First production Grid layer

Project Number: FP6-2005-IST-026996

Deliverable id: D 4.3

Deliverable name: First production Grid layer

Submission Date: 03/03/2008

COVER AND CONTROL PAGE OF DOCUMENT	
Project Acronym:	ACGT
Project Full Name:	Advancing Clinico-Genomic Clinical Trials on Cancer: Open Grid Services for improving Medical Knowledge Discovery
Document id:	D 4.3
Document name:	First production Grid layer
Document type (PU, INT, RE)	INT
Version:	1
Submission date:	03/03/2008
Editor: Organisation: Email:	Juliusz Pukacki PSNC pukacki@man.poznan.pl

Document type PU = public, INT = internal, RE = restricted

ABSTRACT

This deliverable presents the efforts which have been put so far into preparing the first(lowest) layer of production Grid infrastructure. The architecture of ACGT Grid infrastructure has been described in document D4.1. This document summarizes and describes the technical details regarding implementation of the proposed Grid architecture, including definition of required software components and description of the installation and configuration procedures.

KEYWORD LIST:Grid, Grid Services, Globus Toolkit, Service Oriented Architecture, Grid Resource Management Systems, Data Management, Grid Monitoring

MODIFICATION CONTROL			
Version	Date	Status	Author
1.0	20/02/2008	Draft	B. Ludwiczak
2.0	03/03/2008	Draft	B. Ludwiczak
3.0	04/03/2008	Draft	J. Pukacki

List of Contributors

- Bogdan Ludwiczak, PSNC
- Jarek Nabrzyski, PSNC
- Tomasz Piontek, PSNC
- Juliusz Pukacki, PSNC
- Marcin Wolski, PSNC
- Marcin Adamski, PSNC

Contents

Executive Summary

ACGT is an Integrated Project (IP) funded in the 6th Framework Program of the European Commission under the Action Line “Integrated biomedical information for better health”. The high level objective of the Action Line is the development of methods and systems for improved medical knowledge discovery and understanding through integration of biomedical information (e.g. using modelling, visualization, data mining and grid technologies). Biomedical data and information to be considered include not only clinical information relating to tissues, organs or personal health-related information but also information at the level of molecules and cells, such as that acquired from genomics and proteomics research.

ACGT focuses on the domain of Cancer research, and its ultimate objective is the design, development and validation of an integrated Grid enabled technological platform in support of post-genomic, multi-centric Clinical Trials on Cancer. The driving motivation behind the project is our committed belief that the breadth and depth of information already available in the research community at large, present an enormous opportunity for improving our ability to reduce mortality from cancer, improve therapies and meet the demanding individualization of care needs.

1. Overview

1.1.The ACGT Grid

ACGT Grid architecture comprises several layers as shown on figure 1 (as described in deliverable D4.1). This document concentrates on 'Common Grid Infrastructure' and partially on 'Advanced Grid Middleware'. It provides information on required software components and their installation and configuration procedures. This requirements may evolve in the future accordingly to emerging project's needs and as new versions of grid software appears.

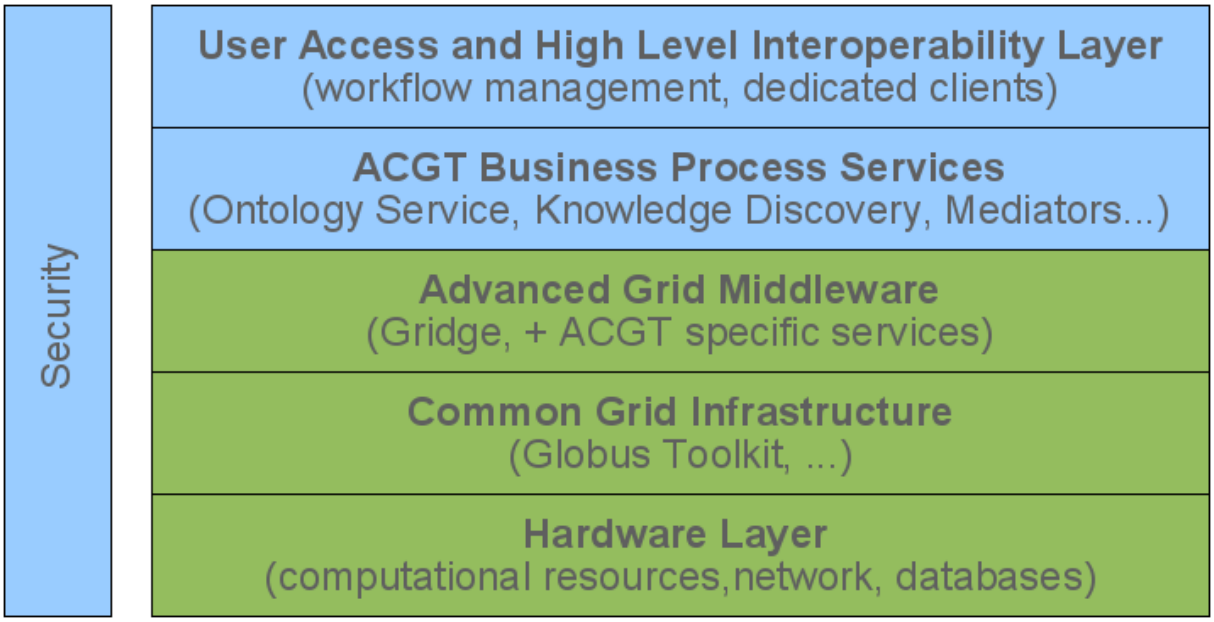

Fig.1. ACGT Architecture Overview

1.2.Common Grid layer requirements

For the moment the only required software in this layer is Globus Toolkit version 4.0.x. The various components of the latest version of Globus Toolkit (GT4) comprise ACGT common grid layer. Globus is not a monolithic software system but consists of some network services that operate together providing unified background for Grid environments. All participating sites should deploy at least the following Globus services:

- MDS4 – the Monitoring and Discovery System which is a suite of services to monitor and discover resources available on the Grid. It also allows to discover what resources are considered part of a Virtual Organization (VO) and to monitor those resources. This kind of information system is required to provide resource management features implemented services in Advanced Grid Middleware layer .
- WS-GRAM – an execution management service and tools which provide an interface to job initiation, management, scheduling, and coordination of remote computations.
- Globus pre-ws gatekeeper – older version Globus GRAM, required for the compatibility reasons, this requirement may be removed in the future.
- GridFTP – Grid File Transfer Protocol provides means for the secure, robust, fast and efficient transfer of data.

- RFT – RFT (Reliable File Transfer) is a Web Services Resource Framework (WSRF) compliant web service that provides "job scheduler"-like functionality for data movement. User can simply provide a list of source and destination URLs and then the service writes the job description into a database and then moves the files on user's behalf.

Every site participating in ACGT Grid testbed is also required to:

- require appropriate host certificates
- register their MDS4 Index Service in central VO Index Service provided by PSNC. That provides a means to monitor and track the state of the whole ACGT Grid infrastructure from one point.
- accept ACGT Certificate Authority certificates.

1.3. Advanced Grid Middleware

Advanced Middleware Layer is responsible for providing more advanced mechanisms in the Grid environment. Services from this layer can be described as "collective" because they operate on set of lower level services, to realize more advanced actions - e.g. metascheduling service that submits jobs to different local queuing systems using Common Grid Infrastructure remote interfaces.

Functionality provided by this layer can be gathered in a following main points:

- resource management - metascheduling
- data management (database access and file storing and transferring)
- services authorization

2. ACGT Grid Infrastructure Security

Security concepts used in ACGT Grid infrastructure are based on Globus Security Infrastructure (GSI) model. GSI uses public key cryptography (or “asymmetric cryptography”) as the basis for its functionality. The primary motivations behind GSI are([5]):

- The need for secure communication (authenticated and perhaps confidential) between elements of a computational Grid.
- The need to support security across organizational boundaries, thus prohibiting a centrally-managed security system.
- The need to support "single sign-on" for users of the Grid, including delegation of credentials for computations that involve multiple resources and/or sites.

The most important concepts of GSI are:

- GSI is Public Key Cryptography

  Public key cryptography relies on two keys. These keys are numbers that are mathematically related in such a way that if either key is used to encrypt a message, the other key must be used to decrypt it. Regarding the current knowledge of mathematics and available computing power, it is next to impossible to obtain the second key from the first one and/or any messages encoded with the first key.

  By making one of the keys available publicly (a public key) and keeping the other key private (a private key), a person can prove that he or she holds the private key simply by encrypting a message. If the message can be decrypted using the public key, the person must have used the private key to encrypt the message.

  Important note: It is critical that private keys be kept private! Anyone who knows the private key can easily impersonate the owner.

- Digital Signatures

  Using public key cryptography, it is possible to digitally “sign” a piece of information. Signing information essentially means assuring a recipient of the information that the information has not been tampered with since it left sender's hands.

  To sign a piece of information, first compute a mathematical hash of the information. (A hash is a condensed version of the information. The algorithm used to compute this hash must be known to the recipient of the information, but it isn't a secret.) Using your private key, encrypt the hash, and attach it to the message. Make sure that the recipient has your public key.

  To verify that your signed message is authentic, the recipient of the message will compute the hash of the message using the same hashing algorithm you used, and will then decrypt the encrypted hash that you attached to the message. If the newly-computed hash and the decrypted hash match, then it proves that you signed the message and that the message has not been changed since you signed it.

- Certificates

  A central concept in GSI authentication is the certificate. Every user and service on the Grid is identified via a certificate, which contains information vital to identifying and authenticating the user or service.

  A GSI certificate includes four primary pieces of information:

  - A subject name, which identifies the person or object that the certificate represents
  - The public key belonging to the subject.

- The identity of a Certificate Authority (CA) that has signed the certificate to certify that the public key and the identity both belong to the subject.
- The digital signature of the named CA.

Third party (a CA) is used to certify the link between the public key and the subject in the certificate. In order to trust the certificate and its contents, the CA's certificate must be trusted. The link between the CA and its certificate must be established via some non-cryptographic means, or else the system is not trustworthy.

GSI certificates are encoded in the X.509 certificate format, a standard data format for certificates established by the Internet Engineering Task Force (IETF). These certificates can be shared with other public key-based software, including commercial web browsers from Microsoft and Netscape.

- Mutual Authentication

  If two parties have certificates, and if both parties trust the CAs that signed each other's certificates, then the two parties can prove to each other that they are who they say they are. This is known as mutual authentication. GSI uses the Secure Sockets Layer (SSL) for its mutual authentication protocol (described in [5]).

  Before mutual authentication can occur, the parties involved must first trust the CAs that signed each other's certificates. In practice, this means that they must have copies of the CAs' certificates which contain the CAs' public keys and that they must trust that these certificates really belong to the CAs.

- Confidential Communication

  By default, GSI does not establish confidential (encrypted) communication between parties. Once mutual authentication is performed, GSI gets out of the way so that communication can occur without the overhead of constant encryption and decryption.

  GSI can easily be used to establish a shared key for encryption if confidential communication is desired. Recently relaxed United States export laws now allow us to include encrypted communication as a standard optional feature of GSI.

  A related security feature is communication integrity. Integrity means that an eavesdropper may be able to read communication between two parties but is not able to modify the communication in any way. GSI provides communication integrity by default. (It can be turned off if desired). Communication integrity introduces some overhead in communication, but not as large an overhead as encryption.

- Securing Private Keys

  The core GSI software provided by the Globus Toolkit expects the user's private key to be stored in a file in the local computer's storage. To prevent other users of the computer from stealing the private key, the file that contains the key is encrypted via a password (also known as a passphrase). To use GSI, the user must enter the passphrase required to decrypt the file containing their private key.

- Delegation, Single Sign-On and Proxy Certificates

  GSI provides a delegation capability: an extension of the standard SSL protocol which reduces the number of times the user must enter his passphrase. If a Grid computation requires that several Grid resources be used (each requiring mutual authentication), or if there is a need to have agents (local or remote) requesting services on behalf of a user, the need to re-enter the user's passphrase can be avoided by creating a proxy.

  A proxy consists of a new certificate and a private key. The key pair that is used for the proxy, i.e. the public key embedded in the certificate and the private key, may either be regenerated for each proxy or obtained by other means. The new certificate contains the owner's identity, modified slightly to indicate that it is a proxy. The new certificate is

signed by the owner, rather than a CA. The certificate also includes a time notation after which the proxy should no longer be accepted by others. Proxies have limited lifetimes.

The new certificate is signed by the owner, rather than a CA.

The proxy's private key must be kept secure, but because the proxy isn't valid for very long, it doesn't have to kept quite as secure as the owner's private key. It is thus possible to store the proxy's private key in a local storage system without being encrypted, as long as the permissions on the file prevent anyone else from looking at them easily. Once a proxy is created and stored, the user can use the proxy certificate and private key for mutual authentication without entering a password.

When proxies are used, the mutual authentication process differs slightly. The remote party receives not only the proxy's certificate (signed by the owner), but also the owner's certificate. During mutual authentication, the owner's public key (obtained from her certificate) is used to validate the signature on the proxy certificate. The CA's public key is then used to validate the signature on the owner's certificate. This establishes a chain of trust from the CA to the proxy through the owner.

3. Globus Toolkit installation.

This sections provides overview of Globus Toolkit ver. 4.0.3 installation procedure.

Globus Toolkit is provided both in binary and source versions. It is also available as Java WS-core only or full version. For the purpose of the ACGT Grid Infrastructure the full source installation is required, as it provides the full set of Globus Toolkit features and in most cases is more reliable (though more troublesome in rare cases) and robust then binary version.

Below are the steps required to compile and install Globus Toolkit. Note: this is only an outline of the installation process which should be sufficient to prepare ACGT Grid compatible computational resource, full and detailed information regarding toolkit installation can be found in official Globus documentation ([4]).

- Check if target system has all require software components:
  - Java 1.4.2+ SDK (preferably SUN, do not use GCJ)
  - Ant 1.6+
  - gcc (latest 3.4 or 4.1)
  - GNU tar, sed, make
  - sudo
  - zlib
  - JDBC compliant database – e.g. PostgreSQL 7.1+
  - tomcat

  Note: sudo and database are not required during compilation and installation of the toolkit, but they are essential during configuration and regular usage.
- Create a dedicated, non-privileged user named “globus”. This user
  - owns the installation/deployment files
  - compile and install Globus Toolkit
  - performs administrative tasks such as starting and stopping the container, deploying services, etc.
- Create deployment directory

  - make sure that user "globus" has read and write permissions in the installation directory

- NOTE: Following steps should be carried out by the "globus" user
- Download and untar Globus Toolkit ver.4.0.3 all-source-installer (available at the web site: http://www.globus.org/toolkit/downloads/4.0.3/#source)
- `cd` to source directory (i.e. `~/gt4.0.3-all-source-installer`)
- set environmental variable `GLOBUS_LOCATION` to point to installation directory (e.g. "`/opt/globus`")
- make sure that `JAVA_HOME` variable points to the top-level directory of your Java installation before using `./configure` script
- finally use `~/gt4.0.3-all-source-installer/configure` to set up compilation and installation option. Use at least –prefix option to set the deployment directory. By default, only simple "fork" job execution manager will be build and installed. Depending on queueing system available on the resource, other job managers can by enabled by appropriate command line argument:
  - `--enable-wsgram-condor` for CONDOR,
  - `--enable-wsgram-lsf` for LSF
  - `--enable-wsgram-pbs` for PBS or TORQUE.

  Note: before running `./configure` script, make sure that queuing system's client commands (like `qsub` or `qstat`) are on the standard search path

  The minimum required configuration command is:

  ```
  ./configure --prefix=${GLOBUS_LOCATION}
  ```

- Compile and install Globus Toolkit by running:

  ```
  make
  make install
  ```

  Note:
  - GT4 unlike other software packages is made directly to target location (`$GLOBUS_LOCATION`)
  - you can monitor the build progress by checking the size of the `$GLOBUS_LOCATION` (e.g.: `` `du –sk $GLOBUS_LOCATION` ``). Full GT4.0.3 install takes ca. 240MB
  - "`make`" phase may take several hour to complete.
  - during "`make install`" phase Globus configuration will be initialized
- The above steps completes Globus Toolkit installation. Figure 2. summarizes shell commands used during typical installation process.

bogdanl@cress:~ -Shell konsole <1>

```
bogdanl@cress ~ $ su -
cress ~ # useradd -m globus
cress ~ # mkdir /opt/globus
cress ~ # chown globus:users /opt/globus
cress ~ # su - globus
globus@cress ~ $ wget http://www-
unix.globus.org/ftppub/gt4/4.0/4.0.3/installers/src/gt4.0.3-all-source-installer.tar.bz2
globus@cress ~ $ tar xjf gt4.0.3-all-source-installer.tar.bz2
globus@cress ~ $ cd gt4.0.3-all-source-installer
globus@acgt ~ $ cd gt4.0.3-all-source-installer
globus@acgt ~/gt4.0.3-all-source-installer $ ./configure \
--prefix=${GLOBUS_LOCATION} \
--with-prewsmds
globus@acgt ~/gt4.0.3-all-source-installer $ make
```

which will give on stdout:

```
build_gpt ====> installing GPT into /opt/globus
build_gpt ====> building /home/globus/gt4.0.3-all-source-installer/gpt/support/Compress-
Zlib-1.21
...
```

...and a series of:

```
/opt/globus/sbin/gpt-build   -srcdir=source-trees-thr/core/source gcc32dbgpthr
gpt-build ====> Changing to /home/globus/gt4.0.3-all-source-
installer/BUILD/globus_core-4.30/
gpt-build ====> BUILDING FLAVOR gcc32dbgpthr
gpt-build ====> Changing to /home/globus/gt4.0.3-all-source-installer/BUILD
...
```

what cat take more then 2 hours and in the end it says:

```
Your build completed successfully.  Please run make install.

globus@acgt ~/gt4.0.3-all-source-installer $ make install
/opt/globus/sbin/gpt-postinstall
running /opt/globus/setup/globus/setup-globus-common..[ Changing to
/opt/globus/setup/globus ]
creating globus-sh-tools-vars.sh
creating globus-script-initializer
creating Globus::Core::Paths
checking globus-hostname

...

..Done
globus@acgt ~/gt4.0.3-all-source-installer $
```

Fig.2. Typical Globus Toolkit installation.

4. Globus Toolkit basic configuration

4.1.Security

After Globus Toolkit installation, but before staring Globus services several aspects of GSI security need to be configured:

- configure Globus to trust a particular set of CAs (Certificate Authorities), i.e. place certificates of trusted CAs into designated directory – CA is trusted only if its CA certificate exists with the appropriate name in an appropriate directory. Moreover, for pre-ws services, signing policy file must exist in the same location as CA certificate. In other words, one needs two files to trust given CA: `cert_hash.0` – the trusted CA certificate and `cert_hash.signing_policy` – the signing policy. Java-based components ignore it and accept all valid certificates issued by trusted CAs. Globus services and tools looks for that directory in following locations:
  - the value of `$X509_CERT_DIR` environment variable if it is set and the directory exists,
  - otherwise, in `$HOME/.globus/certificates` if it exists,
  - otherwise, in `/etc/grid-security/certificates` if it exists,
  - otherwise, in `$GLOBUS_LOCATION/share/certificates` if it exists.

  Note: We suggest to use `/etc/grid-security/certificates` as system wide trusted CAs directory, but remember that `$X509_CERT_DIR` and `$HOME/.globus/certificates` have higher priority.

  `cert_hash.0`, i.e. certificate of the CA, is provided by CA, usually with appropriate hash name. Hash name consists of 8 hex-digits and suffix "`.0`".(e. g. `8a661490.0`). Valid has can be obtained by following command (available in `$GLOBUS_LOCATION/bin/`):

  ```
  openssl x509 -hash -noout -in ca_certificate
  ```

- `cert_hash.signing_policy` usually is also provided by CA, but it can constructed manually. The signing policy file has the following format:

  ```
  access_id_CA X509 'CA Distinguished Name'
  pos_rights globus CA:sign
  cond_subjects globus '"Name Pattern1" "Name Pattern2" ...'
  ```

  - to get 'CA Distinguished Name' execute:

    ```
    openssl x509 -subject -noout -in cert_hash.0
    ```

  - "name pattern" is a string used to match the distinguished names of certificates granted by the given CA. Usually, it is a CA name with common name replaced by wild card '*', e.g.:

    `"/C=PL/O=GRID/CN=Polish Grid CA" -> '"/C=PL/O=GRID/*"'`

    it accepts "/C=PL/O=GRID/OU=PSNC/CN=Bogdan Ludwiczak"

    it rejects "/O=GRID/OU=PSNC/CN=Bogdan Ludwiczak"

    `'"*"'` pattern accepts all certificates.

- Configure appropriate default values for use by the grid-cert-request command which is used to generate certificates requests. The following files have to be properly configured to enable Globus tools to generate valid certificate requests:
  - `/etc/grid-security/globus-user-ssl.conf` – defines the distinguished name to use for a user's certificate request.
  - `/etc/grid-security/globus-host-ssl.conf` – defines the distinguished name for a host and service certificate request.
  - `/etc/grid-security/grid-security.conf` – is a main configuration file that contains the name and email address for the given CA.

  These files are usually provided by the CA, particularly ACGT CA does provides these files. Typically, CA configuration files are placed in `/etc/grid-security/certificates/` directory with additional extension "`.CA_hash_name`" and only appropriate symbolic links are created in `/etc/grid-security/`. Globus Toolkit provides `grid-default-ca` command which can be used to automatically create appropriate links.
- acquire host X.509 certificates from your CA

  Use Globus command `grid-cert-request` to generate host and users certificate request. Make sure that CA configuration files and certificate are in place before generating requests. This command will create 3 files:
  - an empty (file length is 0) `/etc/grid-security/hostcert.pem` or `~/usercert.pem`,
  - `/etc/grid-security/hostkey.pem` or `~/userkey.pem` file containig host/user private key which must be kept secret – make sure that the unix access mode is set to `0400` or `0600` at most,
  - `/etc/grid-security/hostcert_request.pem` or `usercert_request.pem` file containing acctula request to be signed by CA.

  Send newly generated certificates request to the appropriate CA and wait for the certificate which should be send in return by your CA. Save the new certificate in `hostcert.pem` or `usercert.pem`. Now, request file can be deleted. Host certificate and private key should be owned by "root" user.
- specify identity mapping information

  Globus services map distinguished names (retrieved from certificates) to local identities (unix account) by means of grid-mapfile. Mappings have the folowing form:

  "Distinguished Name" local_name

  (every file of the file defines one mapping)

  To let user in, create an account and add mapping to grid-mapfile.

  Globus looks for the file in the following locations:
  - the value of `GRIDMAP` environment variable if it is set,
  - otherwise, if service is run as root then grid map file is `/etc/grid-security/grid-mapfile`,
  - otherwise, the grid map file is `$HOME/.gridmap`
  - otherwise, in `/etc/grid-security/grid-mapfile`.

Accepting ACGT CA and installation of related CA files.

ACGT CA certificate and and configuration files are provided in a form of a tar ball available at: http://moss1.man.poznan.pl/acgt-ca.tar. The hash name of ACGT CA is `e622f687`. To accept ACGT CA and install its configuration files simply:

- download the tar ball
- and as "root" user, untar it into `/etc/grid-security` directory.

This should place all files into `/etc/grid-security/certificates/` and create required links in `/etc/grid-security/` directory.

4.2.Firewall configuration

Detailed information on GT firewall issues can be found inn [6]. This paragraph only lists the minimum required firewall configuration and gives a short overview of the basic issues.

To enable remote access to Globus Toolkit services the following TCP port should be opened for the incoming connections:

- static ports:
  - 2119 for pre-ws GRAM
  - 2811 for GridFTP
  - 8080,8443 for Globus service container
  - arbitrary chosen port for gsi-enabled ssh, we suggest to leave regular ssh on standard 22/TCP port
- Ephemeral TCP ports
  - Various Globus services require an arbitrary chosen TCP port range. It is controllable by environmental variable `GLOBUS_TCP_PORT_RANGE` for pre-ws components or by jav system property `"org.globus.tcp.port.range"` for java-based ws components. It can be set as follows:

```
$ java -Dorg.globus.tcp.port.range=5000,6000
$ export GLOBUS_TCP_PORT_RANGE=5000,6000
```

  - make sure it is defined in daemons environment (i.e. put it in (x)inetd configuration entries and in `GLOBUS_OPTIONS` env variable for Globus service container)
  - mind that number of of ports limits the number of controllable jobs which can be submitted via pre-ws GRAM

Figures 3. and 4. gives an overview of typical Globus communication flow for pre-ws GRAM and Grid-FTP appropriately. They show the meaning of the aforementioned Globus TCP port range.

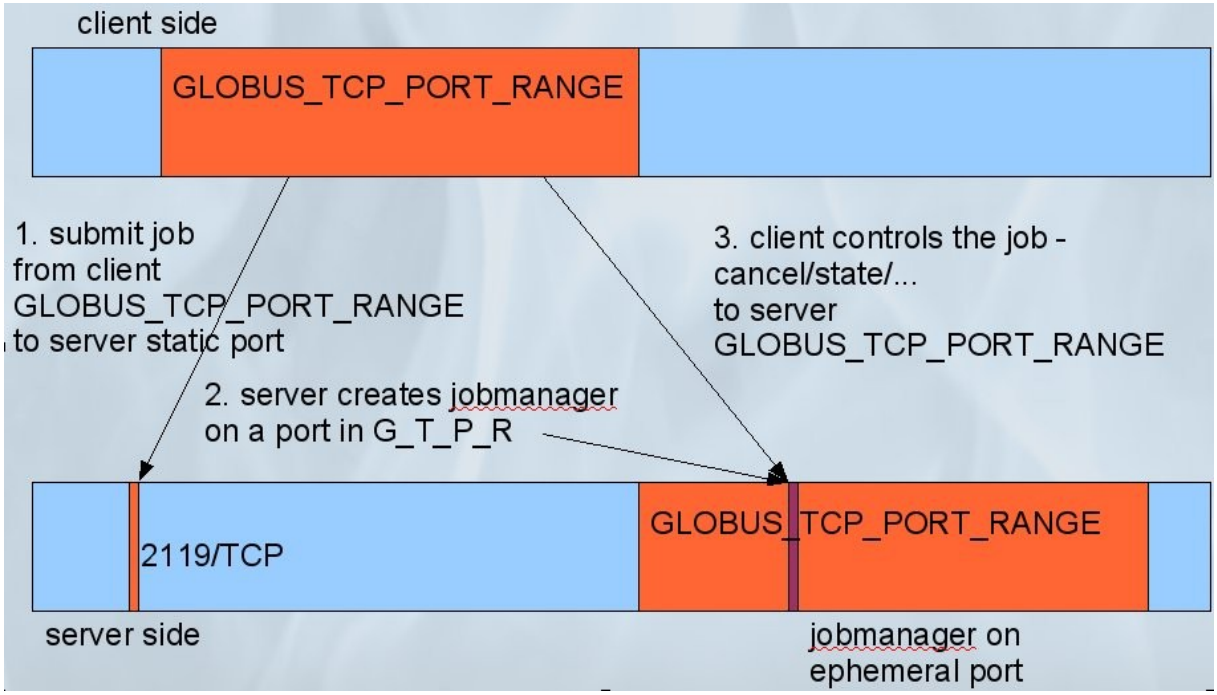

Fig.3. Pre-ws GRAM communication diagram

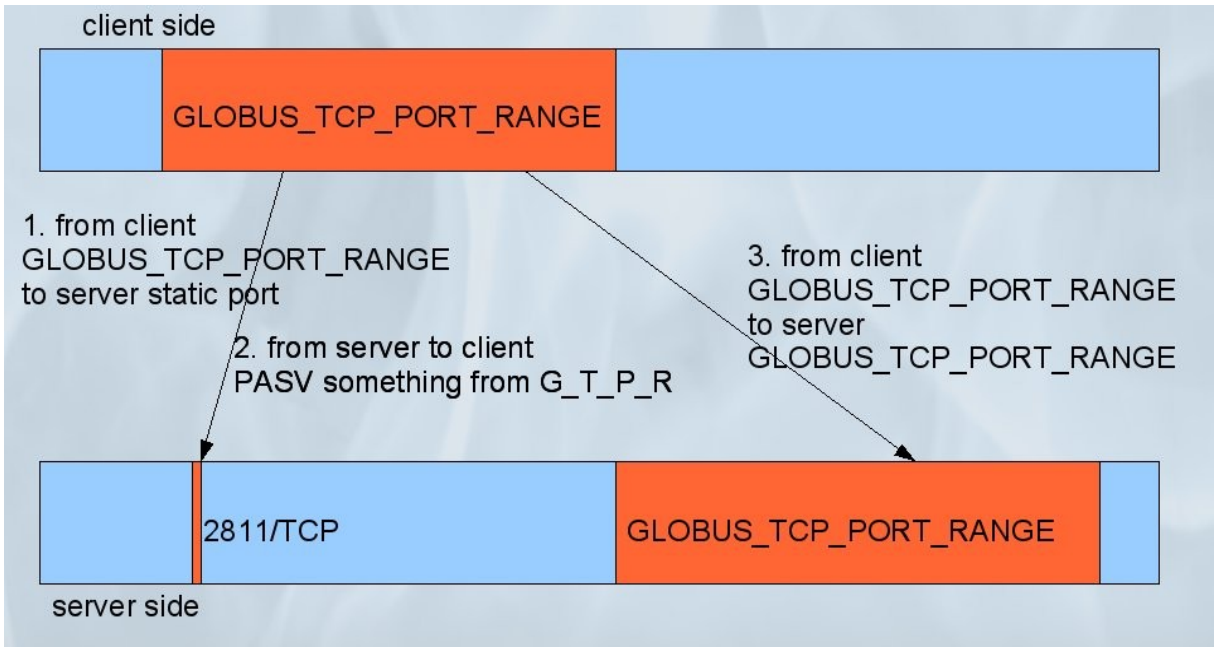

Fig.4. GridFTP communication diagram.

4.3.Globus Toolkit configuration – GridFTP

After successful installation of GT, the GridFTP daemon binary is loacated in:

`$GLOBUS_LOCATION/sbin/globus-gridftp-server` file. The daemon could be run in standalone mode, but we suggest using (x)inetd mode. First add the following entry to `/etc/services` file:

```
gsiftp          2811/tcp                # GridFTP
```

(Port 2811 is the IANA registered GridFTP port)

Next add appropriate entry to (x)inetd configuration file(s). Remember to set LD_LIBRARY_PATH, GLOBUS_LOCATION and GLOBUS_TCP_PORT_RANGE in the (x)inetd entry environment, like in the xinetd example below:

```
service gsiftp
{
        instances               = 100
        socket_type             = stream
        wait                    = no
        user                    = root
        env                     = LD_LIBRARY_PATH=/opt/globus/lib
        env                     += GLOBUS_LOCATION=/opt/globus
        env                     += GLOBUS_TCP_PORT_RANGE=9000,9999
        server                  = /opt/globus/sbin/globus-gridftp-server
        server_args             =  -i -c /opt/globus/etc/gridftp.conf
        log_on_success          += DURATION
        log_on_failure          +=
        nice                    = 10
        disable                 = no
}
```

Note that the above example specifies grid-ftp configuration file (`-c .../gridftp.conf` in `server_args` line). It is not required – GridFTP daemon should work without this file. A useful example of this file is:

```
log_level ALL (or 'EERROR', 'WARN', 'INFO', 'DUMP')
log_module stdio
log_single /var/log/gridftp
```

it enables logging which can be helpful by resolving potential problems.

Finally, restart (x)inetd to activate the GridFTP daemon.

4.4. Globus Toolkit configuration – WS-GRAM

Since Globus container acts on non-privileged account, it need a mean to execute jobs on the submitters behalf. The current GT version uses "`sudo`" command for that purpose. WS GRAM requires that the "`sudo`" command is installed and functioning on the service host. Appropriate authorization rules will need to be added to the `sudoers` file to allow the WS GRAM service account to execute (without a password) the scheduler adapter in the accounts of authorized GRAM users (job submitters). Below are the required `sudoers` rules:

```
globus     ALL=(ALL) NOPASSWD: \
     /opt/globus/libexec/globus-gridmap-and-execute \
     -g    /etc/grid-security/grid-mapfile \
     /opt/globus/libexec/globus-job-manager-script.pl *
```

```
globus     ALL=(ALL)  NOPASSWD:
      /opt/globus/libexec/globus-gridmap-and-execute \
      -g /etc/grid-security/grid-mapfile \
      /opt/globus/libexec/globus-gram-local-proxy-tool *
```

4.5.Globus Toolkit configuration – RFT

RFT requires JDBC compliant database system to operate. The prefered database system is PostgreSQl. To enable RFT:

- install PostgreSQL in a way suitable for the resource's operating system.
- Configure the postmaster daemon so that it accepts TCP connections. This can be done by adding the "`-o -i`" switch to the postmaster script.
- create "`globus`" PostgresSQL user:

```
$ su postgres
$ createuser globus
```

- create "`rftDatabase`" database owned by "`globus`"

```
$ su postgres
$ createdb -O globus rftDatabase
```

- set the security on the "`rftDatabase`" – i.e. add the following line to `pg_hba.conf`:

```
host rftDatabase globus "host-ip" 255.255.255.255 md5
```

- and restart/reload PostgreSQL
- initialize database structure:

```
psql -U globus -d rftDatabase \
   -f $GLOBUS_LOCATION/share/globus_wsrf_rft/rft_schema.sql
```

- Check database parameters in RFT configuration file `$GLOBUS_LOCATION/etc/globus_wsrf_rft/jndi-config.xml`
- Find the <resource name="dbConfiguration" element and change the `connectionString` parameter to point to the machine on which you installed PostgreSQL and to the name of the RFT database,
- Change the `userName` and `password` to the name of the user who owns the RFT database
- default values for other parameters should work well in most cases.

4.6.Globus Toolkit configuration – MDS4

As described in [3] WS MDS4 consist of:

- Index Service – collects monitoring and discovery information from Grid resources, and publishes it in a single location. It aggregates the WS-Resources properties
- Trigger Service – collects data from resources on the grid and, if administrator defined rules match, can perform various actions.
- Aggregator Framework – the software framework on which Index and Trigger are built.

- WebMDS – an additional, helper component. It is a web browser interface for viewing monitoring information (not required)

This document covers only minimum Index Service configuration which is required for proper operation of ACGT common grid infrastructure layer. More information on configuring MDS4 is available in [4].

By default every GT4 container provides container-wide MDS4 index service. Local container's services (by default WS-GRAM and RFT) automatically registers to container index when correctly configured

MDS4 Index Services can register to each other comprising multi level hierarchy. Particularly, local indexes can be registered to VO wide index. For the ACGT current Grid infrastructure the VO-Index service is provided and maintained by PSNC. It is available at:

https://moss1.man.poznan.pl:8443/wsrf/services/DefaultIndexService

PSNC also provides WebMDS interface for the ACGT VO Index. It is available at:

http://moss1.man.poznan.pl/webmds/

Figure 5. shows

Figure 6. presents sample view of the ACGT WebMDS interface. This view list services available in ACGT Common Grid Layer for the moment of taking the screen shot.

4.6.1. ACGT required MDS4 Configuration

- turn on host name publishing (instead of default IP publishing) – add two parameters to `<globalConfiguration>` section in

  `${GLOBUS_LOCATION}/etc/globus_wsrf_core/server-config.wsdd`

  file:

```
<parameter name="publishHostName"
           value="true"/>
<parameter name="domainName"
           value="man.poznan.pl"/>
```

  this will change information available in index:

  from: `https://150.254.173.173:8443/wsrf/services/ManagedJobFactoryService`

  to: `https://moss1.man.poznan.pl:8443/wsrf/services/ManagedJobFactoryService`

- Building MDS4 hierarchy

  MDS4 supports both, upstream and downstream registration. Its configured in

  `${GLOBUS_LOCATION}/globus_wsrf_mds_index/hierarchy.xml`

  by any combination of any number of the following entries:

```
<upstream>
https://vo-host:8443/wsrf/services/DefaultIndexService
</upstream>
<downstream>
https//member-host:8443/wsrf/services/DefaultIndexService
</downstream>
```

to register in ACGT VO-level index add the following entry:

```
<upstream>
https://moss1.man.poznan.pl:8443/wsrf/services/DefaultIndexService
</upstream>
```

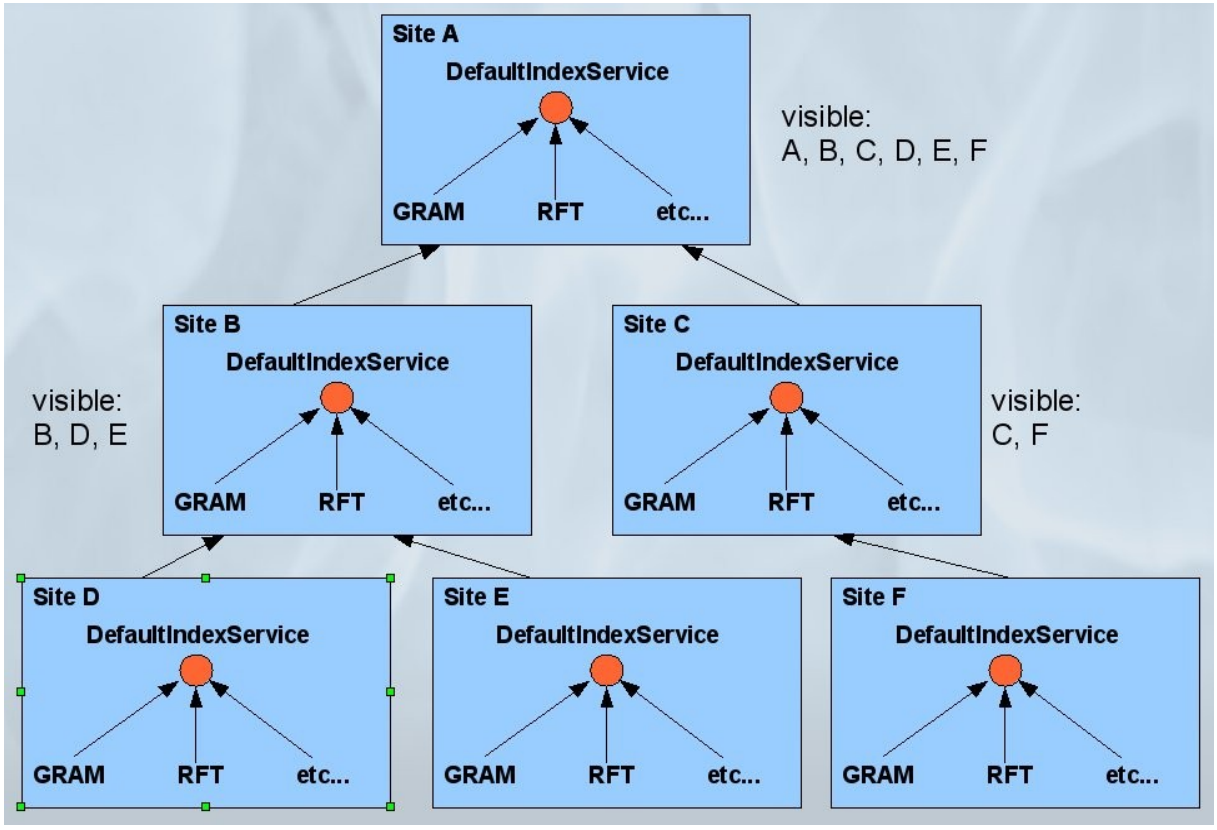

Fig.5. MDS4 hierarchy.

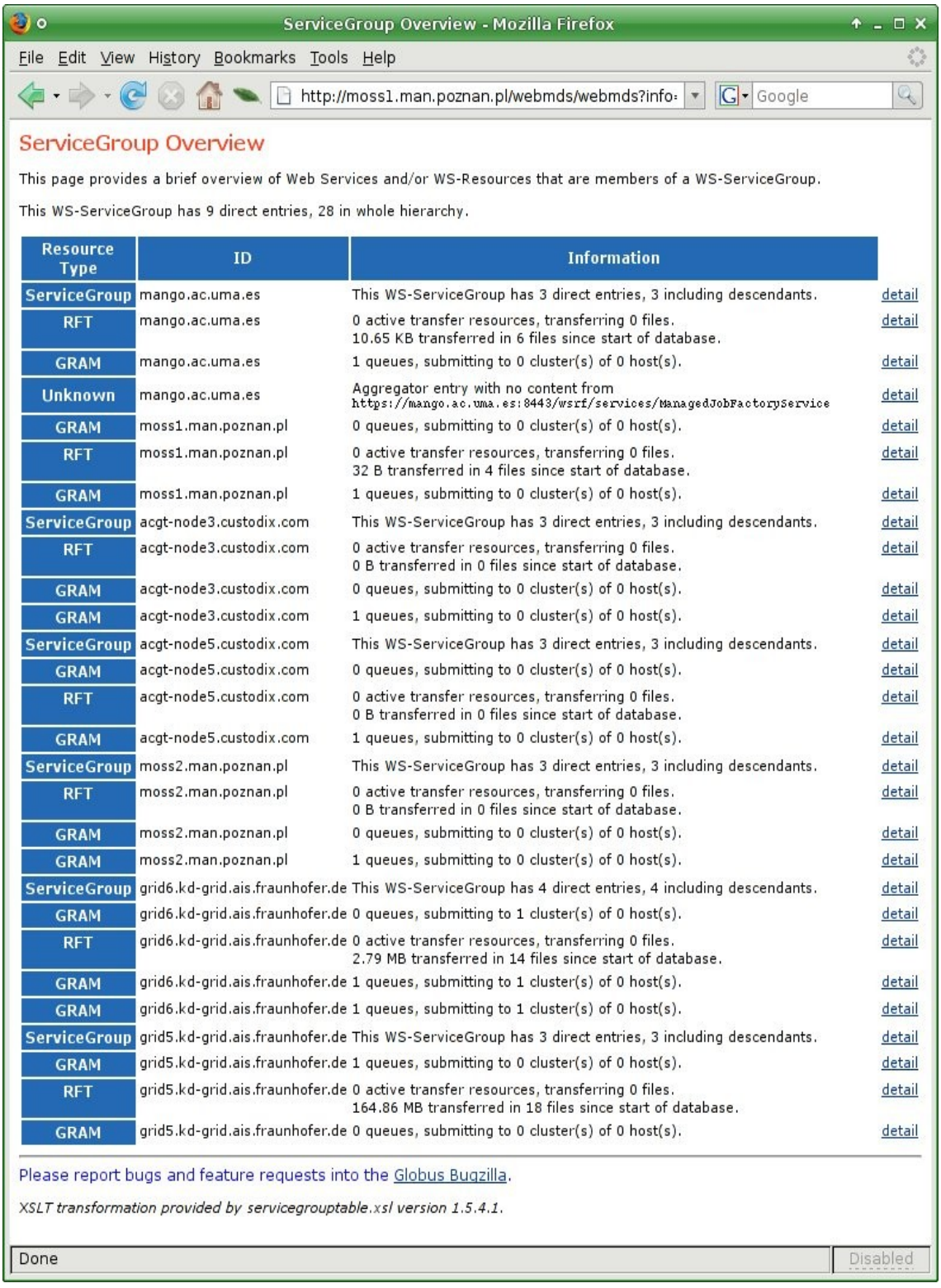

Fig.6. ACGT WebMDS view.

4.7.Globus Toolkit – starting service container

Globus Toolkit service container hosts various WSRF services including WS-GRAM, RFT and MDS4. So, it is essential component enabling most important services of ACGT common

grid infrastructure layer. Before staring the container one has to prepare container certificate and private key. In most cases this would be copies of host certificate and key. Simply:

- copy host certificate and key to apropriate location:

```
cp /etc/grid-security/hostcert.pem \
   /etc/grid-security/containercert.pem
cp /etc/grid-security/hostkey.pem \
   /etc/grid-security/containerkey.pem
```

- and change the ownership to the container ("`globus`") user.

Globus container is started/stoped by the following commands (syntax for bash shell):

```
$ su – globus
$ export GLOBUS_LOCATIO=/opt/globus/gt403
$ . $GLOBUS_LOCATION/etc/globus-user-env.sh
$ globus-start-container-detached
$ globus-stop-container-detached
```

4.8.Globus Toolkit – testing basic services

- Before testing Globus instalation by means of following commands one must obtain user certificates and then create user proxy by running command:

  ```
  grid-proxy-init
  ```

  also note that all these commands (including grid-proxy-init) require appropriately configured environment. Globus Toolkit provides scripts for that purpose. To configure users environment for GT command do the following for bash shell:

  ```
  export GLOBUS_LOCATION=/opt/globus/gt403
  . $GLOBUS_LOCATION/etc/globus-user-env.sh
  ```

  or for csh:

  ```
  setenv GLOBUS_LOCATION /opt/globus/gt403
  source $GLOBUS_LOCATION/etc/globus-user-env.sh
  ```

- submit simple jobs via WS-GRAM:

  ```
  globusrun-ws -submit -c /bin/touch touched_it
  globusrun-ws -submit -F https://acgt-xy:
  8443/wsrf/services/ManagedJobFactoryService -c /bin/sleep 123
  ```

  and check the results (by unix `ps` or `ls` command)

- query Default Index Service

  ```
  wsrf-query -s \
     https://moss1:8443/wsrf/services/DefaultIndexService
  ```

- copy some files:

  ```
  globus-url-copy gsiftp://moss1/~/file file:///tmp/testfile
  ```

4. Advanced Grid Middleware layer.

4.1. Gidge Resource Management System (GRMS)

4.1.1. Overview.

GRMS is composed from two main parts: GRMS Service - GSI enabled web service that provides interface for GRMS clients, and Broker Engine - module that provides main functionality of GRMS.

GRMS Service is a web service written in Java and running on a Java based hosting environment. The hosting environment consists of Tomcat, which is the servlet container and Apache Axis, which is a SOAP ("Simple Object Access Protocol") engine. The communication between the service and clients is done either by HTTPS protocol with support for proxy certificates (RFC 3280), or by GSI-enabled HTTP-based protocol called "httpg" implementing transport-level security introduced by Globus Community. To provide the transport-level security the part of GT4-core was used.

Broker is an engine of GRMS. It is implemented in Java, and can be accessed through RMI interface.

4.1.2. Reqiurements

• GRMS installer requires installed Perl interpreter and some modules. For full functionality two additional Perl modules should be installed in system:XML::LibXML and IO::Pty If first of these two modules won't be accessible on target machine, GRMS installer will skip configuration step. Also GRMS configuration/runtime control tools won't be accessible. In case of lack of IO::Pty module, during database setup, GRMS installer script will ask several times for user account password to PostgreSQL database.

• JDK 1.4.2 from: http://java.sun.com/j2se/1.4.2 [http://java.sun.com/j2se/1.4.2]

• MyProxy server from: http://www.ncsa.uiuc.edu/Divisions/ACES/MyProxy/

• One of RDBMS:

- PostgreSQL (the recommended version is 7.3.2) from: www.postgres.org
- MySQL from: www.mysql.org

4.1.3. Installing GRMS with script installer

1) Download GRMS installer from: http://gridge.man.poznan.pl/grms/download

2) Full GRMS installer syntax can be obtained by running installer without any parameter:

Usage:

```
grms-install.pl [--quiet] [--nodbsetup] --dest {destination_path}
  options:
     --quiet           - do not show detail information during
                                                    installation
     --nodbsetup       - do not create GRMS databases
     --dest {destination_path} - path to directory where GRMS will
                                                      be place
```

3) To begin installation destination path must be passed to installation script after --dest switch. For example:

```
# ./grms-install.pl --dest /usr/local/grms
```

4) After unpacking GRMS archive, installation script will ask for general settings. Default parameters are in square brackets, so if there is nothing put there, default value is used by installer:

a) *Enter local host name [cardaria.man.poznan.pl]:*

Default host name is taken from hostname system command. In 99% cases, default value is valid.

b) *Enter broker rmi port [60020]:*

RMI port on which Broker Engine will communicate with Web Service interface. If no other GRMS instance is running on host, this value should be proper. If multiple GRMS instances will run on singlehost, every one of them should operate on different RMI port.

c) *Enter webservice shutdown port [8005]:*

The port to which grms-control script sends a message when GRMS Web Service interface should be stopped. If more than one instance of GRMS is installed on single host, all of them should have different shoutdown port.

d) *Enter interface http port [7743]:*

HTTP protocol port, on which Web Service interface will be listening for anonymous requests. Multiple Web Service interfaces on single host should listen on different ports.

e) *Enter interface https port [7744]:*

HTTPS (HTTP over SSL) protocol port, on which Web Service interface will be listening for requests. Multiple Web Service interfaces on single host should listen on different ports.

f) *Enter interface httpg port [7745]:*

HTTPG (HTTP over GSI) protocol port, on which Web Service interface will be listening for requests. Multiple Web Service interfaces on single host should listen on different ports.

g) *Enter myproxy host name [cardaria.man.poznan.pl]:*

Hostname of MyProxy service, which Broker Engine and Web Service interface will be using.

h) *Enter myproxy port [7512]:*

Port on which MyProxy service is listening for requests.

i) *Enter database system [postgresql/mysql] [postgresql]:*

Determines RDBMS GRMS should use.

In further installation procedure description, assumption has be made, that postgresql option has been chosen, but for mysql RDBMS, all questions and their meaning remain the same.

j) *Enter database host [localhost]:*

Hostname of PostgreSQL/MySQL database engine. GRMS installer assumes, that PostgreSQL/MySQL service is running and properly configured, and that account which GRMS will use is createdand configured. In case when RDBMS is running on nonstandard port, hostname should contain character ':' followed by port on which database is running.

k) *Enter database user [grms]:*

PostgreSQL/MySQL account name which will be used by GRMS. GRMS installer assumes that this account is created and configured.

l) *Enter database user password [grms]:*

PostgreSQL/MySQL account password.

m) *Enter jobregistry database name [grms_jobreg]:*

Name of Job Registry database. If database for Job Registry already exists and is configured, it's name should be passed here (in that case installer should be invoked with --nodbsetup option). If database wasn't earlier created, it will be created and populated with necessary tables, unless --nodbsetup option was passed to installation script.

Caution

In case of mysql RDBMS choice, attention should be given on special characters (e.g '-') which are not permitted in database names.

n) *Enter queue database name [grms_queue]:*

Name of Queue database. If database for Queue already exists and is configured, it's name should be passed here (in that case installer should be invoked --nodbsetup option). If database wasn't earlier created, it will be created and populated with necessary tables, unless --nodbsetup option was passed to installation script.

Caution

In case of mysql RDBMS choice, attention should be given on special characters (e.g -) which are not permitted in database names.

o) *Enter proxy database name [grms_proxy]:*

Name of proxies database. If database for proxies already exists and is configured, it's name should be passed here (in that case installer should be invoked --nodbsetup option). If database wasn't earlier created, it will be created and populated with necessary tables, unless --nodbsetup option was passed to installation script.

Caution

In case of mysql RDBMS choice, attention should be given on special characters (e.g -) which are not permitted in database names.

p) Enter path to certificate file [/etc/grid-security/grmscert.pem]:

Path to certificate file which will be used by GRMS Web Service interface and Broker Engine.

q) *Enter path to key file [/etc/grid-security/grmskey.pem]:*

Path to certificate key file which will be used by GRMS Web Service interface and Broker Service. This file should be owned by user on which account GRMS will be run and should have permissions set to 600 (chmod 600 key_file).

r) *Enter GRMS server Distinguished Name []:*

If grid-cert-info isn't placed in PATH environment variable, installer will ask about GRMS certificate subject. This subject is needed for proper configuration of grms-client tool.

5. If option --nodbsetup wasn't used, in this step GRMS installer will try to created Job Registry and Queue databases, and populate them with necessary tables. If these databases already exist, PostgreSQL/MySQL engine isn't running, GRMS account doesn't exists, GRMS password is wrong or access to database engine is not proper configured, GRMS installer will display following error:

```
can't   update db: can't create jobregistry database !
can't   create queue database !
could   not create tables in job registry database !
could   not create tables in queue database !
```

and installation will finish with error.

6. After optional database setup, GRMS installer will finish and display installation status message. If every thing went ok, following message will be displayed:

```
installation successfully completed
```

Otherwise, when installation finish with error, following message will be displayed:

```
installation did not completed successfully
```

Detailed actions performed by installer are recorded to install.log file.

Datailed information about GRMS instalation and configuration can be found at [7]

4.2. Gridge Data Management System

4.2.1. Overview.

DMS is a distributed component system, consisting of three separate modules: Data Broker, Repository and Data Container by default. These modules create together the basic layer of a data management environment - *DMS core*. Each of them can be treated as a separate network service communicating with external applications through the Web Service interface. For this reason it is required to apply separate installing procedures for all DMS component mentioned above.

From the administrator's point of view, the DMS can be deployed in one of three possible architectures:

- *multi-tier architecture* (named further with the symbol I) - each DMS component runs on separate machine. It is the recommended architecture with regard to performance and reliability of the system.
- *one-tier architecture with multiple containers* (named further with symbol II) - each DMS component runs on the same machine, and they are deployed on separate containers. Acceptable option.
- *one-tier architecture* (named further with symbol III - each DMS component runs on the same machine, and they are deployed on the same container. Not recommended option with regard to performance and scalability.

Beside the three components which create the basic layer of the DMS, we can distinguish the additional modules, which complete the data management environment and constitute together Data Management Suite:

- Access Portal (Web Portal) - It makes full functionality offered by DMS available to the end users. The Web Portal is a natural way to assure access to the distributed environment of DMS. The main functionality offered by the portal provides access to an intuitive interface to navigate over the hierarchy of the meta-directories and meta-files and a possibility to manage physical and logical information on each of the meta-elements.

The Access Portal was initially designed as an access interface to the Data Management System. Therefore this guide covers all information related to the administration of this component, although Web Portal constitutes currently a separate application enabling an access to DMS as well as Toth.

- Proxy (SRS Container) - it provides a SOAP interface to the external databases. Within the PROGRESS installation Proxy enables access to SRS (Sequence Retrieval System) resources - biological sets of data banks. This module, named internally as SRS Container, is an example of the succeed integration of DMS structures with the third party storage system. The SRS Container acts as a separate DMS module, combining the functionality of the Metadata Repository and Data Containers. Thereby the entries from biological data banks are treated as regular files and they can be accessed through the DMS services. The SRS Container provides an access to the bio-informatics data with use of the FTP or GSIFTP protocol which is not possible with the default SRS interfaces.
- Heimdall - Heimdall is an internal security system, based on ACL (Access Control List) conception, allowing to control the access and manage the rights to DMS resources (authorization process). It also provides user authentication services on the basis of information stored in the external database (e.g. LDAP catalogue or GAS). Heimdall is tightly integrated with the DMS, which ensures the appropriate effectiveness while processing the authentication or authorization requests.
- Toth - The Toth Logging System acts as a supporting tool for the administrators. It focuses on collecting events occurring in the monitored system. It's main role is to solve the problem of gathering events generated by various components of Data Management System. Toth Logging System was designed as an independent system and can be used in various environments and cooperate with all kinds of applications.

  The Toth Logging System was designed as an independent system and it can be used in various environments and cooperate with all kinds of applications. Therefore all information related to its maintenance, operation and administration can be found in the Toth Administration Guide.

4.2.2. Requirements.

The following components are necessary to run the DMS:

- Java virtual machine, recommended Java(TM) 2 Runtime Environment,Standard Edition 1.4.1 or higher.
- Web Server with Servlet Container. DMS modules are pre-configured to work with Tomcat container or Jetty server. It is also possible to use another servlet container, but this documentation deals with the installation process for Tomcat and Jetty server only.
- A database server for Metadata Repository. Metadata Repository is accommodated to run on Oracle or PostgreSQL engine:
  - Oracle - Oracle8i or higher recommended;
  - PostgreSQL - version 7.3 or higher is required with the additional extends: `chkpass` and `tablefunc` from `contrib` package and `plpqsql` support.
- Globus-enabled Web server. In order to run DMS with the Globus certificates (HTTPS protocol) it is necessary to deploy the Globus ws-core package from Globus Toolkit on the target Web server. Please refer to the Globus toolkit ws-core package documentation for further instructions.

4.2.3. Installation

The DMS installation procedure should be carried out in the following order:

1. Preparation of the host environment: determine the target locations (network addresses in particular) of every DMS modules - Data Broker, Repository, Data Container;
2. Install Metadata Repository (Heimdall system embedded);
3. Install Data Broker(s);
4. Install Data Container(s).

After performing these steps it is recommended to check DMS status with the help of the administrative console delivered with installation package of Data Broker.

In the next stage it is possible to install and run:

- Access Portal,
- SRS Container.

The installation procedure for each DMS module is identical. It consists of three steps performed on each account created in the previous step:

1. Web Server installation - in case of the architecture I the server has to be installed on each machine that belongs to DMS environment. In the other cases (the architecture II and III) Web Server is installed only once.
2. The installation of the selected DMS component (e.g. Data Broker).
3. The Integration of the DMS component with the Web Server.

Note

Steps 2 and 3 are performed automatically by the graphical installer. Installation of Web Server must be done manually (using appropriate shell commands).

Datailed information about GDMS instalation and configuration can be found at [9]

4.3. Gridge Authorization Service (GAS)

4.3.1. Overview

The main goal of GAS is to provide functionality that would be able to fulfill most authorization requirements of grid computing environments. GAS is designed as a trusted single logical point for defining security policy for complex grid infrastructures. As the flexibility is a key requirement, it is to be able to implement various security scenarios, based on push or pull models, simultaneously. Secondly, GAS is considered as independent of specific technologies used at lower layers, and it should be fully usable in environments based on Globus (supporting compatibility scenario with CAS) as well as other toolkits. The high level of flexibility is achieved mainly through modular design of GAS. It is divided into five logical components, with the main GAS core module (Core Functionality) responsible for performing authorization decisions based upon defined security policy, which is maintained as a set of permissions for specific subjects (e.g. user) and objects (e.g. resource).

4.3.2. Reqiurements

There is list of software packages which are need for installing the GAS service which are not included to the GAS source distribution:

- unixODBC - required for compiling the GAS server. Please use '--with-unixODBC switch to specify correct path pointing to the unixODBC installation. If is not set installation program looks to the unixODBC installation directory: /usr /usr/local

- Globus Toolkit - required for compiling GAS server with the GSI interface (for other interfaces Globus is not needed). $GLOBUS_LOCATION environment variable have to be specified so that location of the Globus toolkit installation could be properly located.

4.3.3. Instalation

Install the unixODBC drivers (look for more information at: http://www.unixodbc.org, GAS was tested with versions 2.2.5, 2.2.6, 2.2.7, 2.2.8, 2.2.9, 2.2.10)

- Install Globus Toolkit (look for more information at: http://www.globus.org, GAS was tested with versions 3.2.0 and 3.2.1)
- Check if the `$GLOBUS_LOCATION` and the `$LD_LIBRARY_PATH` environment variables were set correctly
- Install version of the SQL server which has drivers compatible with the unixODBC. GAS was tested using the MySQL server in version number from 4.0.13 to 4.0.18
- Under the SQL server create a new user account named gas_admin. You can use the following command: create user gas_admin password <your_password>
- Logon to this new user account and create database named GAS (create database named GAS)
- Modify the odbc.ini file like as shown below. Depends on configuration the `.odbc.ini` file have to be modified in the user home directory or the `/etc/odbc.ini` file (see the unixODBC manual for more information)

```
[GAS]
Driver       = /usr/local/lib/libmyodbc3.so
Description  = MySQL ODBC 3.51 Driver DSN
SERVER       = localhost
PORT         = 3306
USER         = gas\_admin
Password     =
Database     = GAS
OPTION       = 3
SOCKET       =
```

- Configure sources: (for example: ./configure --with-prefix=/usr/local --with-unixODBC=/usr/local). For configuration option see:
- Compile and link sources:

  ```
  make
  ```

- Install sources (when the GAS server will be installed in system directories, root privileges can be needed):

  ```
  make install
  ```

- Setup `config.gas` file.

Datailed information about GDMS instalation and configuration can be found at [11].

5. Testbed status

5.1. Services deployment

The table below presents current status of services deployment for Common Grid Layer. This services are componets of Globus Toolkit.

Table 1. list both, machines and the status of Grid services supposed to be installed on them.

Hostnames of resources participating in ACGT Grid testbed (as of February 2008)	Status of most important services and components of ACGT Common Grid Infrastructure layer					
	preWS GRAM	WS GRAM	Grid FTP	RFT	MDS4	ACGT CA
mango.ac.uma.es	✓	✓	✓	✓	✓	✓
iapetus.ics.forth.gr	✓	✓	✓	✓	✓	✓
grid5.kd-grid.ais.fraunhofer.de	✓	✓	✓	✓	✓	✓
grid6.kd-grid.ais.fraunhofer.de	✓	✓	✓	✓	✓	✓
acgt-node3.custodix.com	✓	✓	✓	✓	✓	✓
acgt-node5.custodix.com	✓	✓	✓	✓	✓	✓
moss1.man.poznan.pl	✓	✓	✓	✓	✓	✓
moss2.man.poznan.pl	✓	✓	✓	✓	✓	✓

Table.1. ACGT Grid – current status – February 2008.

For the Advanced Grid Middleware there are three components deployed:

- GRMS - location: moss1.man.poznan.pl
- GDMS - location: moss1.man.poznan.pl
- GAS - location: acgt-node.custodix.com

5.2. Grid Monitoring Portal.

For a dynamically changing Grid environment there is a need to provide tools for administrators for monitoring state of the infrastructure services.

Grid Monitoring Portal is one of such a tool used for quick identification of services failures or other errors. It consists of two parts:

- testing tools
- publishing service

The testing tools are a framework for running parallel tests written in Java or any programming language through executing new processes, and a set of ready-to-go tests implemented mostly using Globus Java CoG Kit and shell scripts running native clients for different grid services. Results of these tests are written into a relational database. The framework can extended very easily to fit current needs of Grid Testbed configuration.

The publishing service is used for access to results of tests stored by testing tools in a relational database. Portlet is used as client for this service which displays current result set and history of tests. The web-client was extended to work with useful possibility of launching the test directly from the portlet, giving the administrator not only the tool for monitoring the testbed status, but also the ability to force the framework to start the desired test and see on-line the possibly changing test status/result.

In the table below the map the user can see a matrix of all hosts and services – so, it is easier to see the whole status and determine what is down and up. For every host we can also check the actual state of a service by clicking on the check hyperlink. (Fig. 4). A suitable colour tells if a service is up (green) or down (red), grey fields mean that on the given host there is no such service installed.

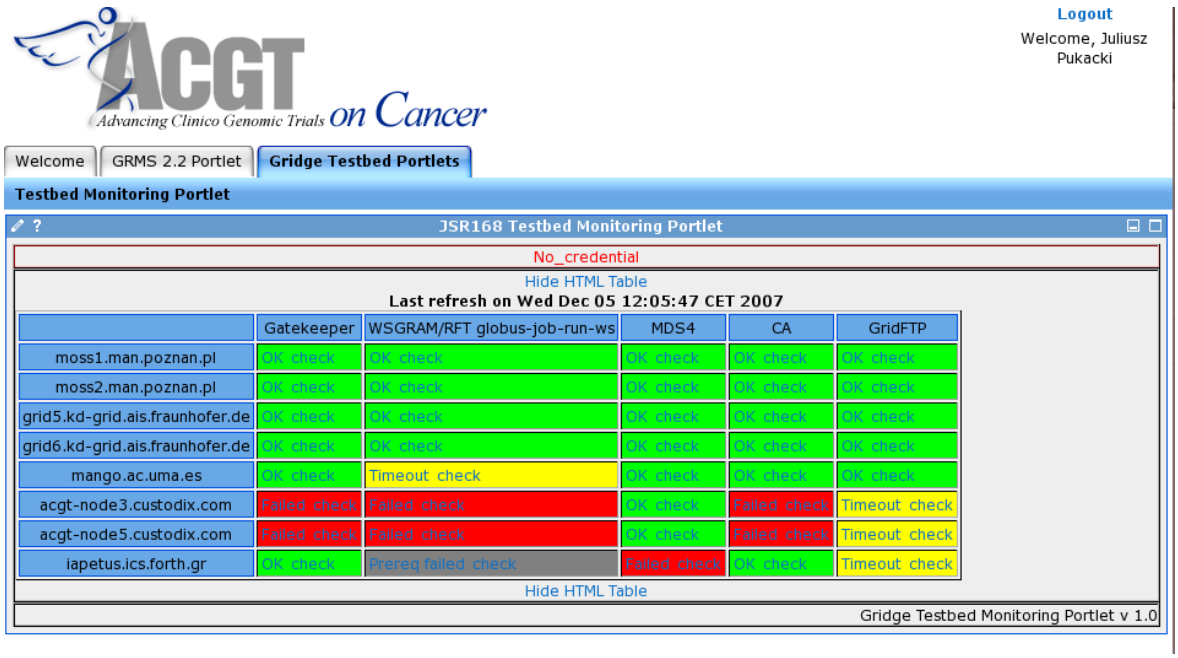

Fig.7. Grid Monitoring portlet.

After clicking on the status value -OK or FAILED- another table is shown (), which displays a list of all conducted tests and their results. If a result is FAILED then the cause could be checked by clicking on the *failed* hyperlink (), then the reason is shown to the user.

The portlet is highly configurable – the user is able to add more services, hosts and centers which appear in the database, more human readable names can be set, details for administrator, web page, graphic files can be changed, depth of the tests' history can be set.

There is also other portlet within Monitoring Portal. It shows a map of Europe and all the machines participating the testbed. It uses data gathered in the database and renders it so that users can see the actual status of the whole portlet.

Fig.8. Grid Testbed Map

References

[1] Globus Tooklit http://www.globus.org

[2] Gridge Toolkit http://www.gridge.org

[3] ACGT D4.1 Prototype and report of the ACGT GRID layer

[4] http://www.globus.org/toolkit/docs/4.0/

[5] http://www.globus.org/toolkit/docs/4.0/security/key-index.html

[6] http://www.globus.org/toolkit/security/firewalls/

[7] "GRMS Admin Guide"
http://www.gridge.org/files/grms/doc/admin/pdf/view/GrmsAdminGuide.pdf

[8] "GRMS User Guide"
http://www.gridge.org/files/grms/doc/user/pdf/view/GrmsUserGuide.pdf

[9] "GDMS Admin Guide"
http://www.gridge.org/files/dms/doc/admin/pdf/view/DMSAdminGuide.pdf

[10] "GDMS User Guide"

http://www.gridge.org/files/dms/doc/user/pdf/view/DMSUserGuide.pdf

[11] "GAS Admin Guide"
http://www.gridge.org/files/gas/doc/admin/pdf/view/gas-admin-guide-1.0.pdf

Appendix A - Abbreviations and acronyms

SOA	Service Oriented Architecture
GRMS	Gridge Resource Management System
GAS	Gridge Authorization Service
GDMS	Gridge Data Management System
RFT	Reliable File Transfer
MDS	Monitoring & Discovery Service
WSDL	Web Service Definition Language