org/DC=doegrids/OU=Service/CN=voms.mysite.com" \
  "/DC=org/DC=doegrids/OU=People/CN=Gabriele Carcassi 12345"
LocalId[userName: special1]
```

Here we see the same user being assigned a different account.

gums generateGridMapfile

This commands allows to retrieve a grid-mapfile for the host. Be careful: generating a map will force all the policy to be explored, and it might have undesired side effects. For example, when using the account mapping, this will force assigning an account to each user, even if they are never going to come on site.

Here are a couple of examples:

Without arguments, it will generate the map for the hostname assigned to the machine where the command is run.

```
[root@mygk bin]# ./gums-host generateGridMapfile
#---- members of vo: usatlas ----#
"/C=CH/O=CERN/OU=GRID/CN=Frederik Orellana 5894" usatlas1
"/C=CH/O=CERN/OU=GRID/CN=Michela Biglietti 4798" usatlas1
"/C=CH/O=CERN/OU=GRID/CN=Miguel De Oliveira Branco 2423" usatlas1
"/C=CH/O=CERN/OU=GRID/CN=Shulamit Moed 9840" usatlas1
"/DC=org/DC=doegrids/OU=People/CN=Alden Stradling 409738" usatlas1
"/DC=org/DC=doegrids/OU=People/CN=Aldo Saavedra 942457" usatlas1
"/DC=org/DC=doegrids/OU=People/CN=Alexandre V Vaniachine 778117" usatlas1
...
```

By default, the hostname used by ./bin/gums-host is simply the name of the current host. There are some cases in which this differ from the certificate host: for example, if an alias was setup for a service, or if a gatekeeper has more than one interface. In these cases, one has to specify the name of the host.

```
[root@mygk bin]# ./gums-host generateGridMapfile gatekeeper.mysite.com
#---- members of vo: usatlas ----#
"/C=CH/O=CERN/OU=GRID/CN=Frederik Orellana 5894" usatlas1
"/C=CH/O=CERN/OU=GRID/CN=Michela Biglietti 4798" usatlas1
"/C=CH/O=CERN/OU=GRID/CN=Miguel De Oliveira Branco 2423" usatlas1
"/C=CH/O=CERN/OU=GRID/CN=Shulamit Moed 9840" usatlas1
"/DC=org/DC=doegrids/OU=People/CN=Alden Stradling 409738" usatlas1
"/DC=org/DC=doegrids/OU=People/CN=Aldo Saavedra 942457" usatlas1
"/DC=org/DC=doegrids/OU=People/CN=Alexandre V Vaniachine 778117" usatlas1
...
```

gums generateGrid3UserVoMap

This command allows to retrieve the inverse map used by Grid3/OSG accounting. Be careful: generating a map will force all the policy to be explored, and it might have undesired side effects. For example, when using the account mapping, this will force assigning an account to each user, even if they are never going to come on site.

Here are a couple of examples:

Without arguments, it will generate the map for the hostname assigned to the machine where the command is run.

```
[root@mygk bin]# ./gums-host generateGrid3UserVoMap test.mysite.com
#User-VO map
# #comment line, format of each regular line line: account VO
# Next 2 lines with VO names, same order, all lowercase, with case (lines starting
wi
th #voi, #VOc)
#voi usatlas ivdgl ligo btev uscms sdss gridex grase
#VOc ATLAS iVDgL LIGO BTeV CMS SDSS GRIDEX GRASE
```

```
#---- accounts for vo: usatlas ----#
usatlas1 usatlas
#---- accounts for vo: ivdgl ----#
ivdgl ivdgl
...
```

By default, the hostname used by ./bin/gums-host is simply the name of the current host. There are some cases in which this differ from the certificate host: for example, if an alias was setup for a service, or if a gatekeeper has more than one interface. In these cases, one has to specify the name of the host.

```
[root@mygk bin]# ./gums-host generateGrid3UserVoMap gatekeeper.mysite.com
#User-VO map
# #comment line, format of each regular line line: account VO
# Next 2 lines with VO names, same order, all lowercase, with case (lines starting
wi
th #voi, #VOc)
#voi usatlas ivdgl ligo btev uscms sdss gridex grase
#VOc ATLAS iVDgL LIGO BTeV CMS SDSS GRIDEX GRASE
#---- accounts for vo: usatlas ----#
usatlas1 usatlas
#---- accounts for vo: ivdgl ----#
ivdgl ivdgl
...
```

1.7 Logging

Log

GUMS is designed with 3 logs in mind: developer's log, administrator's log, site security log. This means that you won't find the same things in all of them, and you shouldn't. For example, say that GUMS connects to a VO server to retrieve a list of users, and the VO server replies with an empty list. From the developer's perspective the code has worked fine; but from an administrator's perspective it's probably the sign that something not going well.

The logs come with a predefined configuration, which is what we describe here. To know more of the details, especially how to change the configuration, refer to the logging implementation.

Administrator's log

This log is meant for the maintainer of GUMS at a particular site. He is responsible of installing and configuring GUMS. To manage the mapping by keeping all the information up-to-date.

The log is placed under the service directory (/opt/gums-service/logs/gums-resource-admin.log). The log can also be configured to be forwarded by mail in case of error. This is particularly useful as the admins can be informed right away of any problem. Look at the log4j.properties configuration file.

The log includes every command that is being executed by any admins. This allow the administrator to keep full control of what is happening, together with a history of what has happened to be able to troubleshoot automatic procedures. The main features are:

- All successful commands are logged as INFO with both input and output parameters
- All unsuccessful commands (including failure do to authorization) are logged as ERROR

Developer's log

The developer log is meant for someone developing the code or fixing bugs.

The log is located under the service directory (/opt/gums-service/logs/gums-developer.log).

Site security log

The site security log is meant for the cybersecurity department of a lab. It includes [TODO check requirement and implement] all the information for auditing the GUMS service. This information will be limited to accesses to the service that are going to modify the state of the service. All the access to the information will be typically already be logged at the gatekeeper.

The log can be configured to be forwarded to the AUTHPRIV facility of syslogd. To enable logging to the syslogd deamon, you have to modify the log4j.properties and make sure it allows logging from the network. To enable logging from the network, you need to start syslogd with -r option.[root@atlasgrid13 log]# cat /etc/sysconfig/syslog # Options to syslogd # -m 0 disables 'MARK' messages. # -r enables logging from remote machines # -x disables DNS lookups on messages recieved with -r # See syslogd(8) for more details SYSLOGD_OPTIONS="-r -m 0" # Options to klogd # -2 prints all kernel oops messages twice; once for klogd to decode, and # once for processing with 'ksymoops' # -x disables all klogd processing of oops messages entirely # See klogd(8) for more details KLOGD_OPTIONS="-x"

The reason is that Apache log4j SyslogAdapter can only log through the network (to allow portability), even if you are logging to the localhost.

Another possibility is to log directly to a remote server: you can do that by modifying the log4j.properties configuration file in the service.

1.7.1 Logging implementation

Log implementation

All information in GUMS is logged through the apache commons logging package. The implementation used in GUMS is apache log4j. To change the logging implementation you have to refer to the commons.logging implementation. Be aware that some library that GUMS uses may not be as well behaved in regard to logging.

The configuration is controlled by the log4j.properies file. This is a normal log4j configuration file: refer to the log4j manual for more information.

GUMS using the follow conventions for log names:

- The developer's log uses one log for each different class, with the name being the class name. Given GUMS package structure, "gov.bnl.gums" contains the whole development log for GUMS. This allows the develop to filter the log of the code he is working on.
- The admin log uses the log named "gums.resourceAdmin"
- The site security log is at "gums.siteAdmin"

Administrator's log

This log is meant for whoever is maintaining GUMS installation at a particular site. The log is designed to be used in different ways: on standard error, in a log file and in e-mails. E-mails will get from WARN up, the standard error will receive from INFO up and the log can go down to TRACE. The breakdown on the logging level is:

TODO: The admin log still needs a little thinking

- FATAL - GUMS is unable to operate: no functionalities are available. For example, a configuration file was not written correctly.
- ERROR - A particular operation failed or was incomplete. For example, the CMS VOMS server couldn't be contacted, so it's members weren't refreshed (even though the ATLAS group was); the NIS server didn't respond, so it is impossible to generate the grid-mapfile for atlasgrid25, though the grid-mapfile for atlasgrid26 could be generated since it doesn't require the NIS information.
- WARN - A condition that hints to a misconfiguration or incorrect usage. For example, a VO server returned no users.
- INFO - A condition that might hint to a problem, but is not critical per se. For example, the NIS mapper couldn't find a match. A big difference is that INFO doesn't trigger a mail, while WARN does. It is preferable to log many similar problems as info and then send a WARN to actually send the mail.
- DEBUG - Not used

- TRACE - Not used

Developer's log

The developer log is meant for someone developing the code or fixing bugs. Each class will use the log named as their full class name. The breakdown on the logging level is:

- FATAL - An exception or an inconsistency that forces GUMS to terminate or not function at all. For example, a configuration file was not written correctly or couldn't be found.
- ERROR - An exception or an inconsistency that doesn't allow GUMS to complete a particular operation or part of it. For example, the CMS VOMS server couldn't be contacted, so it's members weren't refreshed (even though the ATLAS group was); the NIS server didn't respond, so it is impossible to generate the grid-mapfile for atlasgrid25, though the grid-mapfile for atlasgrid26 could be generated since it doesn't require the NIS information.
- WARN - An exception or an inconsistency that is not necessarily going to affect functionalities, or an error condition that was recovered. For example, a particular cache was found to be out of synch and was rebuilt.
- INFO - The successful completion of a macro-event (i.e. something that happens only once in a while). For example, the configuration file was read, the server was started. Typically used to debug configuration problems.
- DEBUG - The attempt or successful completion of a smaller event. For example, a query was executed, a user was mapped. Typically one shouldn't have more than one DEBUG statement in a method.
- TRACE - Shows the internal execution of the code. As a contrast, building a query would be at this level. Inside method logging should be done at this level

Site security log

The site security log will log all accesses.

- INFO - Will log all the "write" accesses
- DEBUG - Will log all the "read" accesses

1.8 Integration

GUMS extension and site integration

All the main components of GUMS are developed against interfaces with minimal coupling to GUMS itself, allowing a site to rewrite those components to interface their systems. In this article we will describe these interfaces and provide some integration examples. We link to the online GUMS code for examples; ***if you choose to print this, you might also want to print the code to which the online version of this article links.***

GUMS doesn't require integration: it can work fine by itself. But if a site requires integration such that GUMS communicates with the site's information systems, it is easy to do, it just requires a little knowledge of Java. You don't need to know about the internal workings of GUMS. You can write code (external to GUMS) to deal with special mapping circumstances, and write the policy file to tell GUMS when to run it. Here are examples of site-specific needs:

- Store all the information that GUMS uses for mapping in a separate system that's used for the rest of the site's accounts (e.g., in LDAP, in an Oracle or MySQL database, and so on)
- Use some pre-existing software to perform the mapping (e.g.,. as part of the user information database, a user was already able to select the grid certificate at the site.) In this case you're using GUMS for its role-based capabilities and for integration with OSG; it's just the glue between grid and local site. This is good for transition situations, for maintaining compatibility with existing systems.
- Use a different database for group information (e.g.,. the site wants to map his admins in a different way, and wants to take the list of admins directly from its databases or store them in LDAP instead of the GUMS MySQL)
- Use some other information service to decide which service should use which mapping (i.e. one wants to set on all production machines a particular mapping, and the list of production machines is stored within their own information service)

If you have a use case that is not site-specific, we may be willing to either help in the development or to distribute it as part of GUMS. Go to the GUMS site and use our mailing lists to contact us!

Changing storage for GUMS data

Suppose you already have a user management system that keeps some grid-to-username mapping and you want GUMS to use it. Or suppose that you want to tightly couple the GUMS pool account system with your site LDAP. Or suppose that you want to perform special mapping on a list of users taken directly from some database. How would you do it?

In GUMS there is a PersistenceFactory class which is responsible for creating all the objects that write to/from a specific external storage system (as opposed to MySQL which we'll consider internal). We've implemented this in GUMS in the form of a MySQLPersistanceFactory class which implements the logic of how to write/read to/from the default MySQL implementation. A site can implement its own

PersistenceFactory class to define where the data is written to and read from.

The PersistenceFactory is just a factory class used to create the objects that actually implement the persistence layer. There are several different kinds of these objects, e.g.,:

- UserGroupDB is used by GUMS to cache lists of users so that GUMS doesn't have to contact the VO server every time.
- ManualAccountMapperDB is used by the ManualAccountMapper to store a mapping table.

You do not need to implement the all the different kinds of classes, only the kinds you need. For those you do choose not to implement, you can use:

```
throw new java.lang.UnsupportedOperationException("...");
```

This way, when gums tries to use a method that implements it (due to a configuration or usage error), GUMS generates an error. This is also true for any other method you do not want to implement (e.g., you might not want GUMS to modify the information in your site databases).

You can use the MySQLPersistenceFactory as a trace: it creates an inner class for any interface it needs to implement, and when asked to give a persistence object it just creates an instance of the appropriate inner class. You can thus implement a ManualAccountMapper2DB that reads the site user management system, an AccountPoolMapperDB that reads account information from the site LDAP, and/or a ManualUserGroupDB that reads the list of users from your database.

Creating a mapping policy

A site can create its own policies for the mapping. For examples, we suggest that you look at the mapping code that comes with GUMS. A mapping is particularly useful for sites that have a pre-existing user management system which already contains some certificate-to-username mapping. Through an extension, GUMS can be made to use that mapping.

All the mappings implement the AccountMapper interface. The only method you're required to implement is mapUser, which returns the username given the user credential.

The GroupAccountMapper interface maps all the users to the same account. The account to which they're mapped is set through the groupName property (notice the getGroupName and setGroupName) in the configuration file. For example:

```
<accountMapping className='gov.bnl.gums.GroupAccountMapper' groupName='test' />
```

sets the groupName to "test". You can create any property you like: while reading the configuration, GUMS looks at your class for a name match [This is actually provided by the Apache Jakarta Commons Digester library].

We suggest you develop and test a new class by itself first, without running it in GUMS. You can have a main method, or a set of unit tests, to simulate some requests using the mapUser method, and see that it behaves correctly. Once you have done that, you can prepare a jar, and put it in the lib directory of the GUMS service. You will have to restart the service, as tomcat creates the list of available jars when the service is started. You can change the policy configuration file, instead, at any time.

Creating a group

You can create your own groups. A group is a list of users that get mapped using the same mapping criteria. Suppose, for example, that a site wants to use a new VO server which is not supported by GUMS. Or suppose that the site wants to grant all its admins a different mapping, and wants to get this list directly from their LDAP system, so that there is only one place to keep updated. All of these can be supported by creating a group.

Look at the at the UserGroup class: a group is essentially a function that is able to tell: "is this person in the group?". The isInGroup() function is going to check, by whatever means , if the Grid Identity is within that group.

There is a getMemberList(). This is actually optional: if you have an EveryoneGroup, for example, you can't name everyone. In that case, you should use something like:

```
throw new java.lang.UnsupportedOperationException("Group cannot be enumerated.");
```

The catch is that GUMS won't be able to generate a grid-mapfile for those hosts that make use of this group; you can only use the gatekeeper callout functionality of GUMS with groups.

The updateMembers() function is intended for groups requiring that GUMS access some remote service (e.g., VOMS); this cannot be done on a per request basis. For cases like these, one should implement the updateMembers() function to retrieve and store the data on a site-local service. If you do this, you might want to use a UserGroupDB to store the information, so that it integrates with the rest of persistence. You can look at the VOMSGroup class for an example.

1.9 Operation modes

1.10 FAQ

Frequently Asked Questions

General

1. Is GUMS being used in production anywhere?
2. I hear GUMS allows you to have different mappings on different gatekeeper. Why do you want to do that? Doesn't it complicate things?

Using GUMS

1. Does GUMS have to run as root?

Building GUMS

Comparing GUMS with other tools

1. What's the difference between GUMS and VOMS (or VOMRS)?
2. What's the difference between using GUMS and using grid-mapfiles?
3. What's the difference between GUMS and edg-mkgridmap?
4. What's the difference between GUMS and LCMAPS?

General

General

Is GUMS being used in production anywhere?

Yes, GUMS is being used at BNL for RHIC and ATLAS gatekeepers.

I hear GUMS allows you to have different mappings on different gatekeeper. Why do you want to do that? Doesn't it complicate things?

At BNL we have different gatekeepers for different experiments , each of which has its own individual requirements. Furthermore, each experiment may have some gatekeepers in production and others in test; these might require slightly different configurations. It also comes in handy for troubleshooting (an admin may want to temporarily change his mapping to a user's if the user is having troule with an operation) and when testing/implementing a new policy. Allowing different mappings is only one way to address these situations. For example, they could be addressed by implementing roles in the VO servers. It turns out that GUMS actually helps in keeping the mapping identical at all gatekeepers: this is what we generally want to do at BNL and one of the reasons we developed GUMS. But we still need to cope with the "irregularities" that a production environment may present.

Using GUMS

Using GUMS

Does GUMS have to run as root?

No. GUMS runs under tomcat, and will run fine no matter what userid you use to run tomcat.

Building GUMS

Building GUMS

Comparing GUMS with other tools

Comparing GUMS with other tools

What's the difference between GUMS and VOMS (or VOMRS)?

First of all, GUMS is a **site** tool and VOMS is a **VO** tool. I.e., you have a BNL GUMS, whereas you have an ATLAS VOMS. A VO uses VOMS to keep a list of members and their roles within the organization. A site uses GUMS to maintain the mapping between members' GRID credentials (certificates) and local site credentials (e.g., UNIX accounts).

GUMS can contact VOMS to retrieve the list of VO users that require a particular mapping. For example, if the GUMS configuration says: "all ATLAS members should be mapped to the 'atlas' account" then GUMS would contact the ATLAS VOMS server to find out who all the ATLAS members are.

What's the difference between using GUMS and using grid-mapfiles?

If you want to use grid-mapfiles, GUMS can be used to generate them, as can various other external tools. Usually grid-mapfiles are generated according to the information present in the VOMS servers. For example, the external tool would contact the ATLAS VOMS, download the list of current users, and add them to the grid-mapfile.

Using grid-mapfiles by itself is typically good only in testing environments. GUMS provides an alternative. GUMS can be configured to provide dynamic mapping, thereby making grid-mapfiles unnecessary. In this configuration, the gatekeeper contacts GUMS directly when it needs a mapping, instead of consulting a grid-mapfile.

What's the difference between GUMS and edg-mkgridmap?

Edg-mkgridmap and the GUMS host (client) tool can both be used to create a grid-mapfile for a host. Edg-mkgridmap is a client (gatekeeper) tool only; it connects to the VOMRS databases, downloads info and creates a grid-mapfile for that gatekeeper only. GUMS has server and client portions. The GUMS server (which serves all the gatekeepers at the site) connects to the VOMRS databases, downloads info, and performs the mapping, then the GUMS host tool creates the site-wide grid-mapfile. GUMS provides a way to centrally manage resource access and the mapfile generation. Edg-mkgridmap can be used to contact GUMS to retrieve an already prepared grid-mapfile.

In addition, GUMS supports a richer, more complex and flexible policy than does edg-mkgridmap. You can also use GUMS for both grid-mapfile generation and in conjuction with a gatekeeper (or service) callout. For a small site with a simple configuration, edg-mkgridmap might be a simpler solution. For a bigger site, with a more complicated environment, GUMS offers more control and flexibility.

What's the difference between GUMS and LCMAPS?

WARNING: I am not an expert in LCMAPS. This is my understanding of the differences.

The short (and not 100% precise) answer is: GUMS is a Policy Decision Point while LCMAPS is a Policy Enforcement Point. The longer answer is: GUMS allows you to set a policy at the site level for all your gatekeepers or resources. It's a service that waits for and responds to questions like: "Who should I map this guy to?". It doesn't actually enforce the mapping. In fact, GUMS cannot stand by itself; it needs to have other software contact it to either retrieve a grid-mapfile or to request a specific mapping (e.g., the GUMS host tools, or the gatekeeper callout).

LCMAPS, on the other hand, is inside the gatekeeper (or the gridftp server). It implements the callout, it determines and enforces the mapping. There is one for every gatekeeper; there is no central mapping, and no central policy. You configure each gatekeeper individually.

They are two different things, even though they implement some similar functionalities. In fact, LCMAPS could be used as the PDP for GUMS (i.e., LCAMPS could connect to GUMS as part of its decision process).

1.11 Changes

Release History

Version	Date	Description
1.0.1	2005-04-07	
1.0.0	2005-03-16	
0.7.1	2005-02-01	
0.7	2005-01-14	
0.6.1	2004-08-10	
0.6	2004-07-01	
0.5	2004-05-20	
undetermined	before March 2004	

Get the RSS feed of the last changes

Release 1.0.1 - 2005-04-07

Type	Changes	By
☛	Including setupDatabase with the modification made for VDT	carcassi
✚	gums-client rpm now obsoletes gums-host and gums-admin (no need to remove package).	carcassi
🔨	Error code from scripts was incorrect: that caused cron job to update maps in case of error too.	carcassi

Release 1.0.0 - 2005-03-16

Type	Changes	By
☛	Repackaging of the client components: merged admin and host, and named it client.	carcassi

Type	Changes	By
🔨	Fixed minor issues with authorization obligations for storage system.	mlorch
🔨	Double add to a manual group now fails.	carcassi
➕	Script that creates the database and changes the configuration accordingly.	carcassi
🖐	Repackaging of the service component.	carcassi
🔨	Server identity is now gums host name.	carcassi
🔨	GUMS accept new style proxies.	carcassi
🖐	The accounts in the pool are used in alphabetical order.	carcassi
🔨	Inverse map is now generated exploring all the DN/FQAN combinations.	carcassi
🔨	Gridmapfile is generated simulating users with no FQAN.	carcassi
➕	LDAP integration for primary gid change at BNL once an account is assigned.	carcassi

Release 0.7.1 - 2005-02-01

Type	Changes	By
🔨	Log names review so that they both client and server can stay (through links) in the same directory.	carcassi
🖐	NIS update is done every hour and is now thread safe.	carcassi
🔨	Log file permission for the command line tools are set so multiple users can use it (important for admin).	carcassi
➕	GUMS host can now be used for stress testing and timing the server response.	carcassi
➕	Added connection pooling on mysql server.	carcassi
🔨	Solved a race condition that would make GUMS hang in some circumstances.	carcassi

Type	Changes	By
➕	AuthZ callout without GT3, both client and server stubs.	mlorch
➕	Added code for Privilege Project in GUMS repository and build process.	carcassi

Release 0.7 - 2005-01-14

Type	Changes	By
🔧	Better logging: server logs all commands with both input and output	carcassi
🔧	More complete command line interface	carcassi
➕	Web service implementation	carcassi
🔧	HostWildcards can be more than one, comma separated	carcassi
➕	Support for VOMS Fully Qualified Attribute names	carcassi
➕	AuthZ service to be contacted by Globus callout	carcassi
➕	Support for grid3-user-vo-map.txt generation	carcassi
➕	Many many other refinements...	carcassi

Release 0.6.1 - 2004-08-10

Type	Changes	By
➕	Nightly build and reporting with Maven	carcassi
✖	Removed all the old code from 0.6	carcassi
🔧	Better log system: logs for developer, resource admin and site admin in place	carcassi

Type	Changes	By
➕	Ability to retrieve groups from within a VOMS server (finally)	carcassi
🔨	No more duplication in the mapfiles	carcassi
👉	Improved database caching for grid-mapfile: you specify on the server which gatekeeper maps should be generated	carcassi
➕	Improved error handling (i.e. a failed update on one group doesn't block the others)	carcassi
➕	Installation through RPMs (cron jobs installed automatically)	carcassi
➕	Unit tests to Grid3 VOs included	carcassi
🔨	LDAP access improved: can access LCG dev VO	carcassi

Release 0.6 - 2004-07-01

Type	Changes	By
➕	XML configuration file for mapping policy	carcassi
➕	Log infrastructure	carcassi
👉	More flexible architecture	carcassi
➕	Decoupled grid-mapfile generation from database caching for distribution on gatekeeper	carcassi
➕	Web interface to generate grid-mapfiles and map users	carcassi
👉	Better command line interfaces (feel like Unix commands)	carcassi

Release 0.5 - 2004-05-20

Type	Changes	By
🔨	GUMS in production at BNL	carcassi
➕	`NISMapper`retrieves the GECOS field and matches with certificate CN.	carcassi
➕	Architecture to allow different type of mappings for different hosts	carcassi

Release undetermined - before March 2004

Type	Changes	By
➕	Script to fetch user from VOMS	dtyu
➕	Script to map user to local account	tomw