GUMS v.0.7

Project Documentation

Table of Contents

1 **Manual**

1.1 Introduction 1

1.2 Installation 3

1.3 Logging 8

1.3.1 Logging implementation 9

1.4 Configuration 11

1.4.1 gums.config 12

1.4.2 db.properties 17

1.4.3 hosts.conf 18

1.5 Commands 19

1.6 FAQ 23

1.7 Changes 24

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

1.1 Introduction

Introduction

The first thing you should do is ask yourself: is GUMS the right tool for what I want to do?

GUMS solves one and just one function: mapping grid certificates to local identities (i.e. UNIX account). A job comes to a site with a GRID credential (the PROXY certificate). The site resources might not use GRID credentials natively, and will use some different mechanism to identify users, such as UNIX accounts, Kerberos principals, and the like. The gatekeeper will need to map the GRID credential to the site credential. GUMS is a service that provide that mapping: tells you which site user the GRID user should be using. Notice that it doesn't authenticate for you: it doesn't 'su', it doesn't retrieve Kerberos credentials. It just tells the gatekeeper which site credentials should get. The gatekeeper is still in charge of enforcing the site mapping established by GUMS.

What is GUMS?

GUMS stand for Grid User Management System, and it is a Grid User Identity System. It is a site service, usually used only within a site, that provides a mapping between GRID identity (i.e. user certificate) to site identity (i.e. Unix user account names). It does so by using a policy defined in an XML file by the resource admin.

Within the policy you specify something like: I want these gatekeepers to allow these set of users mapped according to this rule. For example, on 'atlas*.mysite.org' I want to allow all the USATLAS users mapped to a group account named 'usatlas01', and I want to allow all the ATLAS users mapped to accounts taken from a site pool.

GUMS then provide 3 main commands:

- Refresh the vo members (i.e. by contacting the VO servers)
- Generate the grid-mapfile for a particular gatekeeper
- Return which local user should be used for a specific user on a specific gatekeeper

GUMS past, present and future

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 the code was refactored to allow the business logic to be called either from command line or from a web application, opening the door to a GT3/4 service implementation.

Current work is going toward a web application that would provide both a web interface for the administrator and a web service that implement the OGSA AuthZ interface. This is done within the

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

Privilege Project, a joint project between USCMS and USATLAS.

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

1.2 Installation

Pre-requites

Prepare the database

The first to do is to prepare the database. There is currently no script to do it: it's something we plan on doing.

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. Log in as root and run the following script:

```
CREATE DATABASE GUMS_0_6_1;

GRANT ALL
  ON GUMS_0_6_1.*
  TO gums@'<gumsserver.site.gov>' IDENTIFIED BY '<password>';

GRANT SELECT
  ON GUMS_0_6_1.*
  TO gums@'%.site.gov' IDENTIFIED BY '<readPassword>';

USE GUMS_0_6_1;

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

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

CREATE TABLE HostGridMapfile (
  mapfileID INTEGER AUTO_INCREMENT PRIMARY KEY,
  hostname VARCHAR(255) NOT NULL,
  mapFile MEDIUMTEXT NOT NULL
);
```

in which you should replace <gumsserver.site.gov> with the machine on which you will run the GUMS

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

server component. You will also notice that GUMS requires a read-only access from within your site: the client component will need to contact the db server to retrieve the mapping. This is will no longer necessary once GUMS will run as a service.

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.

Install the server

Go to Download at GUMS site and retrieve the server rpm. Install it.

```
[root@gums root]# rpm -Uvh gums-server-cli-0.6.1-build200408101627.noarch.rpm
Preparing... ########################################### [100%]
1:gums-server-cli ########################################### [100%]
```

As a default, GUMS is installed in /opt/gums

```
[root@gums root]# cd /opt/gums
[root@gums gums]# ls
etc lib sbin var
```

GUMS is now installed, but it needs to be configured

db.properties

You need to create the db.properties file in ./etc. There is an example provided:

```
[root@gums gums]# cat etc/db.properties.example
jdbc.driver = com.mysql.jdbc.Driver
jdbc.url = jdbc\:mysql\://mysql.database.org/GUMS
user = gums
password = passwd
```

You will want to change the server name (mysql.database.org), the database name (GUMS to GUMS_0_6_1) and the username and password. If you need to customize more the connection (i.e. use SSL), this set of properties is passed as it is to the mysql driver. You can, therefore, use any of the mysql

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

jdbc configuration parameters.

gums.config

Now you need to create a policy for your site: this part is the most time consuming. You have an example within ./etc, and you have the manual for more details. A simple policy, to get you started, could be:

```
<?xml version="1.0" encoding="UTF-8"?>
<gums>
  <persistanceFactories>
    <persistenceFactory name='mysql'
className='gov.bnl.gums.MySQLPersistanceFactory' />
  </persistanceFactories>
  <groupMappings>
    <groupMapping name='atlas'>
      <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' />
      <accountMapping className='gov.bnl.gums.GroupAccountMapper2'
groupName='usatlas1' />
    </groupMapping>
  </groupMappings>
  <hostGroups>
    <hostGroup className="gov.bnl.gums.WildcardHostGroup" wildcard='*.mysite.org'
groups='atlas' />
  </hostGroups>
</gums>
```

This will allow all the atlas members at mysite.org, mapped to the usatlas1 account. Notice that if you use subdomains, you will need a double wildcard: *.*.mysite.org (i.e. the wildcard doesn't work on the '.').

hosts.conf

This is a simple text file that tells for which hosts the grid-files will be generated this is connected to the read-only access we talked about before: each host will connect to the database to retrieve the mapping. This is a temporary solution until GUMS will run as a service.

Testing the server

Now you are ready to test the server. First, you need to tell GUMS to refresh it's groups by connecting to all the VO servers:

```
[root@gums gums]# ./sbin/gums-updateGroups
```

If everything goes well, you shouldn't see any message. After that you can make it generate a map on the fly, to confirm that everything has gone well:

```
[root@gums gums]# ./sbin/gums-generateGridMapfile -n machine.mysite.org
#---- members of vo: atlas ----#
"/C=AT/O=UIBK/OU=HEPG/CN=Reinhard Bischof/Email=reinhard.bischof@uibk.ac.at"
```

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

```
usatlas1
"/C=CA/O=Grid/OU=lps.umontreal.ca/CN=Georges Azuelos" usatlas1
"/C=CA/O=Grid/OU=lps.umontreal.ca/CN=Rashid Mehdiyev" usatlas1
"/C=CA/O=Grid/OU=lps.umontreal.ca/CN=Wen Chao Chen_01/Email=chen@lps.umontreal.ca"
usatlas1
"/C=CA/O=Grid/OU=phys.ualberta.ca/CN=Bryan L Caron" usatlas1
"/C=CA/O=Grid/OU=phys.uvic.ca/CN=Ashok Agarwal" usatlas1
"/C=CA/O=Grid/OU=physics.carleton.ca/CN=Wade Hong" usatlas1
"/C=CA/O=Grid/OU=physics.utoronto.ca/CN=Leslie Groer" usatlas1
"/C=CA/O=Grid/OU=physics.utoronto.ca/CN=Rachid Mazini" usatlas1
...
```

You can now generate the maps for all the hosts in the hosts.conf file by running:

```
[root@atlasgrid13 gums]# ./sbin/gums-mapfileCache-refresh
```

GUMS has also installed a cron job that every night will refresh the groups and refresh the mapfiles. You will need to run the previous commands only in those cases you are making modification on the configuration and you either want to make sure everything is correct, or you want to push the new maps right away.

For all the other commands and options, refer to the manual.

Install the client

Now that the server component is running, you can install the client. Retrieve the client rpm and install it:

```
[root@gums root]# rpm -Uvh gums-client-0.6.1-build200408101627.noarch.rpm
Preparing... ########################################### [100%]
1:gums-client ########################################### [100%]
```

db.properties

You will need to configure the database, as you did for the server. The file works in exactly the same way.

```
[root@gums gums]# cat etc/db.properties.example
jdbc.driver = com.mysql.jdbc.Driver
jdbc.url = jdbc\:mysql\://mysql.database.org/GUMS
user = gums
password = passwd
```

Testing the client

Now you can see whether the client is reading the map:

```
[root@machine root]# /opt/gums/bin/gums-mapfileCache-retrieve
```

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

```
#---- members of vo: atlas ----#
"/C=AT/O=UIBK/OU=HEPG/CN=Reinhard Bischof/Email=reinhard.bischof@uibk.ac.at"
usatlas1
"/C=CA/O=Grid/OU=lps.umontreal.ca/CN=Georges Azuelos" usatlas1
"/C=CA/O=Grid/OU=lps.umontreal.ca/CN=Rashid Mehdiyev" usatlas1
"/C=CA/O=Grid/OU=lps.umontreal.ca/CN=Wen Chao Chen_01/Email=chen@lps.umontreal.ca"
usatlas1
"/C=CA/O=Grid/OU=phys.ualberta.ca/CN=Bryan L Caron" usatlas1
"/C=CA/O=Grid/OU=phys.uvic.ca/CN=Ashok Agarwal" usatlas1
"/C=CA/O=Grid/OU=physics.carleton.ca/CN=Wade Hong" usatlas1
"/C=CA/O=Grid/OU=physics.utoronto.ca/CN=Leslie Groer" usatlas1
"/C=CA/O=Grid/OU=physics.utoronto.ca/CN=Rachid Mazini" usatlas1
...
```

The command will take the name of the machine, contact the database and retrieve the appropriate map. The rpm also installed a cron job that every 6 hour retrieves a new copy of the map and installs it in /etc/grid-security/grid-mapfile

Problems?

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

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

1.3 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.

[TODO: the log is still not logging everything it should]

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. He will need to be informed whether there are any problems retrieving information from the VO server, if the mapping is not functioning properly for some users, and so on.

The log is sent to two places: the standard error and ./var/log/gums-admin.log [TODO: needs to be configured like that. WARN to stderr, INFO to the file].

Developer's log

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

The log is located ./var/log/gums-developer.log.

Site security log

The site security log is meant for the cybersecurity department of a lab. It will include 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.

As per default configuration, the site security log is forwarded to the AUTHPRIV facility of syslogd. To enable logging to the syslogd deamon, you have to make sure it allows logging from the network. The reason is that Apache log4j SyslogAdapter can only log through the network (to allow portability), even if you are logging to the localhost.

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

1.3.1 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

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

- 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

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

1.4 Configuration

Configuring GUMS

There are three files used by GUMS for the configuration.

- gums.config - holds the policy used by GUMS to create the mapping. It defines where to get the VO members, how should they be mapped, and which groups can access which gatekeepers
- hosts.conf - contains a list of hosts for which to generate the mapfile for the cache
- db.properties - holds the configuration for the mysql database

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

1.4.1 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>
  <persistanceFactories>  <!-- with the 'a', it's a typo in the code: next release
will be fixed -->
    <persistenceFactory/>  <!-- with the 'e', no type here -->
  </persistanceFactories>

  <groupMappings>
    <groupMapping>
      <userGroup/>
      <accountMapping/>
    </groupMapping>
    <groupMapping>
      <userGroup/>
      <compositeAccountMapping>
        <accountMapping/>
        <accountMapping/>
        <accountMapping/>
      </compositeAccountMapping>
    </groupMapping>
    ...
  </groupMappings>

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

There are 3 sections:

- persistanceFactories - 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)

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

- hostGroups - defines which groupMappings are used for the different hosts

persistanceFactories

This section just contains a list of persistenceFactory elements.

```
  <persistanceFactories>
    <persistenceFactory name='mysql'
className='gov.bnl.gums.MySQLPersistanceFactory' />
  </persistanceFactories>
```

persistenceFactory

The type of persitenceFactory is determined by the class which has to implement the PersistanceFactory 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.PersistanceFactory.	gov.bnl.gums.MySQLPersistanceFactory org.mysite.HRDatabaseFactory

Other attributes are implementation specific.

All the elements that will be using the database, will need to set the 'persistanceFactory' 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.MySQLPersistanceFactory

Currently, the only implementation provided is the MySQLPersistanceFactory. No other attributes are specified: the class will look in db.properties for all the connections parameters. For example:

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

groupMappings

This section will contain a list of groupMappings elements.

groupMapping

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

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
persistanceFactory	The persistence layer to be used to store locally the list of users. GUMS doesn't contact the server at every request, but it keeps a local cache, which is refreshed from time to time	grid-vo.nikhef.nl
name	The name of the cache within the persistence factory. Refer to the specifics of the persistence factory itself.	ou=usatlas,o=atlas,dc=eu-datagrid,dc=org ou=People,o=atlas,dc=eu-datagrid,dc=org

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

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

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.AccountMapping2 interface.

Attribute	Description	Examples
className	The class that is going to provide the implementation for the mapping. It must implement gov.bnl.gums.AccountMapping2. (The '2' comes out of the evolution in GUMS code; it will be taken away in the next refactoring step).	gov.bnl.gums.ManualAccountMapper2 gov.bnl.gums.NISAccountMapper2 gov.bnl.gums.AccountPoolMapper2 gov.bnl.gums.GroupAccountMapper2

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:

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

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.

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

1.4.2 db.properties

db.properties

This file contains the configuration for the connection to the database. It is the same syntax for both the server and the client. It is basically the property file that will be sent to the JDBC driver plus two other properties that allow to select the driver. The database syntax was only tested on MySQL, and this is the only database supported out of the box.

This is scheme temporary: once we make a transition to a grid service, there will be no need for the client to use the database. Also, we plan to reimplement the persistence layer using EJB to allow clustering. In that case the datasource will be configured through JNDI as any other J2EE application, and the database supported will be all the ones supported by the application server used.

```
jdbc.driver = com.mysql.jdbc.Driver
jdbc.url = jdbc\:mysql\://hostname/GUMS
user = gums
password = mypass
```

Those accustomed to JDBC will recognize the standard jdbc syntax for the driver and url. More information about the mysql driver connection parameters should be found at http://dev.mysql.com/doc/connector/j/en/index.html

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

1.4.3 hosts.conf

hosts.conf

This file contains the list of hosts for which gums is going to create the mapfiles in the cache. As of now, the server components stores the maps in the database when the gums-mapfilecache-refresh is executed. This command relies on hosts.conf to know for which hosts to create the maps. The client will use the host of the machine on which is installed when running gums-mapfilecache-retrieve. If the client has multiple names due to multiple interfaces, you should use the name you see once logged in the hosts (i.e. run 'hostname').

The configuration file is simply a text file with the full name of each host on each line. For example:

```
grid01.mysite.org
grid02.mysite.org
testgrid.mysite.org
```

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

1.5 Commands

List of commands

The commands available on the client and on the server side are different. Typically the client side will just contain few of them to allow to retrieve the generated maps.

Server side

gums-cron-updateGroups

It's the cron script that is linked in /etc/cron.hourly. It will run once a day and will call gums-updateGroups and gums-mapfileCache-refresh to update the maps

gums-generateGridMapfile

```
usage: gums-generateGridMapfile [-n hostname] [-f filename]
Create a grid-mapfile for a specified host.
Options:
 -n,--name    the hostname for which to generate the grid-mapfile;
              localhost by default
 -f,--file    saves in the specified file; prints to the console by
              default
 -h,--help    print this message
```

For example:

```
[root@atlasgrid13 sbin]# ./gums-generateGridMapfile -n atlasgrid10.usatlas.bnl.gov
#---- 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=Shulamit Moed 9840" usatlas1
"/C=PT/O=UP/OU=FE/CN=Miguel Branco/Email=msbranco@fe.up.pt" usatlas1
"/DC=org/DC=doegrids/OU=People/CN=Alden Stradling 409738" usatlas1
"/DC=org/DC=doegrids/OU=People/CN=Alexandre V Vaniachine 778117" usatlas1
"/DC=org/DC=doegrids/OU=People/CN=Ambreesh Gupta 104392" usatlas1
"/DC=org/DC=doegrids/OU=People/CN=Catalin Lucian Dumitrescu 766712" usatlas1
"/DC=org/DC=doegrids/OU=People/CN=Daniel Levin 43685" usatlas1
"/DC=org/DC=doegrids/OU=People/CN=Dantong Yu 542086" usatlas1
"/DC=org/DC=doegrids/OU=People/CN=David Adams 268704" usatlas1
"/DC=org/DC=doegrids/OU=People/CN=Davide Costanzo 130939" usatlas1
"/DC=org/DC=doegrids/OU=People/CN=Denis Oliveira Damazio 383391" usatlas1
...
```

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

gums-manualGroup-add [persistanceFactory] [manualGroupName] [UserDN]

This will add a user in a manually managed group. The first two parameters given must the ones used within the configuration file. For example, if in the configuration file you have:

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

Then the command will look like:

```
[root@atlasgrid13 sbin]# gums-manualGroup-add mysql myGroup
"/DC=org/DC=doegrids/OU=People/CN=Gabriele Carcassi"
```

gums-manualGroup-remove [persistanceFactory] [manualGroupName] [UserDN]

This will remove a user in a manually managed group. The first two parameters given must the ones used within the configuration file. For example, if in the configuration file you have:

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

Then the command will look like:

```
[root@atlasgrid13 sbin]# gums-manualGroup-remove mysql myGroup
"/DC=org/DC=doegrids/OU=People/CN=Gabriele Carcassi"
```

gums-manualMapping-add [persistanceFactory] [manualMappingName] [UserDN] [account]

This will add a map in a manually managed mapping. The first two parameters given must the ones used within the configuration file. For example, if in the configuration file you have:

```
...
      <accountMapping className='gov.bnl.gums.ManualAccountMapper2'
persistanceFactory='mysql' name='bnlMapping' />
...
```

Then the command will look like:

```
[root@atlasgrid13 sbin]# gums-manualMapping-add mysql myGroup
"/DC=org/DC=doegrids/OU=People/CN=Gabriele Carcassi" carcassi
```

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

gums-manualMapping-remove [persistanceFactory] [manualMappingName] [UserDN]

This will remove a map in a manually managed mapping. The first two parameters given must the ones used within the configuration file. For example, if in the configuration file you have:

```
...
      <accountMapping className='gov.bnl.gums.ManualAccountMapper2'
persistanceFactory='mysql' name='bnlMapping' />
...
```

Then the command will look like:

```
[root@atlasgrid13 sbin]# gums-manualMapping-remove mysql myGroup
"/DC=org/DC=doegrids/OU=People/CN=Gabriele Carcassi"
```

gums-mapfileCache-refresh

It regenerates the maps and stores them in the database for the clients to retrieve them.

gums-mapfileCache-refresh

Contacts the database, retrieves the grid-mapfile for the local host and displays it on the standard output.

gums-mapUser

```
[root@atlasgrid13 sbin]# /opt/gums/sbin/gums-updateGroups -h
usage: gums2-updateGroups
Updates the local databases containg the list of users in the groups.
Options:
 -h,--help   print this message
```

gums-updateGroups

```
usage: gums2-mapUser [-n hostname] userDN1 [userDN2] ...
Maps the grid identity to the local user.
Options:
 -h,--help    print this message
 -n,--name    hostname; localhost by default
```

Client side

gums-cron-generateGridMapfile

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

It's the cron script that is linked in /etc/cron.hourly. It will run once a day and will call gums-mapfileCache-retrieve to update the grid-mapfile.

gums-mapfileCache-retrieve

Contacts the database, retrieves the grid-mapfile for the local host and displays it on the standard output.

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

1.6 FAQ

Frequently Asked Questions

General

1. What is GUMS?
2. Is GUMS being used in production anywhere?

Using GUMS

1. Can I install GUMS at my site?

Building GUMS

General

General

What is GUMS?

GUMS is a Grid Identity Mapping Service that manages how Grid identities (i.e. certificates) are mapped to local identities. It's main two functionalities are creating grid-mapfiles and mapping a single user for a specific server. It's intended for a site that have multiple gatekeepers and you want to create a policy defining who has access to those resources and with what account. You can say things like: "on all the nodes that match *.exp1.site.org, the users defined in group 'admin' in the 'exp1' VO are going to be mapped to a group account 'adminExp1', while all the other users defined in the 'exp1' VO are to be mapped to their local account".

Is GUMS being used in production anywhere?

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

Using GUMS

Using GUMS

Can I install GUMS at my site?

In theory yes, in practice you can't just dowload an rpm... GUMS started as an internal system and we are now in the process of creating a product that can be used on different sites. Basically, we lack a packaged version with manual and installation instructions.

Building GUMS

Building GUMS

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

1.7 Changes

Release History

Version	Date	Description
0.7	in CVS HEAD	
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 0.7 - in CVS HEAD

Type	Changes	By
	Support for VOMS Fully Qualified Attribute names	carcassi
	GT3 AuthZ service	carcassi
	Grid3 test gatekeeper with account pooling enabled	carcassi
	Support for grid3-user-vo-map.txt generation	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

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

Type	Changes	By
	Better log system: logs for developer, resource admin and site admin in place	carcassi
	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

©2003 BNL SITEAAA • ALL RIGHTS RESERVED

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

©2003 BNL SITEAAA • ALL RIGHTS RESERVED