

Grid Data Mirroring Package

User Guide for GDMP 4.0

GDMP Team: Heinz Stockinger 1), Shahzad Muzaffar 2)*, Flavia Donno 3), Aleksandr Konstantinov 4), Asad Samar 5)†, Erwin Laure 1)

1) CERN, European Organization for Nuclear Research, Geneva, Switzerland,

`Heinz.Stockinger@cern.ch`, `Erwin.Laure@cern.ch`

2) Fermi National Laboratory, Batavia, Illinois, `muzaffar@fnal.gov`

3) INFN Pisa, Italy, `Flavia.Donno@pi.infn.it`

4) University of Oslo, `aleks@fys.uio.no`

5) California Institute of Technology, Pasadena, California, `Asad.Samar@cern.ch`

November 27, 2002

Abstract

The GDMP client-server software system is a generic file replication tool that replicates files securely and efficiently from one site to another in a Data Grid environment using several Globus Grid tools. In addition, it manages replica catalogue entries for file replicas and thus maintains a consistent view of names and locations of replicated files. Files of arbitrary file format can be replicated. For Objectivity database files a particular plug-in exists. All files are assumed to be read-only.

GDMP is a collaboration between the European DataGrid [3] project (in particular the Data Management work package, Work Package 2 (WP2) [4]) and Particle Physics Data Grid (PPDG) [11]. GDMP version 4.0 is part of the official DataGrid software system and contains certain adaption for the European DataGrid testbed.

This user guide gives detailed instructions for installation and usage of GDMP. In addition, a generic replica catalogue API in C++ and the corresponding command line tool for the Globus Replica Catalogue are provided and described.

*not active anymore in the project

†not active anymore in the project

Contents

1	Introduction	5
2	New Features and Improvements in Version 4.0	5
3	Software Installation	5
3.1	Required Software	5
3.2	Installation Instructions	6
3.3	RPM Installation	9
3.3.1	Installing Source RPMs as non-root User	9
3.4	Full Installation versus GDMP Client Installation	10
4	Configuration Instructions	10
4.1	GDMP Configuration Files	10
4.1.1	gdmp.shared.conf	11
4.1.2	gdmp.conf and GDMP_CONFIG_FILE	11
4.1.3	gdmp.private.conf	13
4.2	configure_gdmp	14
4.2.1	GDMP Server Configuration with inetd	15
4.2.2	Standalone GDMP server	15
4.3	Configuration for Multiple VO support	16
4.4	The GDMP Directories	18
4.5	Security for GDMP server and clients	18
4.5.1	GDMP grid-mapfile Entries	18
4.5.2	Configuration for multiple VOs	19
4.6	Configuration for the GDMP Client Installation	20
5	Data Replication and Background for GDMP Usage	20
5.1	Directories and File Replication	20
5.2	Filenames	21
5.3	Globus Replica Catalogue (LDAP based)	21
5.3.1	Globus Replica Catalogue Security	22
5.4	Replica Location Service	23
5.5	Internal File Catalogues	23
6	Using GDMP	23
6.1	Quick Start Guide to run GDMP	24
6.2	Using the GDMP Client Installation	26
6.3	The Mechanics of GDMP	27
6.4	Security Issues for GDMP Server and Clients	28
6.5	Subscription to Remote Servers	29
6.6	Notification	29
6.7	System States for File Replication Process	29
6.8	Preliminary Space Management	30
6.9	Monitoring GDMP transfers: GDMP Heartbeat Monitor	31
6.10	Network Failures	31
6.11	Some Program Restrictions	31
6.12	Server Logfiles	31

7	GDMP Tools	33
7.1	gdmp_catalogue_cleanup	33
7.2	gdmp_get_catalogue	33
7.3	gdmp_host_subscribe	34
7.4	gdmp_job_status	35
7.5	gdmp_ping	37
7.6	gdmp_prepare_open	38
7.7	gdmp_publish_catalogue	38
7.8	gdmp_register_local_file	40
7.9	gdmp_remove_local_file	42
7.10	gdmp_replicate_get	43
7.11	gdmp_replicate_put	46
7.12	gdmp_server	48
7.13	gdmp_stage_complete	48
7.14	gdmp_stage_from_mss	49
7.15	gdmp_stage_to_mss	49
7.16	get_progress_report	50
7.17	gdmp_version	50
7.18	create_gridmapfile	50
7.19	Other tools in sbin	51
8	GDMP C++ API	52
8.1	Description of the Programming Interface	52
8.2	Usage Instructions	54
9	Support for a Mass Storage System	55
9.1	Motivation	55
9.2	Flow of Control	55
9.3	The Interface	55
9.4	Staging States	56
9.5	Interface to HRM	56
10	Replica Catalogue C++ API and Command Line Tools	57
10.1	Description of the Programming Interface	57
10.2	Usage Instructions	57
10.3	Usage of the Replica Catalogue Command Line Tools	58
11	BrokerInfo API	60
11.1	Description of the Programming Interface	60
11.2	Usage Instructions	61
12	Appendix A: Configuring GDMP with inetd	62
12.1	Inetd Background	62
12.2	Configuration Steps done by GDMP Installation Program	62
12.3	Example Configuration for inetd	63
13	Appendix B: Usage of Grid Security Infrastructure	64
14	Appendix C: Trouble Shooting for GDMP Server Configuration	65

15 Appendix D: Special Instructions for Usage in the EDG Testbed	67
15.1 Additional Hints	67
15.1.1 gdmp_register_local_file	67
15.2 gdmp_replicate_get and others	68
15.3 Miscellaneous	68

1 Introduction

The GDMP [12, 8, 13] software tools provide automatic, asynchronous replication (mirroring) of arbitrary files of any data format (called “files” in this document) and Objectivity database files in a Data Grid environment. In principle, a site, where files are created, has to notify the GDMP software which in turn notifies all the other sites in the Grid about the new files. A file is ready for mirroring only when it is guaranteed that it is closed and no other process will write into it anymore. It is the responsibility of the source site to determine when a file is ready for transfer. The destination sites receive a file listing of all the new files available at the source site and can determine themselves when to start the actual data transfer. The data transfer is done with GridFTP clients and GridFTP servers (i.e. GSI enabled and modified wu-ftp server). Replicas can be registered in a replica catalogue (based on the Globus replica catalogue using an LDAP [15] implementation or based on RLS [2]) and thus made available to the Grid. A user has to be authenticated and authorised before contacting any remote site. Authentication and authorisation are based on the Grid Security Infrastructure (GSI security) layer available from Globus.

WP2 (the Data Management work package in the European DataGrid) is working on a generic replica catalogue API. The GDMP code distribution contains a C++ API and the description of the command line tool (see Section 10) for a central Globus replica catalogue based on LDAP. Note that GDMP internally also uses the Globus replica catalogue for management of replica information.

To sum up, the software package contains three major parts:

- GDMP for file replication (command line tools and C++ client API). The main purpose of this User Guide is to explain its functionality.
- a generic Replica Catalogue API in C++ and respective command line tools (for LDAP based RC only). For a C++ API to RLS, please refer to the `edg-replica-manager` package (`listReplicas(LFN)` ([5]).
- `BrokerInfo`: this is an API to handle information coming from Scheduling system of WP1 (Workload Management, DataGrid) which is provided to Data Grid application running on the first DataGrid testbed but is not required for GDMP itself. Command line tools are available, too.

2 New Features and Improvements in Version 4.0

The following is a list of new features in GDMP version 4.0 and changes from version 3.2.

1. GDMP can be used with LDAP based RC or RLS [2]. One needs to set two new parameter in `gdmp.conf` and `gdmp.private.conf`, respectively: `REPLICA_CATALOGUE` and `GDMP_RLS_URL`
2. configuration is possible for RedHat 6.x and 7.x: Changes for `Inetd` are taken into account.

3 Software Installation

In the following section we give detailed instructions for the installation of the software package. We provide a source code distribution as well as a binary distribution and explain both installation procedures.

3.1 Required Software

The GDMP software runs and has been tested on **Linux RedHat 6.1, 6.2 and 7.3** on top of Globus Toolkit 2.0 Beta 24 (EDG distribution) and Globus 2.0 beta (Globus distribution) and VDT 1.1.5 (Globus

2.0). The GDMP software consists of several executables and a server named `gdmp_server` which runs using the Internet daemon (`inetd`) (see Section 12 for more details) at the host that produces files. The host has to be reachable by the “outside world” and cannot be behind the local firewall since permanent network connections to this machine are required. The same is true for the FTP server.

A site has to have the following software installed locally:

- GDMP software version 4.0
- Globus Toolkit 2.0 Beta 24 (special release for European DataGrid) (including the WU-FTP server), <http://marianne.in2p3.fr/datagrid/testbed1/globus/globus-2.0-b24.html>, or Globus 2.0 beta (Globus release) or VDT 1.1.5 (Globus 2.0)
- g++ compiler gcc-2.95.2
- GNU Make version 3.77 or higher
- GNU Autoconf version 2.13
- GNU libtool 1.4
- GNU automake 1.4-p2
- GNU m4 1.4
- RPMv3 or higher
- classads-0.0.edg2 (for BrokerInfo only !)

GDMP can be used with threaded and non-threaded Globus libraries but the RPMs (see Section 3.3) are built with non-threaded Globus libraries.

GDMP uses file locks and they have to be enabled on the system where GDMP is installed.

In case Objectivity files are replicated, Objectivity/DB Version 5.x or 6.x is required. Furthermore, the Objectivity bootfile as well as all the Objectivity database files have to be reachable by the FTP and the GDMP server. This guarantees a continuous data transfer from the local to the remote disk via FTP. File access via the Objectivity AMS is not supported through GDMP.

If the BrokerInfo library is built/used, also the Condor ClassAd v2.6 software must be installed and available to users.

Note that for a binary distribution the GNU Autoconf and lib tools are not required.

3.2 Installation Instructions

The following instructions apply to the source code distribution. Before starting the compilation, the following environment variables can be set or configured with `configure` (see below):

`GLOBUS_LOCATION`: base directory of the Globus installation

`OBJY_DIR`: base directory for Objectivity installation - only required if you plan to build GDMP with Objectivity support.

`CLASSAD_DIR`: If you plan to build the BrokerInfo library (not required for GDMP nor the Replica Catalogue, the environment variable `CLASSAD_DIR` has to point to the Condor ClassAd installation directory.

After unpacking the GDMP source distribution tar file, or getting the code directly from the CVS repository, change your working directory to be the GDMP base directory and run the following command:

```
./bootstrap
```

At this point you are ready to run the `configure` command. The `configure` command should be invoked as follows:

```
./configure [option] where option can be the following:
```

```
--help
```

```
--prefix=<installation dir> it is used to specify the GDMP installation dir. The default installation dir is /opt/edg.
```

```
--enable-brokerinfo it is used to enable the build of the BrokerInfo User API library. Note that the BrokerInfo API is provided by the Workload management work package and is not required for GDMP. By default this option is turned off. If the environment variable CLASSAD_DIR is not set, you can specify the ClassAd installation directory using the option
```

```
--with-classad-install=<dir>.
```

```
--enable-gdmp it is used to build the GDMP package and the Replica Catalogue User API library. By default this option is turned on. You can use the option --disable-gdmp to only build the Replica Catalogue User API library.
```

```
--with-objectivity it is used to enable Objectivity support. By default this option is turned on. If the environment variable OBJY_DIR is not set, you can specify the ClassAd installation directory using the option --with-objectivity-install=<dir>.
```

```
--with-globus-install=<dir> allows to specify the Globus install directory without setting the environment variable
```

```
--with-classad-install=<dir> specify a non-default classadd install directory
```

```
--with-globus-flavor=flavor allows to specify a specific Globus flavour. Possible values are gcc32dbgpthr, the default, and gcc32dbg.
```

```
--with-rpm-dir=<dir> By default, gmake rpm tries to build the binary and source RPMs in your working directory. One can use this option to create RPMs in a directory given by <dir>.
```

During the `configure` step, a spec file (`gdmp[-objy].spec`) will be produced in the GDMP source directory to produce a flavour specific version with or without Objectivity.

```
gmake
```

Run `gmake` in the GDMP source code directory. Be careful to use the GNU distribution of `make`. Proprietary versions do not work all the time. (GDMP, RC, BrokerInfo have only been built successfully on RedHat6.1 and RedHat6.2).

```
gmake userdoc
```

```
gmake apidoc
```

These two commands are equivalent. They make sure that the documentation exists in the directory `doc`.

```
gmake install
```

In order to install the package in the installation directory specified by the `--prefix` option during the "configure" step, you can now issue the command `gmake install` in the GDMP source tree.

```
gmake install-client
```

In contrast to `gmake install`, `gmake install-client` only installs the client part of GDMP, i.e. only the client command line tools without the GDMP server. See 3.4 for more details.

```
gmake -i dist
```

The command `gmake -i dist` will produce in the GDMP source directory a binary gzipped tar ball of the GDMP distribution. This tar ball can be unwound on a different machine. This step is not required for each installation. This tar ball can be used as source for the RPM creation.

```
gmake rpm
```

This step is only required if source and binary RPMs have to be built. In detail, it invokes `gmake -i dist` and then creates RPMs for the packages enabled during the configure process. By default only `gdmp`, `gdmp-client` and `ReplicaCatalogue` RPMs will be produced. If the configure option `--enable-brokerinfo` has been used, then also the `BrokerInfo` RPM will be produced.

By default, the RPMs are created in the current working directory but the directory can be changed with the option `--with-rpm-dir` for `configure` (see above).

The command creates the following sub directories starting from the current working directory or defined with `--with-rpm-dir`:

```
rpm/redhat/BUILD
rpm/redhat/RPMS
rpm/redhat/SOURCES
rpm/redhat/SPECS
rpm/redhat/SRPMs
```

In the directory `RPMS` you will find the binary RPMs for GDMP (e.g. `gdmp-4.0-0.i386.rpm`) and in the directory `SRPMs` the source RPMs will be created (e.g. `gdmp-4.0-0.src.rpm`)

```
gmake clean
```

Cleans up the object files and binary files created with `gmake`.

```
gmake clean-tree
```

For producing tar files and source RPMs it is recommendable to clean up all files that are not required anymore like `Makefile.in` etc. `gmake clean` removes all the files and only leaves the ones that are necessary for code distribution.

```
rpm -ba gdmp[-objy].spec
```

In order to create an RPM for GDMP 4.0, take the tar ball created during the previous step and copy it into the `rpm SOURCES` directory, usually located in `/usr/src/redhat/SOURCES`. Copy the generated spec file (`gdmp[-objy].spec`) into the `rpm SPECS` directory, usually located in `/usr/src/redhat/SPECS`. Make sure the `PATH` for root is set in such a way that the GNU autotools, `gmake` and the compiler can be used. Execute this command as root.

3.3 RPM Installation

In order to install the GDMP RPM with a given flavour (Objectivity or not) execute the following command as root:

```
rpm -ivh [--prefix <installdir>] gdmf[-objy]-4.0-0.i386.rpm
```

By default the rpm installs the software in the /opt/edg/gdmf directory. Using the `--prefix` directive, you can relocate the software and install it under a different directory.

If you want to install the rpm as a non-root user, you should have a private copy of the RPM databases in a private directory (you can copy all *.rpm files from /var/lib/rpm in a directory where you have write access) or you should have write access to the default RPM database directory /var/lib/rpm and the rpm files in that directory. Then you can use the command:

```
rpm -ivh [--prefix <installdir>] [--dbpath <RPM database dir>] \  
gdmf[-objy]-4.0-0.i386.rpm
```

where `<installdir>` is the directory where you want to install the software and `<RPM database dir>` is your own private RPM database directory (you do not need to specify such a parameter if you have write access to the default /var/lib/rpm dir and its content).

After installing the binary RPM, the user needs to make sure that `GDMP_INSTALL_DIR/lib1` is included in `LD_LIBRARY_PATH`.

Furthermore, the script `configure_gdmf` can be used as root to execute the configuration root steps.

In addition to binary RPMs, also source RPMs (`gdmf[-objy]-4.0-0.src.rpm`) are available on the GDMP web page under “Software” at: <http://cmsdoc.cern.ch/cms/grid>

For further information on RPM please consult the man pages or <http://www.rpm.org>.

3.3.1 Installing Source RPMs as non-root User

It can be done as normal Unix user (i.e. root access is not required) but the file `~/rpmmacros` needs to be in place and contain the right definitions for the topdir. In particular the topdir needs to be defined as a user subdirectory. Also, a local database directory for rpm (RPMdb) needs to be created in the user area and the `.rpmmacros` should contain the right definition for it.

In order to create a correct `~/rpmmacros` file, copy the file `/usr/lib/rpm/macros` in `~/rpmmacros` and change it to contain the right definitions for topdir and dbpath.

To populate the rpm database directory, copy the rpm files from `/var/lib/rpm` to the user created local RPM database directory.

Here is the procedure described step by step:

1. copy the source RPM that you want to install wherever you want
2. create the RPMdb directory + copy rpm files (`mkdir ~/RPMdb`)
3. create the `~/rpmmacros` file as described above, defining in this example the topdir to be `~` and dbpath to be `~/RPMdb`
4. create the RPM topdir structure as follows:

¹GDMP_INSTALL_DIR is set in `gdmf.shared.conf`

```
mkdir -p ~/rpm/redhat/SOURCES
mkdir -p ~/rpm/redhat/BUILD
mkdir -p ~/rpm/redhat/RPMS
mkdir -p ~/rpm/redhat/SRPMS
mkdir -p ~/rpm/redhat/SPECS
```

5. run `rpm -ivh --dbpath ~/RPMdb source-rpmfile.rpm`

no `-p` prefix is required, a tar.gz file will be created in: `~/rpm/redhat/SOURCES`

3.4 Full Installation versus GDMP Client Installation

By default, the GDMP software tool consists of the GDMP server plus several GDMP client tools. The GDMP RPMs `gdmp[-objj]*.rpm` by default install the GDMP server as well as all client applications. This installation is required on the Storage Element. This also means that users running GDMP client applications need to have valid accounts and access to the Storage Element. i.e. the host where GDMP is installed. We call this the **full installation** where all possible GDMP tools are installed.

In several cases (e.g. in the EU DataGrid testbed setup), users do not have access to the Storage Element but need to run GDMP client commands from Worker Nodes within the Computing Element. Thus, a **GDMP Client installation** exists that only installs a minimal set of GDMP client command line tools that are required to successfully use GDMP from remote sites. The following tools are installed:

```
gdmp_get_catalogue
gdmp_host_subscribe
gdmp_job_status
gdmp_ping
gdmp_publish_catalogue
gdmp_register_local_file
gdmp_remove_local_file
gdmp_replicate_get
gdmp_replicate_put
```

For the **client installation** there exist separate RPMs: `gdmp-client[-objj]*.rpm`. Install the GDMP client RPMs on each host from where users need to remotely access GDMP servers.

4 Configuration Instructions

Once GDMP is installed properly as described in Section 3, the GDMP server and the client applications need to be configured and the GDMP configuration files need to be edited. Details on the configuration files are given below and we refer to Section 4.2 to do the actual configuration.

4.1 GDMP Configuration Files

In the GDMP configuration file, parameters like hostname, port where the server is running, storage directories, Replica Catalogue information, GDMP server and Globus installation path etc. need to be stored. Much of the information like storage directories or locations of scripts for MSS or notification are specific to a Virtual Organisation (VO). Since a single GDMP server shall not only serve a single Virtual Organisations but several ones, we need a separation of the configuration files and give access to administrators for different VOs. The configuration files `gdmp.shared.conf`, `gdmp.conf` and `gdmp.private.conf` need to be managed as described in the following subsections.

All values are separated with a “=” and a sample configuration is given below. No extra characters are allowed after the values specified. Some variables are optional (O) and others are compulsory (C). Please refer to the configuration files themselves when you configure GDMP in order to get some additional comments. Here, we give examples for a possible configuration. In the example below we assume that GDMP is installed on host1.cern.ch and a replica catalogue service on a second machine called host2.cern.ch.

Most of the values are set automatically or can be configured with `configure_gdmp` which is explained in detail in Section 4.2.

4.1.1 `gdmp.shared.conf`

Even if several VOs are used for the GDMP server, the server has configuration parameters that are common to all VOs and thus shared among them. Thus, the file `gdmp.shared.conf` is only set up *once* per GDMP installation as opposed to the other two configuration files (`gdmp.conf` and `gdmp.private.conf`, see below). These parameters are stored in the file `gdmp.shared.conf` and contain following information:

- `GDMP_INSTALL_DIR=/opt/edg`: (C) - this variable has to be identical with the path name indicated with `./configure --prefix` for the source code distribution (i.e. it points to the GDMP install directory). GDMP client applications require this path.
- `GDMP_LOCAL_DOMAIN=cern.ch`: (O) - domain name of GDMP host.
- `GDMP_LOCAL_HOST=host1.cern.ch`: (C)- local host name where the GDMP server is installed.
- `GDMP_PORT_NUMBER=2000`: (C) - Port number on which the GDMP server is listening.
- `GLOBUS_LOCATION=/opt/globus`: (C) - Globus installation directory
- `OBJECTIVITY_DIR=/usr/local/bin`: (O) - Objectivity installation directory - only required if Objectivity database files are replicated
- `ORBACUS_DIR=/usr/local/bin`: - (O) - Orbacus installation directory - only required if MSS interface to HRM is used, see Section 9.
- `GDMP_DISK_BLOCKSIZE=1024`: (O) - GDMP uses the command `df` for checking the disk space. The default option for `df` is a block size of 1024. See Section 6.8.
- `GDMP_DISKUSAGE_FACTOR=1.1`: (O) - A file will be transferred if
$$(\text{disk_space_available}) > (\text{filesize} * \text{GDMP_DISKUSAGE_FACTOR})$$

The default value is 1.1. If it is set to than 0, the value will be used. See Section 6.8.

All the compulsory parameters above will be set either by default or with `configure_gdmp` (see Section 4.2) and normally do not need to be edited.

4.1.2 `gdmp.conf` and `GDMP_CONFIG_FILE`

Every GDMP client as well as the GDMP server needs to have the file `gdmp.conf` set up correctly in order to use GDMP commands. The following configuration file contains VO specific information as well as a link to the configuration file `gdmp.shared.conf`.

- `GDMP_SHARED_CONF=/opt/etc/gdmp.shared.conf`: (C) - points to the shared configuration file `gdmp.shared.conf`

- `GDMP_SERVICE_NAME=host/host1.cern.ch`: (C) - The GDMP server needs to use a local certificate and each client requires a service name in order to contact a GDMP server. The service name is part of the X.509 subject of the GDMP server and thus indicates the CN (Common Name) value of GDMP server certificate. We assume that the server uses a host certificate.
- `GDMP_VIRTUAL_ORG=cms`: (O) - The name of a specific VO can be given here, like cms, atlas, biomed etc. The variable should not have any " " (space char) in it. Try to use alphanumeric characters. For each VO you should have a directory `${GDMP_INSTALL_DIR}/etc/${GDMP_VIRTUAL_ORG}` and place the file `gdmp.conf` in that directory.

For instance, if `GDMP_VIRTUAL_ORG=cms` then you should have this file in the directory `${GDMP_INSTALL_DIR}/etc/cms`.

- `GDMP_CONFIG_DIR=${GDMP_INSTALL_DIR}/etc/${GDMP_VIRTUAL_ORG}`: (C) - Defines the path of the configuration directory.
- `GDMP_VAR_DIR=${GDMP_INSTALL_DIR}/var/${GDMP_VIRTUAL_ORG}`: (C) - Defines the path of the var directory.
- `GDMP_TMP_DIR=${GDMP_INSTALL_DIR}/tmp/${GDMP_VIRTUAL_ORG}`: (C) - Defines the path of the tmp directory.
- `GDMP_GRID_MAPFILE=/etc/grid-security/grid-mapfile`: (C) - For each VO, a separate GDMP grid-mapfile can be configured. The format of the file is similar to the Globus grid-mapfile except that no local user id is required. See server security in Section 4.5.
- `GDMP_SERVER_PROXY=${GDMP_INSTALL_DIR}/etc/gdmp_server.proxy`: (C) - For each VO, as separate server proxy can be used but we recommend to use only one for the entire server.
- `GDMP_PRIVATE_CONF=${GDMP_CONFIG_DIR}/gdmp.private.conf`: (C) - For each VO, a private configuration file (see Section 4.1.3) has to be defined.
- `GDMP_STORAGE_DIR=/pool/data/files`: (C) - A common directory path for all physical files has to be provided. All physical filenames then *have* to contain this path in their path names.
- `GDMP_STAGE_FROM_MSS=/opt/edg/sbin/stage_from_mss`: (O) - used for staging a file from a mass storage system to a disk pool. Refer to Section 9 for details.
- `GDMP_STAGE_TO_MSS=/opt/edg/sbin/stage_to_mss`: (O) - used for staging a file from a disk pool to a mass storage system. Refer to Section 9 for details.
- `GDMP_FILE_CATALOG_SCRIPT`: (O) create a listing of files in a directory. If you don't set this variable, `GDMP_INSTALL_DIR/sbin/create_file_export_catalog` will be used to create a file list. Refer to the `gdmp.conf` files for more detailed instructions to create this script.
- `GDMP_NOTIFICATION_FOR_REPLICATE_GET=/opt/edg/sbin/notification_get`: (O) - when a remote site has successfully transferred a file from the local site, the local server is notified and the stated script is called (see Section 6.6).
- `GDMP_NOTIFICATION_FOR_PUBLISH_CATALOGUE=/opt/edg/sbin/notification_publish`: (O) - a notification script is called when the local site publishes the export catalogue (see Section 6.6).
- `GDMP_OBJY_STORAGE_DIR=/pool/objy`: (C) - A common directory path for all physical Objectivity files has to be provided. All Objectivity filenames then *have* to contain this path in their path names.

- `OO_FD_BOOT=/pool/objy/example_federation.boot`: (O) - boot file path for Objectivity federation.
- `DBHOST=${GDMP_LOCAL_HOST}`: (C) - This is the DB host name which you want to use with `-host` option of `ooattach` for Objectivity files. This will be helpful to attach DB files to a remote federation while the files are sitting on a NFS mounted system.
- `GDMP_DEFAULT_NEW_FDID=12345`: (O) - if a new federation is created by GDMP, it will assign the following federation ID. By default it is assumed that the federation exists already.
- `GDMP_SERVER_LOG=YES`: By default, the GDMP Server logs all send/receive messages. This behaviour can be switched off by setting this variable to `NO`.
- `REPLICA_CATALOGUE=RLS`: Switch between RLS and LDAP can either be RLS or LDAP. If no catalogue is provided, LDAP is chosen by default.

By default, the file `gdmp.conf` is supposed to be in `/opt/edg/etc` but can be put to any user defined location. If a user defined location is used, the environment variable `GDMP_CONFIG_FILE` has to be set and point to the location of the file. Note that this is the only environment variable that needs to be set in order to run GDMP client applications!

The file permissions of `gdmp.conf` need to be set to 644, i.e. read/writable to the Unix user running the GDMP server and read-only to user users. `configure_gdmp` (see Section 4.2) takes care of that.

4.1.3 `gdmp.private.conf`

This file is used for Replica Catalogue configurations. GDMP can be used with the following two Replica Catalogue implementations:

- Globus Replica Catalogue based on LDAP (see 5.3)
- Replica Location Service based on SQL [2]

LDAP based RC

In `gdmp.private.conf` mainly private information about the Globus Replica Catalogue (LDAP information) (see also Section 5.3) is stored. Some information like `GDMP_REP_CAT_MANAGER_PWD` is only required if the Replica Catalogue does not support GSI authentication. We recommend to use GSI for higher security. For this information, please check with your VO since many of the parameters are VO specific like collection names etc. For the GDMP client installation, this file is not visible. The following variables need to be set:

- `GDMP_REP_CAT_HOST=ldap://host2.cern.ch:2010`: Information about the LDAP server hosting the Replica Catalogue, in the format: `<protocol>://<hostname>:<port>`.
- `GDMP_REP_CAT_NAME=replica-catalogue`: Name of the Replica Catalogue. Note that there are *no* LDAP specific *parameters* like `dc` or `cn` but only the common name of the collection!
- `GDMP_REP_CAT_MANAGER_DN=RCManager`: (C/O) these are compulsory parameters if a the Globus replica catalogue based on LDAP [15] is used. The parameters are specific to the LDAP setup and need to be checked with the administrator of the replica catalogue. It is only needed when publishing files to the LDAP replica catalogue. Note that there are *no* LDAP specific *parameters* like `cn` !

- `GDMP_REP_CAT_MANAGER_PWD=secret`: (C) - this password is required if new information needs to be inserted into the LDAP replica catalogue. For search operations, the pass word is not required. The pass word is not required if GSI is used for the Replica Catalogue.
- `GDMP_REP_CAT_CN=dc=host2, dc=cern, dc=ch`: CN (Common Name) of Replica Catalogue host. Note that here all LDAP specific parameters like *dc* are required.
- `GDMP_REP_CAT_FILE_COLL_NAME=file-collection`: (C) - Collection name of the Replica Catalogue for file collection. The value is the CN only and *no* LDAP specific parameters like *dc* or *cn* are required.
- `GDMP_REP_CAT_OBJYFILE_COLL_NAME=objectivity-file-collection`: (O) - Collection name of the Replica Catalogue for Objectivity file collection. The value is the CN only and *no* LDAP specific parameters like *dc* or *cn* are required. In more detail, files and Objectivity files are separated in the replica catalogue. All files are in one collection whereas all Objectivity files are in another collection. A collection needs to be identified by a name which is then used in the Globus replica catalogue.

The following variables normally do not need to be modified since they are constructed from the ones above:

- `GDMP_REP_CAT_MANAGER_DN=cn=${GDMP_REP_CAT_MANAGER_CN}, ${GDMP_REP_CAT_CN}`
- `GDMP_REP_CAT_URL=${GDMP_REP_CAT_HOST}/rc=${GDMP_REP_CAT_NAME}, ${GDMP_REP_CAT_CN}`
- `GDMP_REP_CAT_FILE_COLL_URL=${GDMP_REP_CAT_HOST}/\lc=${GDMP_REP_CAT_FILE_COLL_NAME}, rc=${GDMP_REP_CAT_NAME}, ${GDMP_REP_CAT_CN}`
- `GDMP_REP_CAT_OBJECTIVITY_COLL_URL=${GDMP_REP_CAT_HOST}/\lc=${GDMP_REP_CAT_OBJYFILE_COLL_NAME}, rc=${GDMP_REP_CAT_NAME}, ${GDMP_REP_CAT_CN}`

RLS based Catalogue

If GDMP is configured with RLS, none of the above parameters needs to be set except the one below which is the Local Replica Catalogue into which GDMP writes:

- `GDMP_RLS_URL=rls://testbed008.cern.ch`
URL of the Local Replica Catalogue for RLS needs to contain “rls://”

4.2 configure_gdmp

We assume that GDMP has been installed successfully as described in Section 3 and now explain the configuration step. By default, GDMP is configured in a way that it is started via the Internet daemon (`inetd`). However, the GDMP server can also be configured and started in stand-alone mode (see below). GDMP can be configured automatically with `sbin/configure_gdmp`:

```
configure_gdmp <gdmp-install-dir> <userid> <port> [<vo_name>] [standalone]
```

The variables `<vo_name>` and `standalone` are optional whereas the first three variables are compulsory. Details in the following subsections.

4.2.1 GDMP Server Configuration with inetd

The command `configure_gdmp` is executed as *root* and will properly configure the GDMP server and the system files (`/etc/services` and `/etc/inetd`). The script requires three compulsory input parameters: `GDMP_INSTALL_DIR`, `userid` and `port`. The GDMP server will run under the user-ID `userid` and on the specified port.

In detail, `configure_gdmp` updates the `inetd` configuration file for the GDMP server. `Inetd` will accept incoming requests on a certain port and then calls the GDMP server to handle the request. Note that by default GDMP will be assigned a service called “gdmp-server”. If the file `/etc/services` is edited manually, please make sure that the GDMP server name needs to be “gdmp-server-***” where “***” should only be a VO specific name if used in the EU DataGrid.

The file `GDMP_INSTALL_DIR/sbin/gdmp_server_start` handles these interactions and is configured automatically by the installation program. Note that the script `gdmp_server_start` is the main script for starting the GDMP server and is modified automatically for the specific installation. For further information on `inetd`, an example configuration and possible problem shooting refer to Appendix A.

4.2.2 Standalone GDMP server

The GDMP server can also be configured as a standalone server using the option `standalone` for the configuration script. The script also makes sure that the following environment variables are set properly. Note that the configuration script will do set the following two environment variables automatically. Only if you do *not* use the configuration script, you need to set the variables !

- `LD_LIBRARY_PATH` needs to contain the GDMP library (`libgdmp`).
- `GDMP_CONFIG_FILE` needs to point to the configuration file `gdmp.conf` for the GDMP server.

Once the server is configured properly with `sbin/configure_gdmp`, it can be started, stoped etc. via the commands:

```
/etc/rc.d/init.d/gdmp_server start|stop|status|restart|reload|condrestart
```

Note that only the owner of the server process can stop the server whereas everybody can get the status of the server and see if it is running.

Final Remarks

By default, the location of “functions” is `/etc/init.d` in RedHat. However, in the EDG testbed we use the directory `/etc/rc.d/init.d` (note the additional directory “`rc.d`”) and thus all the `gdmp_server` specific files are there.

For information on the libraries used in the testbed, see:

```
[userid@testbed008] more /etc/ld.so.conf
/usr/X11R6/lib
/usr/kerberos/lib
/usr/i486-linux-libc5/lib
/usr/local/lib
/opt/globus/lib
/opt/edg/lib
```

4.3 Configuration for Multiple VO support

The following section can be *skipped* if you only need *one GDMP installation* for a single experiment per machine. Here we describe the configuration step to use one GDMP installation for several VOs.

For the European DataGrid testbed it is required to have one GDMP installation working for several Virtual Organisations on one Storage Element (SE) since one SE is used by several VOs. In other words, for each Virtual Organisation (VO) like CMS or Atlas, a GDMP installation needs to exist and file access on the GDMP storage directories needs to be managed. For instance, the VO cms uses the disk space /home/files/cms and the VO alice uses the disk space /home/files/alice on the same machine. When users connect to the GDMP server, they have to be mapped to a particular VO and then they can operate on files that are specific to this VO. The GDMP server will listen on *one port only* for all possible VOs.

The following configuration steps contain a few more steps than pointed out in the previous subsections. Please follow these steps below rather than the previous ones.

For each VO, one needs to manage an additional `gdmp.conf` and `gdmp.shared.conf` file in the directory `GDMP_INSTALL_DIR/etc/<vo-name>`.

1. Create a Unix user **gdmp** on the SE. We assume that the GDMP server runs as user **gdmp**.
2. Create a Unix group ID on the SE with name equal to `<vo_name>`.
3. Create an area (i.e. a directory on the local disk of the Storage Element) owned by the user **gdmp** and writeable by the VO group in the area eventually exported by the SE to the closest CEs. Let's call it `/home/gdmpstorage/<vo_name>`. In case of multiple VOs, the subdirs `vo_name` should be writeable by the VO group and the directory group sticky bit should be set [e.g.: `chmod g+ws /home/gdmpstorage/|vo_name|`]. Note that `/home/gdmpstorage` should be also the mount point advertised by the close CEs for this SE if the "file" protocol is supported in the CE Information Providers. Note for each VO one `GDMP_STORAGE_DIR` needs to be configured. For this example, `GDMP_STORAGE_DIR=/home/gdmpstorage/cms`. to sum up, the following file system operations need to be done and we assume that we set up GDMP for the VO CMS:

```
mkdir /home/gdmpstorage
mkdir /home/gdmpstorage/cms
chown gdmp /home/gdmpstorage; chmod g+ws /home/gdmpstorage
chown gdmp:cms /home/gdmpstorage/cms; chmod g+ws /home/gdmpstorage/cms
```

4. Run the script `/opt/edg/sbin/configure_gdmp` as **root** with the following arguments:

```
/opt/edg/sbin/configure_gdmp /opt/edg gdmp 2000 <vo_name>
```

Note that we assume again that the server is running as user **gdmp**.

5. The `configure_gdmp` script will change the permissions and ownership of the following files and directories:

```
/opt/edg/etc/[<vo_name>]
/opt/edg/var/[<vo_name>]
/opt/edg/tmp/[<vo_name>]
```

to be owned by the user **gdm** and readable by the VO group ID.

6. The `configure_gdm` script creates the following empty files and make them owned by the user **gdm** and readable by the VO group ID:

```
touch /opt/edg/<vo_name>/etc/export_catalogue
chmod a+rw /opt/edg/<vo_name>/etc/export_catalogue
touch /opt/edg/<vo_name>/etc/import_catalogue
chmod a+rw /opt/edg/<vo_name>/etc/import_catalogue
touch /opt/edg/<vo_name>/etc/local_file_catalogue
chmod a+rw /opt/edg/<vo_name>/etc/local_file_catalogue
touch /opt/edg/<vo_name>/etc/host_list
chmod a+rw /opt/edg/<vo_name>/etc/host_list

touch /opt/edg/<vo_name>/var/progress.log
chmod a+rw /opt/edg/<vo_name>/var/progress.log
touch /opt/edg/<vo_name>/var/replicate.log
chmod a+rw /opt/edg/<vo_name>/var/replicate.log
touch /opt/edg/<vo_name>/var/replicate_debug.log
chmod a+rw /opt/edg/<vo_name>/var/replicate_debug.log
```

7. It is the task of the site manager to set the environmental variable `GDMP_CONFIG_FILE` to point to `/opt/edg/etc/<vo_name>/gdm.conf` in the profile for the VO user to which the VO certificates have been mapped to. Also the environmental variable `RC_CONFIG_FILE` needs to point to the configuration of the replica catalogue when the Replica Catalogue command line interface is used. For details refer to Section 10.
8. Edit the files `/opt/edg/etc/<vo_name>/gdm.conf` and `/opt/edg/etc/<vo_name>/gdm.private.conf` specifying missing information (see Section 4). The information regarding the hostname, the GDMP installation directory and the port number are filled in by the `configure_gdm` script. The storage directory needs to be set as follows

```
GDMP_STORAGE_DIR=<directory>/<vo_name>
```

where `<directory>` is `/home/gdmpstorage` in the example above.

9. As a final configuration step in the EU DataGrid project, the VO specific GDMP-grid-mapfiles (in `/opt/edg/etc/<vo_name>/grid-mapfile`) as well as the SE grid-mapfile in `/etc/grid-security/grid-mapfile` have to be configured with the following script. This script needs to be executed with a certain frequency in order to update the possible SEs and users that have access to the local GDMP installation.

```
sbin/create_gridmapfile
```

See Section 7.18 for more details on `create_gridmapfile`.

4.4 The GDMP Directories

When GDMP is installed correctly, the following directories are available in the GDMP installation tree:

```
bin  etc  userdoc  include  lib  tmp  sbin  var
```

The directory `bin` contains all the GDMP client applications as well as the GDMP server. `userdoc` contains the complete documentation of GDMP. `var` contains the log files created during program execution. The output of the GDMP server and all client applications are also redirected to a file in this directory (`gdmp_server_log.out`).

`etc` contains the proxy certificate to be used by the GDMP server to authenticate itself to the other Grid nodes, and the `host_list` file containing information about all the subscribed remote hosts. The file `host_subscribed` contains all host to which the local host is subscribed.

Furthermore, it contains the `import_catalogue` file containing information about all the files which are to be transferred from the remote hosts and the `export_catalogue` file containing information about the new files on the local host that must be notified to the subscribers. Finally, also the `local_file_catalogue` is stored here.

`tmp` contains temporary files maintained by the server or different clients.

`include` contains the C++ API (Application Program Interface) to GDMP as well as the Replica Catalogue subdirectories `API` and `ReplicaCatalog`, respectively, as well as internal GDMP header files in subdirectory `gdmp`.

Finally, libraries for the the C++ API (`libgdmp_client`), the Replica Catalogue (`libReplicaCatalog`) and GDMP in general (`libgdmp`) are stored in `lib`.

4.5 Security for GDMP server and clients

Before the GDMP server can be tested and used, it needs a certificate/key pair. We strongly recommend to use the `hostcert/hostkey` pair. GDMP requires that both files are concatenated and then the file `gdmp_server.proxy` is used for the GDMP server for authentication and authorisation. `configure_gdmp` does these steps automatically but requires that the `hostcert.pem` and `hostkey.pem` files are in the default location `/etc/grid-security`.

4.5.1 GDMP grid-mapfile Entries

Example: We assume the host called `host1.cern.ch` is supposed to run the GDMP server under the user “`gdmp`” and GDMP is installed under `/opt/edg`. By default, the GDMP server uses the file `gdmp_server.proxy` in the directory `/opt/edg/etc/gdmp_server.proxy` and we further assume that the server refers to the grid-mapfile at its default location `/etc/grid-security/grid-mapfile`. `host1.cern.ch` consequently has the X.509 subject name `/O=Grid/O=CERN/OU=cern.ch/CN=host/host1.cern.ch`. We now assume that the user “`Firstname Lastname`” shall be allowed to access the local GDMP installation at `host1.cern.ch`. In order to do so, the subject name needs to be inserted into the grid-mapfile of the GDMP server, i.e. the grid-mapfile looks like follows:

```
”/O=Grid/O=CERN/OU=cern.ch/CN=Firstname Lastname”
```

Note that no Unix user needs to be specified since the GDMP server will only use the grid-mapfile entries for authentication of the user but will do all file transfers and actions as user “`gdmp`”.

Whenever a client command is issued at one site, the request is sent to the local server (stated in the `gdmp.conf` file) and the local server then contacts the GDMP server on behalf of the user.

Now we extend the example and assume that we have another GDMP server where we want to get a set of files via `gdmp_replicate_get`. In detail, the user “`Firstname Lastname`” issues the command

gdmp_replicate_get locally on host1 which then tries to get files from host2.cern.ch. The user “Firstname Lastname” only needs to be registered in the local grid-mapfile of host1. The actual file transfer will be done by the GDMP server which needs to be registered in the grid-mapfile of host2. Thus, the grid-mapfile of host2.cern.ch needs to have the following entry and we assume that also on host2 the GDMP server is running under the Unix user “gdmp”:

```
"/O=Grid/O=CERN/OU=cern.ch/CN=host/host1.cern.ch" gdmp
```

Vice versa, the subject name of host2 needs to be inserted into the grid-mapfile of host1 in order to allow data transfer from host1 to host2.

To sum up, we assume that GDMP server at host1 knows 3 users and allows host2 to replicate files. Thus, the grid-mapfile of host1 can look like follows:

```
"/O=Grid/O=CERN/OU=cern.ch/CN=Firstname1 Lastname1"  
"/O=Grid/O=CERN/OU=cern.ch/CN=Firstname2 Lastname2"  
"/O=Grid/O=CERN/OU=cern.ch/CN=Firstname3 Lastname3"  
"/O=Grid/O=CERN/OU=cern.ch/CN=host/host2.cern.ch" gdmp
```

At host2, the grid-mapfile can look like follows, where the local host trusts other 2 people locally and all the users that are registered at host1:

```
"/O=Grid/O=CERN/OU=cern.ch/CN=UserA Lastname1"  
"/O=Grid/O=CERN/OU=cern.ch/CN=UserB Lastname2"  
"/O=Grid/O=CERN/OU=cern.ch/CN=host/host1.cern.ch" gdmp
```

4.5.2 Configuration for multiple VOs

We now outline how multiple VOs can be configured and users can be assigned to a certain VO. We assume that the user with CN=FirstnameA LastnameA is member of the VO CMS and the GDMP_CONFIG_FILE environment variable is set to /opt/edg/etc/cms/gdmp.conf. Note the general syntax:

```
GDMP_INSTALL_DIR/etc/<VO_NAME>/gdmp.conf
```

Since the VO wants to give exclusive access only to people in CMS, for the VO cms a separate grid-mapfile is required (will be done automatically with `configure_gdmp`). For instance, it is stored in /opt/edg/etc/cms/gdmp-mapfile. In this grid-mapfile, only CMS users have to be inserted and the GDMP server will look for the user authorisation only in this file.

Final remark

By default GDMP uses the /etc/grid-security/grid-mapfile. It is not necessary that each VO should have its own grid-mapfile. The advantage of the VO specific grid-mapfile is that this will add more security for that VO otherwise every user who's DN is in /etc/grid-security/grid-mapfile file is allowed to use that VO. The disadvantage of VO specific file is that now we should have DN of remote gdmp servers in both /etc/grid-security/grid-mapfile and VO specific grid-mapfile.

4.6 Configuration for the GDMP Client Installation

The GDMP Client Installation can be configured with the script `sbin/configure_gdmp_client` with the following arguments:

```
configure_gdmp_client <gdmp-local-server> <gdmp-install-dir> [<gdmp-vo>]
```

where the `gdmp-local-server` and `gdmp-install-dir` are compulsory arguments and the `<gdmp-vo>` is optional.

Furthermore, the configuration files `gdmp.conf` and `gdmp.shared.conf` will be set properly.

5 Data Replication and Background for GDMP Usage

GDMP is a file replication tool for replicating read-only files of any data format. In addition to transferring files from one site to another and notifying about new files created locally, a replica catalogue is used to store file information about all sites in a Virtual Organisation and having GDMP installed. For details on replica catalogues and file replication refer to the DataGrid WP2 design document [9]. GDMP can either be used with the LDAP based Globus Replica Catalogue or a distributed Replica Location Service (RLS): see below.

Although any file format can be used, GDMP has a particular plug-in for Objectivity files where files are “attached” to an Objectivity federation². Most of the command line tools are by default configured for files rather than Objectivity files.

5.1 Directories and File Replication

GDMP manages files stored in a particular *storage root directory* on a local disk or mounted disk pool. It requires all physical files to be stored in this directory structure which can be configured for each GDMP installation and thus for each storage system. Note that we distinguish between “files” (this can be any arbitrary file format like ROOT, ZEBRA, etc.) and Objectivity files which require particular replication steps. GDMP also requires two different storage root directories for these two file formats.

We start with an example. We assume that a large disk pool is mounted on a host `host1.cern.ch`. Files are stored in the directory `/data/run1/`. In the directory `run1` several subdirectories can exist and a possible directory layout is as follows:

```
/data/run1/day1/file1  
/data/run1/day1/file2  
/data/run1/day1/file3  
/data/run1/day2/fileA  
/data/run1/day2/fileB
```

GDMP requires a common root path for the directory structure since it manages several files in the replication process. In our example the common directory and thus the storage root directory for files is `/data/run1`. GDMP uses a configuration variable called `GDMP_STORAGE_DIR` which needs to be set in the file `gdmp.conf` (see Section 3).

For Objectivity files a similar directory structure is required and the variable `GDMP_OBJY_STORAGE_DIR` must point to the storage root directory for Objectivity files.

²We do not discuss the details of Objectivity but point out that for Objectivity database files additional replication steps are necessary, as indicated when GDMP command line tools are explained.

The main task of GDMP is to mirror the directory structure of one storage system (called *Storage Element* in DataGrid terminology) to another. Thus, a storage root directory is required on both sites that participate in the mirroring process. Note that the storage root directory does *not* have to be identical on all Storage Elements but can be chosen and configured based on the local directory structure. If we now assume that all the files above need to be replicated to a Storage Element at Fermilab, the destination host first needs to set up a storage root directory. For example, on `host1.fnal.gov` (at Fermilab), `GDMP_STORAGE_DIR` has the value `/largedisk/cms/production/run1`. The GDMP replication process now can replicate files from the storage root directory on the source machine to the one on the destination machine.

5.2 Filenames

When files are replicated, identical files (replicas) exist at multiple locations and need to be identified uniquely. A set of identical replicas is assigned a *logical filename* (LFN) and each single physical file is assigned a *physical filename* (PFN). In [9] we also include a *transfer filename* (TFN) but we do not discuss it further and we will assume that all PFNs have the complete paths used by the Storage Element's file system to refer to files resident on disk. The PFN also contains the host name i.e. the domain name of the Storage Element (see "Section 4.2 File Replication", in [9]) where the file is located and accessible via a Grid file transfer tool such as a GSI-enabled FTP server.

A typical example of a physical filename under the above assumptions is as follows:

```
pfn://host1.cern.ch/data/run1/day1/file1
```

We observe that when PFNs are used with GDMP, the prefix "`pfn://`" is not required. Once the file is created and the PFN is available, it can be inserted to the replica catalogue that provides a global name space for file replicas. See "Section 5.1.1 Replica Catalog" in [9] for details on replica catalogue issues. In addition to the PFN, a logical filename must be assigned to the physical file. In the current version of the Globus replica catalogue, the LFN is equal to the last component of the PFN, i.e., to the file name including parts of the directory structure. This is a current restriction that will be removed in future versions. Thus, the logical filename is created automatically by GDMP from the physical filename complying with the implicit mapping enforced by the replica catalogue and for the physical filename in the example above it looks as follows:

```
day1/file1
```

5.3 Globus Replica Catalogue (LDAP based)

In this subsection we describe how the Globus replica catalogue needs to be set up to interact with GDMP.

GDMP uses the Globus replica catalogue implementation which is based on LDAP. For details on the Globus replica catalogue refer to the user guide in [7]. The replica catalogue service is based on the LDAP protocol and a database backend where all replica information is stored. In principle, any possible LDAP server and a corresponding database backend can be used. The currently adopted solution is to use the OpenLDAP server and one of the database backends supported by OpenLDAP. For Solaris and Linux platforms we have tested the SleepyCat database backend (Sleepycat Berkeley DB 2.7.7: (08/20/99)). For Linux we tested the OpenLDAP database backend `ldbm`. Details on LDAP configuration can be found in [6].

As pointed out in Section 3, the following LDAP replica catalogue variables have to be set in `gdmp.private.conf`. We give example values:

```
GDMP_REP_CAT_HOST=ldap://host2.cern.ch:2010
GDMP_REP_CAT_NAME=replica-catalogue
GDMP_REP_CAT_MANAGER_DN=RCManager
GDMP_REP_CAT_MANAGER_PWD=secret
GDMP_REP_CAT_CN=dc=host2, dc=cern, dc=ch
```

Since we distinguish between files and Objectivity files as regards storage root directories, we need to do the same for the replica catalogue configuration. The Globus replica catalogue provides the concept of *collections* that group several logically related files. Collections are not used explicitly in the GDMP user interface, but one collection for files and another collection for Objectivity files are implicitly created by GDMP in the replica catalogue to manage the two types of files. Thus, the following configuration variables have to be set in `gdmp.private.conf` to specify the two possible collections for files and Objectivity files, respectively.

- `GDMP_REP_CAT_FILE_COLL_NAME=file-collection:`
- `GDMP_REP_CAT_OBJYFILE_COLL_NAME=objectivity-file-collection`

Note that all sites inserting information in the replica catalogue of a single Virtual Organisation, such as an experiment in a HEP environment, need to use the same file collection.

5.3.1 Globus Replica Catalogue Security

The current (March 2002) Globus replica catalogue libraries as distributed in Globus 2.0 Beta 21 (EDG distribution), Globus 2.0 Beta 2 (Globus distribution) do not support GSI authentication for Replica Catalogue access.

A patch to the libraries has been provided in order to use normal GSI authentication and authorisation for remote Replica Catalogues that are using LDAP servers. The original patch can be found at the NorduGrid project page (<http://www.quark.lu.se/grid/>) but it is also packaged and distributed by EDG and available on the EDG software distribution web page.

GDMP client commands that access the Replica Catalogue by default (e.g. `gdmp_publish_catalogue`, `gdmp_replicate_get` and `gdmp_replicate_put`) assume that the patched libraries are installed locally. Since the actual update to the Replica Catalogue is done by the GDMP server that uses the host proxy certificate, the host proxy certificate needs to be given to the administrators of the Replica Catalogue server where it needs to be registered. In other words, the standard way of GSI authentication is used.

If the patched libraries are not installed and the Replica Catalogue server allows authentication and authorisation via clear text pass words, the GDMP client commands can use the option. In this case, the pass word for updating the LDAP Replica Catalogue server needs to be read from the configuration file `gdmp.private.conf`. Consequently, GDMP works for both options. For details on how to use clear text pass word updates (command line option `-C`) refer to the GDMP client commands.

If one uses the Replica Catalogue command line tools or API, the specific user also needs to be registered in the Replica Catalogue server for read/write access.

Configuration Details

For writing entries into the Replica Catalogue the GDMP server installation needs the patched `globus_replica_catalog_*` libraries, properly installed SASL plug-in for GSI-GSSAPI authentication (it comes with Globus, but you have to make sure you have your `SASL_PATH` environment set up properly to point to the directory with `_globus_sasl` plug-ins). This especially important on systems with non-Globus LDAP installed.

In addition, one of the following libraries needs to be installed:

- libglobus_sasl_gssapi_gsi_* (EDG distribution)
- libgssapi_v2_gsi_*

5.4 Replica Location Service

GDMP can also be interfaced with RLS [2] by configuration the Local Replica Catalogue in `gdmp.private.conf`. Here we give a very brief overview of how RLS is used but refer to [2] or the WP2 web page for more details.

The Replica Location Service (RLS) consists of several Local Replica Catalogues (LRC) and one or more Replica Location Index (RLI). All updates and inserts are done at the LRC and thus each GDMP server is configured to update a single LRC. This can be any LRC but it is recommended to use an LRC on the same site or even the same Storage Element. GDMP will then update the LRC with the LFN and the PFN information and the LRC then propagates the updates to the RLI.

To sum up, the following two configuration parameters have to be set in `gdmp.conf` and `gdmp.private.conf`, respectively:

```
REPLICA_CATALOGUE=RLS
GDMP_RLS_URL=rls://hostname.cern.ch
```

5.5 Internal File Catalogues

GDMP uses a few catalogues that are used for internal book keeping and monitoring of the replication process. Once a site has finished writing a set of files (or just a single file), every single file needs to be registered in a *local file catalogue* which only contains files that are available at the local site. This catalogue contains the physical filename and logical file attributes like logical filename, file size, creation time, CRC checksum, file type. The local file catalogue is hidden from outside users and is thus only visible to the local GDMP server. The client application `gdmp_register_local_file` is used for inserting files to this catalogue.

At a certain point in time, a site can decide to publish its local files to other Grid sites using `gdmp_publish_catalogue`. In detail, all file entries of the local file catalogue are written into the replica catalogue³ and also sent to subscribed sites at remote sites (see Section 6.5). A list of all newly published files and their related information is written to a local *export catalogue*. The consumer site that wants to receive files creates an *import catalogue* where it lists all the files that are published by the producer and have not yet been transferred to the consumer site. The import catalogue holds the host name of the FTP server and all related physical and logical file information for each file. Figure 1 illustrates this model graphically.

When files are published, all required file information is read from the local file catalogue. Since the catalogue holds file attributes like size, during the execution of the publish command the files to be published do not need to reside on their physical location on disk but can already have been staged to a mass storage system. Note that files have to be at the disk location when `gdmp_register_local_file` is called since file size and CRC check sum are automatically created by GDMP and then stored in the local file catalogue.

6 Using GDMP

In this section we provide information on using the GDMP server and client applications and refer to Section 7 for detailed parameters for the command line tools.

³An insertion to the replica catalogue can also be disabled.

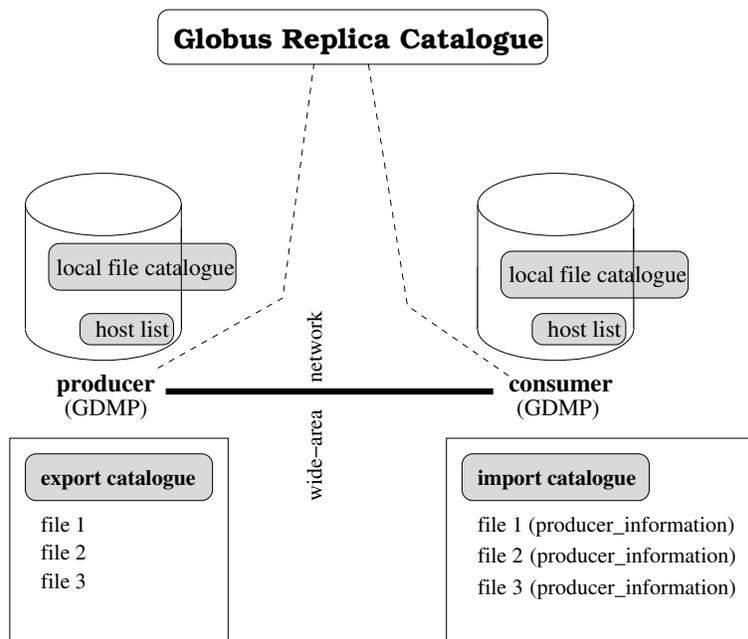


Figure 1: The role of the local file, export, import catalogues

Note: Here, we provide a quick guide. For some more usage instructions for the EU DataGrid testbed refer to the specific documents on the following GDMP web page:

<http://cmsdoc.cern.ch/cms/grid/edg-testbed>

You can reach this link also through the GDMP web page directly under the side bar “Documentation”.

A definition of the Storage Element for the EDG testbed is given in a document (“GDMP User Instructions for the European DataGrid Testbed”) on the page above. Although that document is out-dated since it refers to GDMP 2.1, Sections 1 and 2 are still valid and also Section 3 gives some basic examples.

6.1 Quick Start Guide to run GDMP

Follow these steps to run the GDMP server and transfer files quickly and securely.

Example: Assume that files have been created on testbed008 and have to be transferred (replicated or mirrored) from site testbed008 (A) to tbed0079 (B). In other words, users at site tbed0079 want to retrieve a set of files from testbed008 by issuing `gdmp_replicate_get`. Several GDMP client commands are outlined briefly and you can find more detailed examples for each client command in Section 7.

1. Server Installation - Configuration:

Register `gdmp_server` as an `inetd` service or standalone server on site testbed008 as user ‘`gdmp1`’ and site tbed0079 as user ‘`gdmp2`’. All these steps are described in the sections on installation (Sections 3 and 4). Both servers listen on port 2000.

There are no restrictions on ‘`gdmp1`’ and ‘`gdmp2`’. In other words, the GDMP servers can be installed and configured with any possible Unix user.

You can specify the grid-mapfile in the configuration files of the server The default is `/etc/grid-security/grid-`

You can make your own grid -mapfile on the same format as used by Globus and it will work but make sure that it has correct file access permissions.

Result: The server should start. The output of the server is logged in the file:

```
GDMP_INSTALL_DIR/var/gdmp_server.out
```

2. *tbed0079*: get grid proxy and check GDMP_CONFIG_FILE

We issue client commands with the Unix user called userB and the CN of the user (e.g. CN=UserB) is already registered in the grid-mapfile of *tbed0079*. The client needs to get a valid proxy using `grid-proxy-init`.

```
[userB@tbed0079] grid-proxy-init
Your identity: /O=Grid/O=CERN/OU=cern.ch/CN=UserB
Enter GRID pass phrase for this identity:
Creating proxy ..... Done
Your proxy is valid until Tue Jan 22 23:03:02 2002
```

For each user the environment variable `GDMP_CONFIG_FILE` needs to be configured and set correctly when the user has logged on. However, if the variable is not set, do the following (assuming that the default GDMP installation is used without any VO):

```
[userB@tbed0079] setenv GDMP_CONFIG_FILE /opt/edg/etc/gdmp.conf
```

3. *tbed0079*: run `gdmp_host_subscribe`

Run `gdmp_host_subscribe` as a user 'userB', giving the host and port of the GDMP server on *testbed008*.

```
[userB@tbed0079] gdmp_host_subscribe -r testbed008.cern.ch -p2000
```

Result: This should add the information about the host on *tbed0079* in the `GDMP_INSTALL_DIR/etc/host_list` file at *testbed008*.

At the local site, entry of the remote host appears in the file `host_subscribed`.

4. *testbed008*: run `gdmp_register_local_file`

Run `gdmp_register_local_file` on *testbed008* as user 'userA' where userA (CN=UserA) needs to be in the local grid-mapfile on *testbed008* and the user needs to have a valid proxy (similar as above). In addition, the environment variable `GDMP_CONFIG_FILE` must be set properly.

```
[userA@testbed008] gdmp_register_local_file -d /data/run1
```

All files stored in the storage directory (e.g. `/data/run1` in Section 5.1) are registered.

In case of Objectivity files: the lock-server of the related bootfile should be running.

Result: The files in the corresponding directory are inserted into the local file catalogue.

5. *testbed008*: run `gdmp_publish_catalogue`

Run `gdmp_publish_catalogue` on site *testbed008* as user 'userA'.

```
[userA@testbed008] gdmf_publish_catalogue
```

Files can only be published if they are in the local file catalogue, i.e. `gdmf_register_local_file` must have been used in advance.

Result: This should create entries in the file `GDMP_INSTALL_DIR/etc/import_catalogue` on `tbed0079` which will have the details of files at `testbed008`. Replica information by default is inserted to the replica catalogue. In addition, the newly published files will appear locally on `testbed008` in the export catalogue `GDMP_INSTALL_DIR/etc/export_catalogue`.

6. *tbed0079*: run `gdmf_replicate_get`

Run `gdmf_replicate_get` on `tbed0079` to actually start the file transfer.

```
[userB@tbed0079] gdmf_replicate_get
```

In case of Objectivity files, the lock-server of the `OO_FD_BOOT` should be running.

The `GDMP_STORAGE_DIR` should have enough space to hold these files.

Result: This should transfer the files from `testbed008` to `tbed0079` and validate them. In case of Objectivity files, the files are attached to the federation at site B. Replica information by default is inserted to the replica catalogue.

6.2 Using the GDMP Client Installation

All the examples above assume the full installation (see Section 3.4) but now we briefly explain how to use GDMP client commands on a host where the GDMP client installation is configured properly. You first need to make sure that your environment variable `GDMP_CONFIG_FILE` points to the VO specific `gdmf.conf` file. For the example here, we assume that the GDMP client installation is installed on the Worker Node `lxshare0224.cern.ch` and the possible Storage Elements are `testbed008.cern.ch` and `tbed0079.cern.ch`. Note that both SEs *are not available in the EDG testbed* but we only give these examples for reference. A valid grid-proxy needs to exist, too.

For every client command, one needs to contact a remote GDMP server since the GDMP server does all the necessary authentication and authorisation steps and then executes the service on behalf of the user. Thus, the remote GDMP server is always contacted with the command line options `-S` for the server and `-P` for the port of the remote server.

```
[hst@lxshare0224] gdmf_ping -S tbed0079.cern.ch -P 2000
Message: The local GDMP server tbed0079.cern.ch:2000 is listening and you are an
authorized user [ Thu Mar 28 09:04:11 2002 ]
```

If we want to register a file that is reachable from a GDMP server, we first need to identify the GDMP server we want to contact with `-S` and `-P`.

```
[hst@lxshare0224] gdmf_register_local_file -S testbed008.cern.ch -P 2000 \
-d /home/edg-replica-manager/testfiles/cms
Server Message [testbed008.cern.ch:2000]: A client has been started to
register the requested files. [ Thu Mar 28 09:09:31 2002 ]
Message: Server Log ID=gdmf_testbed008.cern.ch_32231_1017302971_1 [ Thu Mar 28 09:09:31 2002 ]
```

In addition, one can specify the virtual organisation one belongs to by using the option `-V`.

General Remarks

In principle, there is not much difference in using GDMP client tools on a full installation or a client installation only. If you use it on a client installation (which is normally the case in the EU DataGrid testbed) you always need to add the optional parameters `-S` and `-P` to the client command line tool. Most of the examples below assume the full installation but they can easily be applied to the client installation, too.

If any of the options `-S`, `-P` or `-V` is used, this parameter is used not the one from a possible GDMP configuration file!

6.3 The Mechanics of GDMP

Registering a file in the Local File Catalogue

Once files are available on disk for replication to remote sites, the files need to be registered in a local file catalogue. This catalogue keeps track of all files that GDMP manages. GDMP mirrors *file sets* (a set can also contain just a single file) and automatically detects which files have been added to the local catalogue but have not been published yet. By inserting files into the local catalogue, GDMP gets control over the files and the corresponding replica information. Consequently, every single file needs to be registered first before it can be replicated. `gdmp_register_local_file` needs to be used for this. A possible *local file catalogue* looks like follows (in one single line):

```
file:host1.cern.ch_.pool.data.testfiles:file1:1012039680:
603077316:1001660649
```

Which corresponds to:

```
filetype:file_id:relative_file_path_under_the_root_dir:size:checksum:timestamp
```

`filetype` can either be “file” or “objectivity”. Note that `file_ID` corresponds to a unique logical file ID which is currently maintained by GDMP and cannot be changed. `checksum` is created by GDMP using the checksum command line tool. `timestamp` is a integer value for a file modification time.

Creating the Export Catalogue

The export catalogue contains information about all the files which are ready to be exported to other sites. The program `gdmp_publish_catalogue` is the trigger for the replication mechanism. This program has to be called when new files are ready for transportation to another site. It will compare the current and old `local_file_catalogue`. New file entries are written in the file `GDMP_INSTALL_DIR/etc/export_catalogue` which lists all the new files that have been added to the local file catalogue since the last time `gdmp_publish_catalogue` was called.

A possible *export catalogue* looks like follows:

```
file:host1.cern.ch_.pool.data.testfiles.file1:file1
```

Which corresponds to:

```
filetype:file_ID:filename
```

`file_ID` corresponds to a unique logical filename which is currently maintained by GDMP and cannot be changed. `filename` is the relative path starting from the directory after `storage_root_dir` which is either `GDMP_STORAGE_DIR` or `GDMP_OBJY_STORAGE_DIR` (depending on the file type).

Publishing the Export Catalogue

The file update is based on the following fact: newly added files to the local file catalogue are detected and a difference between the new and the old local file catalogue is created. Based on the differences with the old catalogue, the export catalogue is created and transferred to all hosts subscribed and listed in the file `GDMP_INSTALL_DIR/host_list`. A client can only send this host list to a remote server if the user running the client is present in the grid-mapfile(s) being used by the FTP server and the `gdmp_server` on the remote machine. The export catalogue is renamed to `import_catalogue` at the destination site in order to distinguish between imported export catalogues and locally created export catalogues. Thus, the import catalogue holds the list of files which have been created newly by a remote site.

A possible *import catalogue* looks like follows:

```
file:host1.cern.ch_.pool.data.testfiles.file1:file1:
/pool/data/testfiles:host1.cern.ch:3001:1
```

Which corresponds to:

```
filetype:file_ID:filename:storage_root_dir:hostname:port:flag
```

`file_ID` corresponds to a unique logical filename which is currently maintained by GDMP and cannot be changed. `filename` is the relative path starting from the directory after `storage_root_dir` which is either `GDMP_STORAGE_DIR` or `GDMP_OBJY_STORAGE_DIR` (depending on the file type). `hostname` and `port` are the host name and port of the remote host where the file is located. In general, all information in the import catalogue refers to the remote file where it resides physically. `flag` is an internal flag to GDMP which is used when a remote file needs to be staged and is not registered in the replica catalogue.

Transferring the files

The remote site can decide when to start the data transfer from the remote to the local site. The program `gdmp_replicate_get` uses an FTP client library to securely transfer files. The program transfers each of the files listed in the import catalogue automatically to the local site. Each file is validated on arrival using the CRC checks, (is attached via `ooattachdb` in case of an Objectivity file to the local federation) and the file entry is deleted from the import catalogue. Finally, a file is registered in the replica catalogue.

6.4 Security Issues for GDMP Server and Clients

A GDMP server must run at each data production site. The port generally used is 2000, however you can set any other port through the `inetd` by editing the `/etc/services` file. Note that you will need root privileges to do this.

The server flags `-m` and `-l` can be used to specify the grid-mapfile and a Grid log file, respectively.

Note that the server uses its own service certificate and creates its Grid proxy automatically. Furthermore, the proxy is acquired for an unlimited time while client proxies are created by the user and by

default are restricted to 12 hours. For large data transfers, the client proxy might not be available long enough to transfer several Gigabytes. Thus, it is recommended that the transfer time for a set of files be estimated and the proxy time be adjusted accordingly with the `-hours` option:

```
grid-proxy-init -hours xxx
```

where xxx corresponds to the number of hours the proxy will be available.

6.5 Subscription to Remote Servers

The Data Grid is most efficient when many hosts are part of the whole Grid and hence data is available in many different sites. Since a new site has to announce that it is available in the Grid and ready to get notified about the creation of files and replicas, the program `gdmp_host_subscribe` is used to subscribe the local host to any remote host. The local host will then be integrated into the host list of the remote host. When a remote host has written new files, the notification message and the export catalogue are transferred to each host in the host list. Currently, a local host has to subscribe to each remote host separately.

Even if a private grid-mapfile is used, the grid-mapfile which is used by the GridFTP server (normally, `/etc/grid-security/grid-mapfile`) needs the entries of remote servers (hosts) that are allowed to transfer files.

6.6 Notification

When a user publishes a local file catalogue with `gdmp_publish_catalogue`, a remote server gets notified and calls a configurable script which can then be used by external programs to start a data transfer request. In principle, `gdmp_replicate_get` can be executed and a fully automatic replication process can be set up. On the other hand, a similar notification script is called at the producer site when a consumer has successfully replicated a file. Thus, producers can keep track of consumers requesting and replicating files and can delete files again if local storage space is required. The following two variables in `gdmp.conf` need to point to the notification scripts:

```
GDMP_NOTIFICATION_FOR_REPLICATE_GET  
GDMP_NOTIFICATION_FOR_PUBLISH_CATALOGUE
```

As for the replicate get notification, GDMP will pass 2 arguments to this script: 1) full file name 2) host which has transferred the file. A possible example is as follows:

```
GDMP_NOTIFICATION_FOR_REPLICATE_GET /root/dir1/file1 host1.fnal.gov
```

For the publish notification, three arguments are passed to the script: 1) host name which has published files 2) file type 3) filename which contains the list of all newly published files. It has the same format as import catalogue, and so one can pass this to `gdmp_replicate_get` with the option `-c`. For example, a set of files has been published:

```
GDMP_NOTIFICATION_FOR_PUBLISH_CATALOGUE host1.fnal.gov file /some/dir/catalogue
```

6.7 System States for File Replication Process

In the entire replication process, we distinguish several system states that are indicated by files with particular extensions. Thus, one can check the current status of a file transfer and discover possible problems. All the system states are stored in the directory `var` in the GDMP installation tree.

We now assume that a file “largeFile.extension” is transferred. In particular, a client at site A requests the file from site B using `gdmp_replicate_get`. The file name is first retrieved from the import catalogue.

The **file ID** (stored in the local file catalogue) is used for indicating files and a particular extension (see below) is added to a status file of 0 size in the directory `var`.

The file ID contains the host name and all path information including the actual file name. In principle, the file ID corresponds to a logical filename and is unique for all replicas of this file. A file that should be transferred from `host1.cern.ch/data/file1` has the following file ID:

```
host1.cern.ch_.data.file1
```

After the host name a “_” is attached and “/” is converted to “.” The following transfer states are defined:

- **file_ID.stat**: the transfer of a file is currently going on or the file has been requested for transfer.
- **file_ID.replicated**: a file has been copied to its destination but not yet validated nor registered.
- **file_ID.validated**: a file has been validated locally after the copy process. The validation contains file size checks and CRC checksum comparison with the original file at the remote site.
- **file_ID.registered**: a file has been successfully validated and registered to the local file catalogue.
- **file_ID.transferred**: the file has already been successfully transferred, validated and registered but not yet deleted from the import catalogue. The tool `gdmf_catalogue_cleanup` needs to be used to delete these files plus the entries in the import catalogue. Once the file is in “transferred state”, it can be used by applications.
- **file_ID.notified**: a remote site has been notified about the correct file transfer (including validation and registration).
- **file_ID.req**: a file is requested from a remote site and is on tape rather than on disk. A staging request is initiated (see Section 9 for details on mass storage systems).
- **file_ID.done**: This file only exists for a very short time: when a staging requests has been done at a remote site (from MSS to disk) and the local server can start to get a file. Once the file transfer starts, the file is renamed to **.stat**.

In addition to the successful states, we also have a few error states:

- **file_ID.not_validated**: file validation failed
- **file_ID.not_registered**: file is not registered to the local file catalogue
- **file_ID.not_notified**: file notification failed
- **file_ID.error**: an error has occurred during the attachment of Objectivity files.

6.8 Preliminary Space Management

For all data transfers done with `gdmf_replica_get` and `gdmf_replicate_put` GDMP does some preliminary space management. Before a file transfer is started, the GDMP server checks the available disk space on the disk where `GDMP_STORAGE_DIR` is located and only transfers the file if enough disk space is available. The variables `GDMP_DISK_BLOCKSIZE` and `GDMP_DISKUSGAE_FACTOR` in the configuration file `gdmf.private.conf` can be set accordingly.

6.9 Monitoring GDMP transfers: GDMP Heartbeat Monitor

In order to find out if two full GDMP installations work properly and over a long period of time, a GDMP Heartbeat Monitor setup exists that constantly creates and transfers files between two GDMP hosts. In principle, the Heartbeat Monitor consists of a few scripts that use two existing GDMP installations. Note that the GDMP Heartbeat Monitor is not part of the GDMP 4.0 release but can be down-loaded separately from the GDMP web page (see HTTP link below) and configured with GDMP.

Each GDMP Heart Beat site has a dummy file (automatically created for you when you run the `gdmp_hb_setup.pl`). `gdmp_hb_publish.pl` creates a unique link in `GDMP_STORAGE_DIR` for this file and then registers that link and publishes it to other hosts. When remote sites receives this published information, then GDMP will start the `gdmp_hb_replicate.pl` which will transfer this file. GDMP will then make a backup of this file if not already taken and deletes the transferred file. As each site is going to publish the same file again and again with different names, only one backup is enough. This backup will be copied by `gdmp_hb_stage_from_mss.pl` into `GDMP_STORAGE_DIR` when someone make request for file staging. `gdmp_hb_stage_to_mss.pl` is actually used to take the backup. When a file is successfully transferred, GDMP calls `gdmp_hb_stage_to_mss.pl` which take the backup for the transferred file if not already taken.

Consequently, GDMP heart beat will not only test the functionality of register, publish and replicate but it also utilises the `GDMP_NOTIFICATION_FOR_PUBLISH_CATALOGUE`, `GDMP_STAGE_FROM_MSS` and `GDMP_STAGE_TO_MSS` scripts. Since GDMP heart beat uses its own VO, this will not interfere with other VO's files.

More information on GDMP Heartbeat Monitoring and the required software can be found at:

http://cmsdoc.cern.ch/cms/grid/gdmp_hb

6.10 Network Failures

In principle, each site that has not been reachable for some time is responsible for getting the latest information from other sites. Once a site is up and the network is working properly, the file `gdmp_get_catalogue` can be used to get the latest information from any site.

When a file transfer fails because of a network problem during the data transfer process, this transfer can be resumed afterwards when the network is back again. Thus, the data transfer continues from the latest checkpoint and prevents re-sending the whole file. This is based on the “resume” feature of the nc-ftp client.

6.11 Some Program Restrictions

The server should be run on a machine which is accessible from the outside world.

The Objectivity “bootfilepath” given to the configuration file should be *exactly* the same as specified in the Objectivity's catalogue. Note that alias names for directories cannot be used.

6.12 Server Logfiles

Every time a client command is executed, the client internally contacts the GDMP server that by default logs all communication in the file `GDMP_VAR_DIR/gdmp_server_log.out`. For instance, if `gdmp_ping` is used, the additional log entry looks like follows:

```
====>START LOGGING OUTPUT FOR PROCESS=gdmp_testbed008.cern.ch_30316_1016127475_1
Server Message: a new client has connected [ Thu Mar 14 18:37:55 2002 ]
Message Received=testbed008.cern.ch\3333:testbed008.cern.ch_30282_1016127475_4\servicename\ [ T
```

Message

Send=testbed008.cern.ch_30282_1016127475_4:ack:0::host/testbed008.cern.ch

[Thu Mar 14 18:37:55 2002]

Message Received=testbed008.cern.ch\3333:testbed008.cern.ch_30282_1016127475_8\:authenticate [Thu Mar 14 18:37:55 2002]

Message: CN=/O=Grid/O=CERN/OU=cern.ch/CN=Heinz Stockinger authorized. [Thu Mar 14 18:37:55 2002]

Message Send=testbed008.cern.ch_30282_1016127475_8:ack:0:Authorized [Thu Mar 14 18:37:55 2002]

Message

Received=testbed008.cern.ch\3333:testbed008.cern.ch_30282_1016127475_15\:pingserver\:

testbed008.cern.ch\3333\0\30 [Thu Mar 14 18:37:55 2002]

Message Send=testbed008.cern.ch_30282_1016127475_15:ack:0:GDMP Server is listening [Thu Mar 14 18:37:55 2002]

Message Received=testbed008.cern.ch\3333:testbed008.cern.ch_30282_1016127475_18\:close [Thu Mar 14 18:37:55 2002]

Message Send=testbed008.cern.ch_30282_1016127475_18:ack:0: [Thu Mar 14 18:37:55 2002]

Server Message: client has disconnected [Thu Mar 14 18:37:55 2002]

==>END LOGGING OUTPUT FOR PROCESS=gdmp_testbed008.cern.ch_30316_1016127475_1

For each client command, the server starts and ends logging with `START LOGGING` and `END LOGGING` respectively and temporarily creates a file in `GDMP_TMP_DIR`. In the example above, the temporary log file is `gdmp_testbed008.cern.ch_30316_1016127475_1`. Once the process has finished, the file is appended to the `gdmp_server_log.out` file.

For GDMP client commands `gdmp_replicate_get` and `gdmp_replicate_put`, the output of the methods is stored in the server log file and it needs to be accessed via a call to `gdmp_job_status` (see Section 7.4) in order to read it at the client side.

By default, the server log file for each GDMP client application (or job) is stored in `GDMP_INSTALL_DIR/var/gdmp_server_log.out` but for the following tools the server log can be created in an additional file called `GDMP_TMP_DIR/<logfile_id>`:

```
gdmp_get_catalogue
gdmp_host_subscribe
gdmp_ping
gdmp_publish_catalogue
gdmp_register_local_file gdmp_remove_local_file
gdmp_replicate_get
gdmp_replicate_put
```

Since for each of these commands the three command line options `-L`, `-O` and `-D` are used in the same way, we describe the functionality here.

```
[-L <logfile_id>] GDMP Server will log information in
                   ${GDMP_TMP_DIR}/<logfile_id>
[-O ]             don't save server output in log file
                   ${GDMP_VAR_DIR}/gdmp_server_log.out
[-D ]             don't delete the temp server log file
```

By using the command line option `-L`, the `logfile_id` for the log file can be specified. Note that the file will always be stored in the directory `GDMP_TMP_DIR` or `GDMP_TMP_DIR/<VO_NAME>` if a VO is used. The option `-O` can be used if the current client command shall not be logged in the standard log file `etc/gdmp_server_log.out`.

Conventionally, the GDMP server creates a temporary log file for each job, attaches the temporary file to the `gdmp_server_log.out` and then deletes the temporary file again. If this temporary file should not be deleted, one can use the option `-D`. The clients which uses the `-D` option will get a `<logfile_id>` back from the server which could be use by `gdmp_job_status` to see the log information.

Note that the `logfile_ID` indicates on which host the file is to find ! `gdmp_job_status` can then be sent to the specified host.

7 GDMP Tools

In this section we describe the command line tools for GDMP and give details on the interaction with remote servers and the replica catalogue.

7.1 `gdmp_catalogue_cleanup`

This tool provides a crash recovery functionality in cases when a remote site using a Mass Storage System has crashed. In addition, the import catalogue is cleaned up for all files that have been transferred successfully, i.e. file entries in the import catalogue are removed. The tool deletes all temporary files (files with the extensions `.req` and `.transferred` in the GDMP directory `var`) and recovers from crashes taking place during the remote staging of files. Note that this command is also called automatically after `gdmp_replicate_get` has succeeded transferring files. Refer to Section 6.7 for further information on status information about the file transfer.

```
Usage: gdmp_catalog_cleanup :
[-t <file|objectivity|all>] filetype, default value is 'all'
[-h ] help message
```

The option `-t` is only available if the GDMP version for is compiled with the Objectivity option or the RPMs `gdmp-objy` are used.

For further details on the use of a Mass Storage System with GDMP refer to Section 9.

7.2 `gdmp_get_catalogue`

In case of a network or server failure, a certain host may not be notified when an update is done. A site which is down for some time is responsible for getting the latest information from other sites. This program contacts a certain host, gets the remote file catalogue and creates the corresponding import catalogue locally. A filter parameter can be applied optionally to filter the import catalogue.

```
Usage: gdmp_get_catalogue :
-r <remotehost>      Remote host name
-p <remoteport>     Remote port number
[-t <file|objectivity>] file type, default value is 'file'
[-f <filter> [-N ]] Only add those files to local import catalogue
                    which have/don't(if -N used) have <filter> in their path.
[-S <server_name>]  Server name. Default is read from gdmp.shared.conf file
[-P <server_port>]  Server port number. Default is read from gdmp.shared.conf file
[-V <virtual_org>]  Virtual Organization
[-L <logfile_id>]   GDMP server will log information in ${GDMP_TMP_DIR}/<logfile_id>
[-O ]              Don't save server output in log file ${GDMP_VAR_DIR}/gdmp_server_log.out
[-D ]              Don't delete the temp server log file
[-h ]              Help message
```

The host name has to be the name of a host which appears in the host list. The information about the remote port and directories which are necessary to get the catalogue is taken from the host list in `etc/host_list`. The local site should already have a local file catalogue although it is permitted to be empty.

GDMP creates two different catalogues for files and Objectivity files, respectively. The option `-t` is used to specify which kind of catalogue is required. The option `-t` is only available if the GDMP version for is compiled with the Objectivity option or the RPMs `gdmp-objy` are used.

By default, all files are included into the `import_catalogue` but one can filter several files by using the option `-f`. `<filter>` contains a string of the file path that will be filtered out. If a positive filter is applied, all files that satisfy the filter criterion will be transferred. A negative filter means that files are sieved out and are not included into the `import_catalogue`. The default filter option is “positive filter”.

The options `-S`, `-P` and `-V` are explained in detail in Section 6.2.

The options `-L`, `-O` and `-D` are described in Section 6.12.

Example

Get the file catalogue from `testbed008.cern.ch`:

```
[hst@tbed0079] gdmf_get_catalogue -r testbed008.cern.ch -p3334
Remote catalog file is successfully added. [ Sat Mar 16 17:58:34 2002 ]
```

7.3 `gdmp_host_subscribe`

A host can subscribe to any other host (GDMP server) in the Grid in order to be notified when new files are published at the remote host with `gdmp_publish_catalogue`. See Section 6.5 for further details on subscription.

Usage: `gdmp_host_subscribe` :

```
-r <remotehost>   Remote host name
-p <remoteport>   Remote port number
[-c ]             Check if local host is already subscribed to the remote host
[-u ]             Unsubscribe the local host from remote GDMP server
[-S <server_name>] Server name. Default is read from gdmp.shared.conf file
[-P <server_port>] Server port number. Default is read from gdmp.shared.conf file
[-V <virtual_org>] Virtual Organization
[-L <logfile_id>] GDMP server will log information in ${GDMP_TMP_DIR}/<logfile_id>
[-O ]             Don't save server output in log file ${GDMP_VAR_DIR}/gdmp_server_log.out
[-D ]             Don't delete the temp server log file
[-h ]             Help message
```

The information about the local host is retrieved from the GDMP configuration file (`gdmp.conf`) and is sent to the specified remote host.

The option `-c` can be used in order to check if the local host is already subscribed to a particular remote host given by `-r` and `-p`. One can unsubscribe the local host from a remote with the option `-u`. The file `GDMP_INSTALL_DIR/etc/host_subscribed` contains all hosts to which the local host is subscribed.

The options `-S`, `-P` and `-V` are explained in detail in Section 6.2.

The options `-L`, `-O` and `-D` are described in Section 6.12.

Examples

Subscribe to remote host tbed0079.cern.ch on 3334:

```
[hst@testbed008] gdmf_host_subscribe -r tbed0079.cern.ch -p 3334
Message: Local host subscribed to tbed0079.cern.ch:3334
[ Sat Mar 16 14:14:59 2002 ]
```

One can then check the file host_subscribed to see to which hosts the local host has subscribed:

```
[hst@tbed008] less etc/host_subscribed
tbed0079.cern.ch:3334
```

Check the subscription to the remote host tbed0079.cern.ch :

```
[hst@testbed008] gdmf_host_subscribe -r tbed0079.cern.ch -p 3334 -c
Message: Local host is already subscribed to tbed0079.cern.ch:3334
[ Sat Mar 16 14:16:56 2002 ]
```

Unsubscribe the local host testbed008.cern.ch from the remote host tbed0079.cern.ch:

```
[hst@testbed008] gdmf_host_subscribe -r tbed0079.cern.ch -p 3334 -u
Message: Local host has been unsubscribed from tbed0079.cern.ch:3334
[ Sat Mar 16 14:18:09 2002 ]
```

7.4 gdmf_job_status

A client can get the GDMP server output written into a log file (similar to globus-job-status).

gdmf_job_status can also be used to display the contents of catalogues and files in GDMP_INSTALL_DIR/etc, GDMP_INSTALL_DIR/var and partly of GDMP_INSTALL_DIR/tmp from a remote site that might not have access to the catalogue files stored on the host where the GDMP server is running. See Examples below.

Usage: gdmf_job_status :

```
(-L <logfile_id>|-c <catalogue_name>)
[-r <remotehost>] Remote host name. Default is local host
[-p <remoteport>] Remote port number. Default is local port
[-D ]           Delete the server temp log file ${GDMP_TMP_DIR}/<logfile_id>
[-a ]           Show name of all jobs
                If -D is used, log files for all jobs will be deleted
                and saved in ${GDMP_VAR_DIR}/gdmf_server_log.out
                If -L <logfile_id> is also provided, only show/delete log files
                which have <logfile_id> in their name
[-S <server_name>] Server name. Default is read from gdmf.shared.conf file
[-P <server_port>] Server port number. Default is read from gdmf.shared.conf file
[-V <virtual_org>] Virtual Organization
[-h ]           Help message
```

The option -L can be used to see the GDMP server log out. <logfile_id> is the file ID returned by GDMP clients like gdmf_publish_catalogue, gdmf_ping, gdmf_replicate_get etc. Once you are done with the logfile, you can delete it by using the option -D. Note that all file ID are stored in the directory GDMP_TMP_DIR (see gdmf.conf) at the GDMP server.

The option `-c` can be used to read the contents of files that are managed by the GDMP server and thus would normally only be accessible for users with local accounts on the host where the GDMP server is installed. This option allows to read the following file contents from remote sites. For instance, when the GDMP Client Installation is used. `<catalog name>` can have the following values:

```
gdmp.conf
export_catalogue
host_list
host_subscribed
import_catalogue
local_file_catalogue
```

The server keeps track of some of the ongoing jobs and one can get all the replication jobs currently active with the option `-a`.

The options `-S`, `-P` and `-V` are explained in detail in Section 6.2.

Examples

Get all jobs that the server has logged:

```
[hst@testbed008] gdm_job_status -a
gdmp_testbed008.cern.ch_24509_1016298553_1
gdmp_testbed008.cern.ch_24564_1016298560_1
```

Delete a job:

```
[hst@testbed008] gdm_job_status -L gdmp_testbed008.cern.ch_15329_1016277786_1 -D
Server Message [testbed008.cern.ch:3334]: Out of 1 log file(s) 1 is/are deleted.
Server Message [testbed008.cern.ch:3334]: Deleted job(s) name(s) is/are
"gdmp_testbed008.cern.ch_15329_1016277786_1" [ Sat Mar 16 18:03:13 2002 ]
```

Display the contents of the `local_file_catalogue`:

```
[hst@testbed008] gdm_job_status -c local_file_catalogue
file:testbed008.cern.ch_.home.hst.gdmp-3_.0-15march2002.testfiles.file1-gdmp3_.0:\
file1-gdmp3.0:9:3498111001:1016273020
file:testbed008.cern.ch_.home.hst.gdmp-3_.0-15march2002.testfiles.file2-gdmp3_.0:\
file2-gdmp3.0:9:3498111001:1016286820
file:testbed008.cern.ch_.home.hst.gdmp-3_.0-15march2002.testfiles.file3-gdmp3_.0:\
file3-gdmp3.0:9:3498111001:1016287291
```

The GDMP Client Installation is configured on the Worker Node `lxshare0224` and we want to read the contents of the file `local_file_catalogue` of the GDMP server on `testbed008`:

```
[hst@lxshare0224] gdm_job_status -S testbed008.cern.ch -P 3336 -c local_file_catalogue
file:testbed008.cern.ch_.home.testfiles.cms.cmsfile1_.test:cmsfile1.test:26:4099148029:1017142140
file:testbed008.cern.ch_.home.testfiles.cms.cmsfile2_.test:cmsfile2.test:26:4099148029:1017142150
```

7.5 gdmp_ping

This tool checks if the GDMP server is running on a particular host and port and accepting clients. The remote/local server acknowledges this request with a message. By default, the programs tries to contact the GDMP server on the local host, i.e. on the host where the client command is executed. The host and port information are read from the gdmp.conf file. If the server is listening correctly and the user is authorised, the server responds with a corresponding message (see below).

Usage: `gdmp_ping` :

```
[ -r <remotehost> ] Remote host name. Default is local host
[ -p <remoteport> ] Remote port number. Default is local port
[ -t <timeout> ] Timeout value in seconds
[ -a ] Check if remote/local server is up(no authentication check)
[ -S <server_name> ] Server name. Default is read from gdmp.shared.conf file
[ -P <server_port> ] Server port number. Default is read from gdmp.shared.conf file
[ -V <virtual_org> ] Virtual Organization
[ -L <logfile_id> ] GDMP server will log information in ${GDMP_TMP_DIR}/<logfile_id>
[ -O ] Don't save server output in log file ${GDMP_VAR_DIR}/gdmp_server_log.out
[ -D ] Don't delete the temp server log file
[ -h ] Help message
```

Remote servers can be contacted with the options `-r` on the remote port given by `-p`. The client uses a timeout value of n seconds (default is 10 seconds but can be changed with the timeout option `-t`). Within these n seconds, the server has to send an acknowledgement. If the server does not respond, the client terminates the connection to the server and assumes that the server is currently not ready to accept messages from the client. Thus, this tool should be called before a file transfer is started in order to ensure that the remote site is responding correctly.

By default, `gdmp_ping` checks if a user is authorised to contact a GDMP server. However, in order just to check if a server is running, one can use the option `-a` to use a non-authorized ping. In this case, the user does not have to be registered in the server's grid-mapfile.

The options `-S`, `-P` and `-V` are explained in detail in Section 6.2.

The options `-L`, `-O` and `-D` are described in Section 6.12.

Examples

Test the local server on the local port:

```
[hst@testbed008] gdmp_ping
Message: The local GDMP server testbed008.cern.ch:3334 is listening
and you are an authorized user [ Sat Mar 16 12:32:22 2002 ]
```

Test a remote server without authorisation:

```
[hst@testbed008] gdmp_ping -r tbed0079.cern.ch -p 3334 -a
Message: The remote GDMP server tbed0079.cern.ch:3334
is listening [ Sat Mar 16 12:34:43 2002 ]
```

7.6 gdmp_prepare_open

The Replica Catalogue returns the physical location on disk but not on tape. Thus, if files are staged to tape, a normal user program cannot open the file. `gdmp_prepare_open` checks if a file is on disk and returns success if this is the case. If the file is not on disk, it first stages the file to disk and then returns to the user.

Usage: `gdmp_prepare_open` :

```
-l <lfn>          logical filename
[-r <remotehost>] Remote host name
[-p <remoteport>] Remote port number
[-S <server_name>] Server name. Default is read from gdmp.shared.conf file
[-P <server_port>] Server port number. Default is read from gdmp.shared.conf file
[-V <virtual_org>] Virtual Organization
[-L <logfile_id>] GDMP server will log information in ${GDMP_TMP_DIR}/<logfile_id>
[-O ]            Don't save server output in log file ${GDMP_VAR_DIR}/gdmp_server_log.out
[-D ]            Don't delete the temp server log file
[-h ]            Help message
```

Example

```
[hst@testbed008] gdmp_prepare_open -l testfile1
Prepare to open file at testbed008.cern.ch
The file testfile1 is on disk in the directory /home/flatfiles
```

7.7 gdmp_publish_catalogue

This tool must be used *after* `gdmp_register_local_file`. It creates the export catalogue locally (based on the content of the local file catalogue), sends a copy to the subscribed hosts and registers complete replica information (LFN, PFN, size, time stamp, CRC checksum, file type) into the Replica Catalogue. Consequently, the Replica Catalogue is updated by default.

Usage: `gdmp_publish_catalogue` :

```
[-t <file|objectivity>] file type, default value is 'file'
[-n ]                  Do not update the Replica Catalogue
[-C ]                  Use manager and password from configuration file
                        while accessing Replica Catalogue
[-f <filter> [-N ]]   Only publish those new files to remote hosts
                        which have/don't(if -N used) have <filter> in their path.
                        It will only filter the newly added files and then publish.
[-S <server_name>]    Server name. Default is read from gdmp.shared.conf file
[-P <server_port>]    Server port number. Default is read from gdmp.shared.conf file
[-V <virtual_org>]    Virtual Organization
[-L <logfile_id>]     GDMP server will log information in ${GDMP_TMP_DIR}/<logfile_id>
[-O ]                  Don't save server output in log file ${GDMP_VAR_DIR}/gdmp_server_log.out
[-D ]                  Don't delete the temp server log file
[-h ]                  Help message
```

When publishing files, either files (arbitrary file format) or Objectivity files can be published by using the option `-t`. The option `-t` is only available if the GDMP version is compiled with the Objectivity option or the RPMs `gdmp-objy` are used.

If the option `-n` is used, replica information is not entered into the replica catalogue.

By default, GDMP tries to use GSI for accessing the Globus Replica Catalogue. If the option `-C` is used, GDMP reads the pass word from the file `gdmp.private.conf` where it is stored in clear text. See Section 5.3.1 for details on Globus Replica Catalogue Security issues.

By default, all new files from the `local_file_catalogue` are published but one can filter several files by using the option `-f`. `<filter>` contains a string of the file path that will be filtered out. If a positive filter is applied, all files that satisfy the filter criterion will be transferred. A negative filter means that files are sieved out and will not be published. The default filter option is “positive filter”.

The options `-S`, `-P` and `-V` are explained in detail in Section 6.2.

The options `-L`, `-O` and `-D` are described in Section 6.12.

Examples

We assume that a remote host (`tbed0079.cern.ch`) has subscribed to the local host (`testbed008.cern.ch`). The catalogue is published without updating the Replica Catalogue:

```
[hst@testbed008] gdmp_publish_catalogue -n
Message: Server Message [testbed008.cern.ch:3334]:
Out of 1 host(s) 1 have received the published catalog [ Sat Mar 16 15:33:26 2002 ]
```

The newly published files then appear in the `export_catalogue`:

```
[hst@testbed008] less etc/export_catalogue
file:testbed008.cern.ch_.home.hst.gdmp-3_.0-15march2002.testfiles.file1-gdmp3_.0:file1-gdmp3.0
file:testbed008.cern.ch_.home.hst.gdmp-3_.0-15march2002.testfiles.file2-gdmp3_.0:file2-gdmp3.0
file:testbed008.cern.ch_.home.hst.gdmp-3_.0-15march2002.testfiles.file3-gdmp3_.0:file3-gdmp3.0
```

From a remote client that does not have access to the directory where the GDMP server is running, the contents of the `export_catalogue` can be obtained via `gdmp_job_status -c export_catalogue`. See Section 7.4.

Note: if you do not have the GSI enabled `globus_replica_catalog` libraries installed, you need to use the option `-C` to get the pass word from `gdmp.conf` and we do that for the VO `alice`.

```
[hst@tbed0079] gdmp_publish_catalogue -V alice -C
```

If you do not use the option `-C` in the case above, you might see the following error at the server side:

```
[hst@tbed0079] var/alice/gdmp_server_log.out
Message: Opening logical collection... [ Tue Apr  2 14:08:44 2002 ]
Message: Adding collection attributes [ Tue Apr  2 14:08:44 2002 ]
RepCatalogue Error: Adding files to collection Insufficient access [globus_replica_catalog, globus
[ Tue Apr  2 14:08:44 2002 ]
```

“Insufficient access” means that the pass word for the LDAP server is not correctly supplied.

Important Note: Creation of First Entries in the Replica Catalogue for a given Storage Element

This note is interesting for site admins, too.

Note that when you set up the Replica Catalogue the first time for a given Storage Element, the Replica Catalogue needs to have the correct LDAP objects. In particular, the object “path” needs to correspond to the `GDMP_STORAGE_DIR` of the given VO. See the following details.

Assume, that for the VO alice we have the `GDMP_STORAGE_DIR` is `/home/files/alice` and thus the Replica Catalogue needs to have the following LDAP entries for a given collection “Alice Files”:

```
dn: re=tbed0079.cern.ch, lc=Alice Files, rc=TESTRC, dc=testbed008, dc=cern, dc=ch
objectclass: top
objectclass: GlobusTop
objectclass: GlobusReplicaInfo
uc: tbed0079.cern.ch
path: /home/files/alice
```

You do not need to understand the details but keep in mind that GDMP creates the variable “path” automatically the first time a file is inserted for the given Storage Element `tbed0079.cern.ch`. Thus, you need to make sure that you publish (`gdmp_publish_catalogue`) a file that contains exactly the file path “`/home/files/alice`” and no additional subdirectory! In the worse case, just create a file `/home/files/alice/dummy`, register it and publish the catalogue. Once this is done, you can also create sub directories in your `GDMP_STORAGE_DIR` and publish files that are in the subdirectory.

The catalogue output can then look like follows:

```
dn: re=tbed0079.cern.ch, lc=Alice Files, rc=TESTRC, dc=testbed008, dc=cern, dc=ch
objectclass: top
objectclass: GlobusTop
objectclass: GlobusReplicaInfo
uc: tbed0079.cern.ch
path: /home/files/alice
filename: dummy
filename: alice-subdir1/alicefile1.test
filename: alice-subdir1/alicefile2.test
filename: alice-subdir2/alicefile1.test
```

7.8 `gdmp_register_local_file`

GDMP keeps a local file catalogue for all files that it manages. Thus, every file that needs to be published and later replicated has to be registered in the local file catalogue (`GDMP_INSTALL_DIR/etc/local_file_catalogue`) by means of this command. Note that registering in the local file does *not* imply that the file is registered in the replica catalogue!

Note that normally files need to be located on the SE and can only be registered when they are available in the SE storage space, i.e. under the `GDMP_STORAGE_DIR` directory!

The tool stores the logical file ID, physical filenames, the size, the modification time, the CRC checksum, and the file type in the local file catalogue.

Usage: `gdmp_register_local_file` :

```
(-p <pfm>|-d <directory>) Register a single file with -p
                          Register all files in a directory with -d
```

```

[-t <file|objectivity>] specify the file type, default value is 'file'
[-S <server_name>]      Server name. Default is read from gtmp.shared.conf file
[-P <server_port>]     Server port number. Default is read from gtmp.shared.conf file
[-V <virtual_org>]     Virtual Organization
[-L <logfile_id>]      GDMP server will log information in ${GDMP_TMP_DIR}/<logfile_id>
[-O ]                  Don't save server output in log file ${GDMP_VAR_DIR}/gtmp_server_log.ou
[-D ]                  Don't delete the temp server log file
[-h ]                  Help message

```

One can register a single file (-p) with its entire physical filename (PFN) or all files of a directory (-d). In both cases, the file IDs are assigned automatically based on the PFN and the storage directory. Based on the file type (file or Objectivity files), additional information will be stored in the local catalogue.

The option -t is only available if the GDMP version for is compiled with the Objectivity option or the RPMs gtmp-objy are used.

The options -S, -P and -V are explained in detail in Section 6.2.

The options -L, -O and -D are described in Section 6.12.

After the file is correctly inserted into the local_file_catalogue, the script GDMP_STAGE_TO_MSS is called. If it is set, the file is staged to MSS.

Once the file is registered in the catalogue, it can possibly be staged to a mass storage system before gtmp_publish_catalogue is called. Information in the local file catalogue will be stored in the replica catalogue and then made available to the Grid. Note that this tool has to be used before gtmp_publish_catalogue is initiated!

Examples

Register all files in the directory /home/hst/gtmp-3.0-15march2002/testfiles. In our examples, two files (file1-gtmp3.0 and file2-gtmp3.0) will be registered:

```

[hst@testbed008] gtmp_register_local_file -d /home/hst/gtmp-3.0-15march2002/testfiles
Message: Obtaining file attribs e.g size, timestamp, checksum etc for /home/hst/gtmp-3.0-1
Message: File attribs for /home/hst/gtmp-3.0-15march2002/testfiles/file1-gtmp3.0 obtained.
Message: Obtaining file attribs e.g size, timestamp, checksum etc for /home/hst/gtmp-3.0-1
Message: File attribs for /home/hst/gtmp-3.0-15march2002/testfiles/file2-gtmp3.0 obtained.
Message: Registering file:testbed008.cern.ch_.home.hst.gtmp-3_.0-15march2002.testfiles.\
file1-gtmp3_.0:file1-gtmp3.0:9:3498111001:1016273020 [ Sat Mar 16 14:56:11 2002 ]
Message: Registered file:testbed008.cern.ch_.home.hst.gtmp-3_.0-15march2002.testfiles.file1
Message: Registering file:testbed008.cern.ch_.home.hst.gtmp-3_.0-15march2002.testfiles.file
Message: Registered file:testbed008.cern.ch_.home.hst.gtmp-3_.0-15march2002.testfiles.file2
Message: Out of 2 file(s) 2 are registered. [ Sat Mar 16 14:56:11 2002 ]

```

The two files then appear in the local file catalogue:

```

[hst@testbed008] less etc/local_file_catalogue
file:testbed008.cern.ch_.home.hst.gtmp-3_.0-15march2002.testfiles. \
file1-gtmp3_.0:file1-gtmp3.0:9:3498111001:1016273020
file:testbed008.cern.ch_.home.hst.gtmp-3_.0-15march2002.testfiles. \
file2-gtmp3_.0:file2-gtmp3.0:9:3498111001:1016286820

```

From a remote client that does not have access to the directory where the GDMP server is running, the contents of the local_file_catalogue can be obtained via gtmp_job_status -c local_file_catalogue. See Section 7.4.

Registering a single file with option -p:

```
[hst@testbed008] gdmf_register_local_file -p \  
/home/hst/gdmf-3.0-15march2002/testfiles/file3-gdmf3.0  
[...]  
Message: Out of 1 file(s) 1 are registered. [ Sat Mar 16 15:02:26 2002 ]
```

We now show an example where the GDMP Client Installation is used on a Worker Node (lxshare0224.cern.ch) and the file is registered at the server side.

```
[hst@lxshare0224] gdmf_register_local_file -S testbed008.cern.ch \  
-P 3336 -d /home/hst/gdmf-3.0-25march2002/testfiles  
Server Message [testbed008.cern.ch:3336]: A client has been started to  
register the requested files. [ Wed Mar 27 22:04:10 2002 ]  
Message: Server Log ID=gdmf_testbed008.cern.ch_28929_1017263049_1 [ Wed Mar 27 22:04:10 2002 ]
```

The following example is executed from a Worker Node (lxshare0224.cern.ch) and we show how we can use `gdmf_job_status` to check for possible errors:

```
[hst@lxshare0224] gdmf_register_local_file -S testbed008.cern.ch -P 3336 -d /hallo  
Server Message [testbed008.cern.ch:3336]: A client has been started to register the requested  
Message: Server Log ID=gdmf_testbed008.cern.ch_1033_1017311964_1 [ Thu Mar 28 11:39:24 2002 ]
```

Now we check the status:

```
[hst@lxshare0224] gdmf_register_local_file -S testbed008.cern.ch -P 3336 -d /hallo  
Server Message [testbed008.cern.ch:3336]: A client has been started to register the requested  
Message: Server Log ID=gdmf_testbed008.cern.ch_1033_1017311964_1 [ Thu Mar 28 11:39:24 2002 ]  
[...]  
==>END LOGGING OUTPUT FOR PROCESS=gdmf_testbed008.cern.ch_1033_1017311964_1  
Error: /hallo either does not exist/accessable or not a valid directory  
No Files in the directory /hallo to register [ Thu Mar 28 11:39:24 2002 ]  
==>END LOGGING OUTPUT FOR  
PROCESS=gdmf_testbed008.cern.ch_1033_1017311964_1
```

We see that there was an error at the server side: the directory `/hallo` does not exist and is not a part of `GDMP_STORAGE_DIR`.

7.9 `gdmf_remove_local_file`

Remove a file from disk and from all possible file catalogues: `export_catalogue`, `local_file_catalogue` and optionally from the `Replica Catalogue`.

```
Usage: gdmf_remove_local_file :  
-p <pfname>           Entire physical file path on disk  
[-t <file|objectivity>] specify the file type, default value is 'file'  
[-n ]                Do not update the Replica Catalogue  
[-C ]                Use manager and password from configuration file  
                     while accessing Replica Catalogue  
[-S <server_name>]  Server name. Default is read from gdmf.shared.conf file
```

```

[-P <server_port>]    Server port number. Default is read from gtmp.shared.conf file
[-V <virtual_org>]    Virtual Organization
[-L <logfile_id>]     GDMP server will log information in ${GDMP_TMP_DIR}/<logfile_id>
[-O ]                Don't save server output in log file ${GDMP_VAR_DIR}/gtmp_server_log.out
[-D ]                Don't delete the temp server log file
[-h ]                Help message

```

The option `-t` is only available if the GDMP version is compiled with the Objectivity option or the RPMs `gdmp-objy` are used.

If the option `-n` is used, replica information is not deleted from the replica catalogue.

By default, GDMP tries to use GSI for accessing the Globus Replica Catalogue. If the option `-C` is used, GDMP reads the pass word from the file `gtmp.private.conf` where it is stored in clear text. See Section 5.3.1 for details on Globus Replica Catalogue Security issues.

The options `-S`, `-P` and `-V` are explained in detail in Section 6.2.

The options `-L`, `-O` and `-D` are described in Section 6.12.

Examples

Delete a single file from the local host but keep the entry in the Replica Catalogue:

```

[hst@testbed008] gtmp_remove_local_file -p \
/home/hst/gtmp-3.0-15march2002/testfiles/file1-gtmp3.0 -n
Message: Deleting /home/hst/gtmp-3.0-15march2002/testfiles/file1-gtmp3.0 of type file [
Message: /home/hst/gtmp-3.0-15march2002/testfiles/file1-gtmp3.0 deleted. [ Sat Mar 16 18

```

7.10 gtmp_replicate_get

This is the main executable to transfer files from a remote machine to the local host. The users should make sure that they do not have an older file in the same directory and with the same name where the new file will be transferred to.

It is possible to start multiple `gtmp_replicate_get` clients on the same import catalogue. The system itself takes care of concurrency issues and whether a file is already being transferred by some other client or not. This functionality provides users with an easy way to do parallel transfers and improve the network throughput obtained.

```

[-x <streams>]        Number of parallel streams (default = 8)
[-i <fileid>]         FileID of the file (see import_catalogue)
                       This option is only required for transferring single files
[-r <remotehost>]     Remote host name. Default is local host
[-p <remoteport>]     Remote port number. Default is local port
[-t <file|objectivity>] filetype, default value is 'file'
[-d ]                attach dummy DB instead of original(for Objectivity files only)
[-n ]                Do not update the Replica Catalogue
[-C ]                Use manager and password from configuration file
                       while accessing Replica Catalogue
[-f <filter> [-N ]]   Replicate only files from local import catalogue
                       which have/don't(if -N used) have <filter> in their path.
[-S <server_name>]    Server name. Default is read from gtmp.shared.conf file
[-P <server_port>]    Server port number. Default is read from gtmp.shared.conf file

```

```

[-V <virtual_org>]    Virtual Organization
[-L <logfile_id>]    GDMP server will log information in ${GDMP_TMP_DIR}/<logfile_id>
[-O ]                Don't save server output in log file ${GDMP_VAR_DIR}/gdmf_server_log.out
[-D ]                Don't delete the temp server log file
[-h ]                Help message

```

By default, 8 parallel streams are used in the GridFTP transfer. The number of parallel streams can be changed with the option `-x`.

The default assumption is that files are taken from the import catalogue and transferred to the client machine. This allows a set of files rather than a single file to be replicated. However, if the file_ID of the file at the source site is known, the options `-i` can be used to replicate single files. The file ID can be obtained from the import_catalogue (see Section 6.3).

The options `-r` and `-p` are only used if one wants to select a certain host/port from where files have to be transferred. If one uses these options, `gdmf_relicate_get` will only transfer files which exist on that host/port. GDMP will not replicate files from any other host except the one specified with the option. By default, GDMP just transfers all files from all hosts that appear in the import_catalogue .

The option `-t` is only available if the GDMP version is compiled with the Objectivity option or the RPMs `gdmf-objy` are used. If the file type “objectivity” is chosen, replicated files are attached to a local federation. Since for large files the attach process can be time consuming, the option `-d` does a “dummy attach” (a simple trick where a different file is attached where Objectivity does not check for associations) that speeds up the attachment process.

By default, when a file is replicated successfully to a local site, the file is validated and then registered in the replica catalogue. In particular, the logical and physical filenames as well as size, modification time, CRC checksum and file type are registered in the replica catalogue. However, a site may request not to insert a file into the replica catalogue by using the option `-n`.

By default, GDMP tries to use GSI for accessing the Globus Replica Catalogue. If the option `-C` is used, GDMP reads the pass word from the file `gdmf.private.conf` where it is stored in clear text. See Section 5.3.1 for details on Globus Replica Catalogue Security issues.

By default, all files from the import_catalogue are transferred but one can filter several files by using the option `-f`. `<filter>` contains a string of the file path that will be filtered out. If a positive filter is applied, all files that satisfy the filter criterion will be transferred. A negative filter means that files are sieved out and shall not be transferred. The default filter option is “positive filter”.

The options `-S`, `-P` and `-V` are explained in detail in Section 6.2.

The options `-L`, `-O` and `-D` are described in Section 6.12.

Note that the data production site can store files in several directories. When these files are transferred to the local site, these directories are also created in the local storage root directory specified by `GDMP_STORAGE_DIR` or `GDMP_OBJY_STORAGE_DIR`.

Some internal details

Before `gdmf_replicate_get` starts a data transfer, it checks if the size of the requested file on disk corresponds to the file size in the remote local_file_catalogue. Only if the file sizes correspond, the file gets replicated. Thus, GDMP makes sure that the requested file is fully available at the remote site and not parts of the file only.

The tool also takes care of some error recovery before it starts to transfer files. It checks the status of transferred files and if it finds some `.not_registered`, `.not_notified`, `.not_replicated` or `.not_validated`, it will start the transfer process from this point again. For instance, if `.not_replicated` exists, it will start

transferring it again. Furthermore, if `.not_validated` exists, this means that the file was not correctly transferred last time and it will start the transfer again.

Examples

By default, `gdmp_replica_get` tries to get (transfer) all files that appear in the `import_catalogue`. For instance, the `import_catalogue` contains the following files which need to be transferred from the remote host to the local host:

```
[hst@tbed0079] less etc/import_catalogue
file:testbed008.cern.ch_.home.hst.gdmp-3_.0-15march2002.testfiles.file1-gdmp3_.0:file1-gdmp3.0:/ho
file:testbed008.cern.ch_.home.hst.gdmp-3_.0-15march2002.testfiles.file2-gdmp3_.0:file2-gdmp3.0:/ho
file:testbed008.cern.ch_.home.hst.gdmp-3_.0-15march2002.testfiles.file3-gdmp3_.0:file3-gdmp3.0:/ho
```

From a remote client that does not have access to the directory where the GDMP server is running, the contents of the `export_catalogue` can be obtained via `gdmp_job_status -c import_catalogue`. See Section 7.4.

Now we start the data transfer without registering the files in the Replica Catalogue:

```
[hst@tbed0079 gdmp-install] gdmp_replicate_get -n
Server Message [tbed0079.cern.ch:3334]: A client has been started to
replicate the requested files. [ Sat Mar 16 15:47:37 2002 ]
Message: Server Log ID=gdmp_tbed0079.cern.ch_14875_1016290057_1 [ Sat Mar 16 15:47:37 2002 ]
```

The client delegates the replication process to the server that starts to transfer the requested files and logs the process in a log file. In the above example, the log file ID is `gdmp_tbed0079.cern.ch_14875_1016290057_1` and the command `gdmp_job_status` can be used in order to display the server log file:

```
[hst@tbed0079] gdmp_job_status -L gdmp_tbed0079.cern.ch_14875_1016290057_1
==>START LOGGING OUTPUT FOR PROCESS=gdmp_tbed0079.cern.ch_14875_1016290057_1
Server Message: a new client has connected [ Sat Mar 16 15:47:37 2002 ]
Message Received=tbed0079.cern.ch\:3334:tbed0079.cern.ch_14842_1016290057_4\:servicename\:
Message Send=tbed0079.cern.ch_14842_1016290057_4:ack:0::host@tbed0079.cern.ch [ Sat Mar 16
Message Received=tbed0079.cern.ch\:3334:tbed0079.cern.ch_14842_1016290057_8\:authenticate
[...]
```

In the following example we only want to replicate files that exist on host `shahzad.fnal.gov`. GDMP will try to transfer all files which were published from `shahzad.fnal.gov:2000` only and will not replicate files of any other remote host:

```
[hst@testbed008] gdmp_replicate_get -r shahzad.fnal.gov -p 2000
```

In the following examples we apply a filter and what to get only files that are stored in the directory "directory1". This is indicated with the filter criterion. Note that the directory needs to exist at the remote site !

```
[hst@testbed008] gdmp_replicate_get -f directory1 -P
```

`gdmp_replicate_get` first creates the directory and second transfers all the files into the new directory.

```
[hst@testbed008 testfiles] ls -lR
./directory1:
total 8
-rwxr-xr-x    1 hst      users          10 Mar 26 08:36 dir1file1
-rwxr-xr-x    1 hst      users          10 Mar 26 08:36 dir2file1
```

Replicate a file from a Computing Element (in particular a WorkerNode):

```
[hst@lxshare0224] gdmf_replicate_get -S testbed008.cern.ch -P 3336 -n
Server Message [testbed008.cern.ch:3336]: A client has been started to replicate the requ
Message: Server Log ID=gdmf_testbed008.cern.ch_29085_1017263433_1 [ Wed Mar 27 22:10:33 2
```

In the following example we show how a filter can be applied and the command is executed from a Worker Node on lxshare0224 using the GDMP Client Installation. The filter is applied to replicate all files in the directory named “directory1”:

```
[hst@lxshare0224] gdmf_replicate_get -S testbed008.cern.ch -P 3336 -n -f directory1
Server Message [testbed008.cern.ch:3336]: A client has been started to replicate the requested fil
Message: Server Log ID=gdmf_testbed008.cern.ch_1007_1017311784_1 [ Thu
Mar 28 11:36:24 2002 ]
```

7.11 gdmf_replicate_put

gdmf_replicate_put has the opposite functionality of gdmf_replicate_get and internally even uses this command. One cannot put arbitrary files to a remote site but only the ones that appear in the remote import_catalogue (!) of the host where one wants to put files. It is required that the user issuing this command is also registered at the remote server. This would anyway be the case if the GDMP Client Installation is used locally but for the full installation, special care needs to be taken. In particular, when users issue the GDMP commands directly on the Storage Element.

All the “normal” functionalities of GDMP also work with it, i.e. if a file is not on disk, it will be staged.

Usage: gdmf_replicate_put :

```
[-x <streams>          Number of parallel streams (default = 8)
[-i <fileid>           FileID of the file (see import_catalogue)
                        This option is only required for transferring single files
[-r <remotehost>]     Remote host name where you want to put the file. Default is local host
[-p <remoteport>]     Remote port number where you want to put the
file. Default is local port
[-t <file|objectivity>] filetype, default value is 'file'
[-d ]                 attach dummy DB instead of original(for Objectivity files only)
[-n ]                 Do not update the Replica Catalogue
[-C ]                 Use manager and password from configuration file
                        while accessing Replica Catalogue
[-f <filter> [-N ]]   Replicate only files from local import catalogue
                        which have/don't(if -N used) have <filter> in their path.
[-S <server_name>]   Server name. Default is read from gdmf.shared.conf file
[-P <server_port>]   Server port number. Default is read from gdmf.shared.conf file
[-V <virtual_org>]   Virtual Organization
[-L <logfile_id>]    GDMP server will log information in ${GDMP_TMP_DIR}/<logfile_id>
```

```

[-O ]          Don't save server output in log file ${GDMP_VAR_DIR}/gdmf_server_log.out
[-D ]          Don't delete the temp server log file
[-h ]          Help message

```

By default, 8 parallel streams are used in the GridFTP transfer. The number of parallel streams can be changed with the option `-x`.

The options `-p` and `-r` are equivalent (reverse) to the options of `gdmf_replicate_get`.

The option `-t` is only available if the GDMP version is compiled with the Objectivity option or the RPMs `gdmf-objy` are used. If the file type “objectivity” is chosen, replicated files are attached to a local federation. Since for large files the attach process can be time consuming, the option `-d` does a “dummy attach” (a simple trick where a different file is attached where Objectivity does not check for associations) that speeds up the attachment process.

By default, all files from the *remote* `import_catalogue` are transferred but one can filter several files by using the option `-f`. `<filter>` contains a string of the file path that will be filtered out. If a positive filter is applied, all files that satisfy the filter criterion will be transferred. A negative filter means that files are sieved out and do not want to be transferred. The default filter option is “positive filter”.

By default, GDMP tries to use GSI for accessing the Globus Replica Catalogue. If the option `-C` is used, GDMP reads the pass word from the `gdmf.private.conf` file where it is stored in clear text. See Section 5.3.1 for details on Globus Replica Catalogue Security issues.

The options `-S`, `-P` and `-V` are explained in detail in Section 6.2.

The options `-L`, `-O` and `-D` are described in Section 6.12.

Examples

A file has been creating on `tbed0079` and then published to `testbed008`. Now, we start `gdmf_replicate_put` on `tbed0079` and replicate the file to `testbed008.cern.ch`.

1. We first check the remote import catalogue of `testbed008.cern.ch`. Note that we send `gdmf_job_status` to `testbed008.cern.ch`!

```

[hst@tbed0079] gdmf_job_status -c import_catalogue -r testbed008.cern.ch -p3335
file:tbed0079.cern.ch_.home.hst.gdmf-3-0-20march2002.testfiles.fileA_.tbed0079:\
fileA.tbed0079:/home/hst/gdmf-3-0-20march2002/testfiles:tbed0079.cern.ch:3335:1

```

The `file_ID` of the file to be replicated is:

```
tbed0079.cern.ch_.home.hst.gdmf-3-0-20march2002.testfiles.fileA_.tbed0079
```

2. Now run `gdmf_replicate_put` with the above `file_ID` and state the remote host `testbed008.cern.ch` since this is the host where we want to put the file:

```

[hst@tbed0079] gdmf_replicate_put -i \
tbed0079.cern.ch_.home.hst.gdmf-3-0-20march2002.testfiles.fileA_.tbed0079 \
-r testbed008.cern.ch -p 3335
Server Message [testbed008.cern.ch:3335]: A client has been started to replicate the req
Message: Server Log ID=gdmf_testbed008.cern.ch_28021_1016644398_1 [ Wed Mar 20 18:13:18

```

3. The result of the data transfer can be checked with:

```

[hst@tbed0079] gdmf_job_status -L gdmf_testbed008.cern.ch_21_1016644398_1 \
-r testbed008.cern.ch -p3335
===>START LOGGING OUTPUT FOR PROCESS=gdmf_testbed008.cern.ch_28021_1016644398
Server Message: a new client has connected [ Wed Mar 20 18:13:18 2002 ]
Message Received=tbed0079.cern.ch\:3335:tbed0079.cern.ch_20065_1016644398_4\:vicename\:[
[...]
Message: Total files requested for replicate=1 [ Wed Mar 20 18:13:20 2002 ]
Message: Total files successfully replicated=1 [ Wed Mar 20 18:13:20 2002 ]
Message: Total files requested for staging=0 [ Wed Mar 20 18:13:20 2002 ]
Message: Total files passed pre-processing stage=0 [ Wed Mar 20 18:13:20 2002
Message: Total files skipped due to other process=0 [ Wed Mar 20 18:13:20 200
Message: Total files failed=0 [ Wed Mar 20 18:13:20 2002 ]
Message: Running /home/hst/gdmf-3.0-20march2002/gdmf-install/bin/gdmf_catalogcleanup
Message: Cleaning Catalogue [ Wed Mar 20 18:13:20 2002 ]
Working on
tbed0079.cern.ch_.home.hst.gdmf-3-0-20march2002.testfiles.fileA_.t0079
[ Wed M
Message: 1 entire(s) removed for tbed0079.cern.ch_.home.hst.gdmf-3-0-arch2002.testfiles.
File Remved:/home/hst/gdmf-3.0-20march2002/gdmf-install/var//tbed0079rn.ch_.home.hst.gdm
===>END LOGGING OUTPUT FOR PROCESS=gdmf_testbed008.cern.ch_28021_1016644398_1

```

4. One can also check the remote local_file_catalogue on testbed008.cern.ch and it shows that the file has been correctly registered at testbed008:

```

[hst@tbed0079] gdmf_job_status -c local_file_catalogue -r testbed008.cern.ch -p3335
file:tbed0079.cern.ch_.home.hst.gdmf-3-0-20march2002.testfiles.fileA_.tbed0079:fileA.tbed0079

```

7.12 gdmf_server

A server has to be started on each site participating in the Data Grid. The server is responsible for answering client messages, sending notification messages, and handling security issues. The server is started by the Internet daemon (inetd) or as a stand-alone server (see Section 4.2.2 for details).

gdmf_server

The server does not accept any command line options since all server configuration is stored and managed in the GDMP configuration files.

7.13 gdmf_stage_complete

This command line tool is used only by the script GDMP_STAGE_FROM_MSS in order to automatically initiate a file transfer when a file has been staged from tape to disk. For further information refer to Section 9.

```

Usage: gdmf_stage_complete :
-f filename
[-h] for help message

```

7.14 gdmp_stage_from_mss

This command line tool calls explicitly the staging scripts on the GDMP server side. The file will be copied to the VO specific GDMP_STORAGE_DIR on the disk of the SE.

Usage: gdmp_stage_from_mss :

```
-l <lfn>          logical filename
[-r <remotehost>] Remote host name
[-p <remoteport>] Remote port number
[-S <server_name>] Server name. Default is read from gdmp.shared.conf file
[-P <server_port>] Server port number. Default is read from gdmp.shared.conf file
[-V <virtual_org>] Virtual Organization
[-L <logfile_id>] GDMP server will log information in ${GDMP_TMP_DIR}/<logfile_id>
[-O ]            Don't save server output in log file ${GDMP_VAR_DIR}/gdmp_server_log.out
[-D ]            Don't delete the temp server log file
[-h ]            Help message
```

The LFN needs to be passed to the script and the GDMP server then calls GDMP_STAGE_FROM_MSS and copies the file to the disk location.

Example

The file testfile1 shall be staged to the disk on testbed008:

```
[hst@testbed008] gdmp_stage_from_mss -l testfile1
```

```
Start staging at testbed008.cern.ch
```

```
The file testfile1 has been succeeded staged to MSS to /home/flatfiles at testbed008.cern.ch
```

7.15 gdmp_stage_to_mss

This is the counter part to gdmp_stage_from_mss and thus copies a file from disk (VO specific GDMP_STORAGE_DIR) to the tape location on the SE.

Usage: gdmp_stage_to_mss :

```
-l <lfn>          logical filename
[-r <remotehost>] Remote host name
[-p <remoteport>] Remote port number
[-S <server_name>] Server name. Default is read from gdmp.shared.conf file
[-P <server_port>] Server port number. Default is read from gdmp.shared.conf file
[-V <virtual_org>] Virtual Organization
[-L <logfile_id>] GDMP server will log information in ${GDMP_TMP_DIR}/<logfile_id>
[-O ]            Don't save server output in log file ${GDMP_VAR_DIR}/gdmp_server_log.out
[-D ]            Don't delete the temp server log file
[-h ]            Help message
```

Example

File testfile1 is located in GDMP_STORAGE_DIR on the SE at testbed008 and is then staged to the tape location by GDMP_STAGE_TO_MSS.

```
[hst@testbed008] gdmp_stage_to_mss -l testfile1
Start staging at testbed008.cern.ch
The file testfile1 has been succeeded staged to MSS at testbed008.cern.ch
```

The GDMP server logs all the staging information in the log file `GDMP_INSTALL_DIR/var/stage_requests` and thus allows for a convenient logging of user access. The log file has the following format:

```
[ Thu Oct  3 18:25:48 2002 ] -- StageToMss Request:
  from   : /O=Grid/O=CERN/OU=cern.ch/CN=Heinz Stockinger
  for LFN: XYYYY
```

7.16 get_progress_report

The executable `gdmp_replicate_get` uses the `.stat` files in the GDMP directory `var` directory for each file currently being transferred. These files contain progress information on the transfer and are updated every few seconds. Once the file has been completely transferred, these `.stat` files are deleted and the final transfer log goes in the file `GDMP_INSTALL_DIR/var/replicate.log`. If you want to find out the state of advancement of all the transfers currently in progress, you can run this script, and it will produce a file `GDMP_INSTALL_DIR/var/progress.log` which will contain the latest progress information.

A typical progress report (stored in the file `progress.log`) looks like follows:

```
filename - total size - bytes transferred - %age completed
/data/file1 - 100000 - 50000 - 50%
```

A typical entry in the file `replicate.log` looks like follows:

```
/data/file1 incoming from host: host1.cern.ch, bytes transferred 100000,
start: [ Wed Oct 17 21:44:52 2001 ], end: [ Wed Oct 17 21:47:12 2001 ]
```

It contains the file to be transferred (the local file path), the start time and the end time of the data transfer for the specific file.

Note that this tool is only available in the full GDMP installation and can only be executed on the machine where the GDMP server is installed.

7.17 gdmp_version

This tool is location in the directory `sbin` and prints the version number of the current release.

7.18 create_gridmapfile

The script is required in the EU DataGrid Testbed in order to automatically create the necessary entries in the VO specific GDMP grid-mapfiles on the SE. It takes the following arguments:

```
create_gridmapfile <GDMP_INSTALL_DIR> <GDMP_USER> <GDMP_VO>
```

`GDMP_USER` is the Unix user under which the GDMP server is running and `GDMP_VO` is the specific VO. Before invoking the script, the user needs to get a valid proxy using the command `grid-proxy-init`.

The script copies the subjects of the SE hosts certificated into the `gdmp VO` specific grid-mapfile. This is done for the specific VO on the SE and consequently the script needs to be called for each VO separately. Thus, the grid-mapfile will then have entries like follows:

```
"/O=Grid/O=CERN/OU=cern.ch/CN=host/lxshare0219.cern.ch" gdmf
"/O=Grid/O=CERN/OU=cern.ch/CN=host/lxshare0222.cern.ch" gdmf
```

In addition, the script contacts the grid-mapfile in order to find out about all possible users and their VOs and then enters the user subject names into the VO specific GDMP-grid-mapfiles.

In addition the scripts searches in the local grid-mapfiles for all users with a certificate mapped to the specific local VO user. It then adds their certificate subjects to the VO specific GDMP grid-mapfile. For instance, the CMS specific grid-mapfile in the /opt/edg/etc/cms/grid-mapfile can look like follows. Note that no mapping to local users is required!

```
"/C=FR/O=CNRS/OU=LPNHE/CN=Firstname Lastname/Email=First.Last@poly.in2p3.fr"
"/C=IT/O=INFN/OU=Personal Certificate/L=Pisa/CN=First Name/Email=Firstname.Lastname@pi.infn.it"
```

7.19 Other tools in sbin

The GDMP directory `sbin` contains some shell utilities that GDMP uses internally. One of the internal tools is the following:

gdmp_server_start: This script is called by the `inetd` to start the GDMP server. See Appendix A Section 12 for installation details.

Further scripts like for staging and notification can be added here.

8 GDMP C++ API

For GDMP client commands, there exists a minimal GDMP C++ API for accessing GDMP servers remotely. The C++ GDMP_API class can be found in the GDMP installation tree under `include/API/gdmp_api.h`.

8.1 Description of the Programming Interface

We point out the main features of the API but refer to the file `gdmp_api.h` and the Section 7 for more details on arguments. We state here the public methods of the GDMP_API class:

The corresponding method for `gdmp_ping` is:

```
GDMP_Result ping(const string &host = "",
                unsigned int port = 0);
```

The following methods corresponds to `gdmp_register_local_file` with option `-S`. In more detail `is_dir = true` corresponds to the option `-d` and `is_dir = false` corresponds to the option `-p`.

```
GDMP_Result register_local_file(const string &path,
                                bool is_dir=false
                                #ifdef OBJECTIVITY
                                ,const string &filetype = ""
                                #endif);
```

The method for `gdmp_publish_catalogue` looks like follows. For instructions on using filters see the last paragraphs in this section.

```
GDMP_Result publish_catalogue(const string &filter = "",
                               bool update_rc=true,
                               bool gsi_auth=true
                               #ifdef OBJECTIVITY
                               ,const string &filetype = ""
                               #endif);
```

The method for `gdmp_remove_local_file` looks like follows:

```
GDMP_Result remove_local_file(const string &path,
                               bool update_rc=true,
                               bool gsi_auth=true
                               #ifdef OBJECTIVITY
                               ,const string &filetype = ""
                               #endif);
```

The method for `gdmp_replicate_get` looks like follows. For instructions on using filters see the last paragraphs in this section.

```
GDMP_Result replicate_get(const string &host = "",
                          unsigned int port = 0,
                          const string &filter = "",
                          const string &catalog_path = "",
```

```

const string &fileid = "",
bool update_rc=true,
bool gsi_auth=true
#ifdef OBJECTIVITY
, const string &filetype = ""
, bool dummy_attach=false
#endif);

```

The method for `gdmf_replicate_put` looks like follows. For instructions on using filters see the last paragraphs in this section.

```

GDMP_Result replicate_put(const string &host = "",
    unsigned int port = 0,
    const string &filter = "",
    const string &catalog_path = "",
    const string &fileid = "",
    bool update_rc=true, bool gsi_auth=true
#ifdef OBJECTIVITY
, const string &filetype = ""
, bool dummy_attach=false
#endif);

```

The following method corresponds to `gdmf_host_subscribe` without the option `-c`:

```

GDMP_Result host_subscribe(const string &host,
    unsigned int port);

```

The following method corresponds to `gdmf_host_subscribe` with the option `-u` but without `-c`.

```

GDMP_Result host_unsubscribe(const string &host,
    unsigned int port);

```

The following method corresponds to `gdmf_get_catalogue`. For instructions on using filters see the last paragraphs in this section.

```

GDMP_Result get_catalogue(const string &host,
    unsigned int port,
    const string &filter = ""
#ifdef OBJECTIVITY
, const string &filetype = ""
#endif);

```

The following method can be used to handle/retrieve log on local server. Set to `true` if you want to store server log in separate file. This also allows to retrieve it later with `get_log`.

```

void set_separate_log(bool = true);

```

Set to `true` if one wants to store server side messages in log:

```

void set_use_log(bool = true);

```

Retrieve server side log (only if `separate_log` is set):

```
GDMP_Result get_log(void);
```

Clean log on server side:

```
GDMP_Result remove_log(void);
```

Set a Virtual Organisation (VO):

```
void set_vo(const string &vo = "");  
void set_local_vo(const string &vo);
```

Using Filters in the API

By default, if a filter is used with the variable `filter` in the API, a positive filter is used. In order to distinguish between a positive and a negative filter, the sign `+` or `-` has to be used as the first character in the filter string. In more detail, the strings `"<value>"` or `"<value>"` will be treated as a positive filter and `"-<value>"` will be treated as negative filter.

For example, if one wants to apply a negative filter where `<value>=muon` then he/she should pass `-muon` to method in the `filter` argument. If `<value>=+myfiles`, the proper positive filter string should be `++myfiles` otherwise GDMP will assume that the first `+` is used to indicate that this is positive filter and GDMP will only search for `myfiles`.

8.2 Usage Instructions

In the GDMP CVS repository

```
http://cdcvs.fnal.gov/cgi-bin/public-cvs/cvsweb-public.cgi/gdmp/test
```

you can find a test program `api_test.C` and the corresponding makefile `api_test_build` for the client API. In order to use the client API, you only need to include the header file `gdmp_api.h`. You need to link your program with `libgdmp` and `libgdmp_client` libraries as well as `libReplicaCatalogue` and a few Globus libraries as you can see in this example makefile:

```
GLOBUS_FLAVOR=gcc32dbg  
g++ api_test.C -g -o api_test -I$GDMP_INSTALL_DIR/include/gdmp -I../API \  
-L$GDMP_INSTALL_DIR/lib -L$GLOBUS_LOCATION/lib \  
-lgdmp -lgdmp_client -lReplicaCatalog \  
-lglobus_replica_catalog_${GLOBUS_FLAVOR} \  
-lglobus_gass_copy_${GLOBUS_FLAVOR} \  
-lglobus_nexus_${GLOBUS_FLAVOR} \  
-lglobus_common_${GLOBUS_FLAVOR} \  
-lpthread
```

If you are using shared libraries, you need to make sure that your `LD_LIBRARY_PATH` contains the location of all the libraries above.

9 Support for a Mass Storage System

9.1 Motivation

GDMP supports a simple interface for a Mass Storage System (MSS). Since we assume that a site may run out of disk space, this interface is responsible for staging files from disk to the MSS when the disk-pool is full and staging them back to disk when requested.

9.2 Flow of Control

Let us assume a case where a file is in the local file catalogue but because of storage-space restrictions it has been staged to the MSS. Now this file is requested by a remote site via `gdmp_replicate_get`, and the file cannot be found in the directory `GDMP_STORAGE_DIR` (or `GDMP_OBJY_STORAGE_DIR` in case of Objectivity files). GDMP always looks into this directory first. If the file is not found there, the remote client will send a request to stage the file. The local GDMP server will receive this request and will start a staging script which is pointed to by `GDMP_STAGE_FROM_MSS`. If the script is started successfully, the local server will send a message back to the remote client saying that staging is now in process. The remote client will disconnect and move on to the next files in its import catalogue. When the file staging is complete, the script will call `gdmp_stage_complete` which will notify the remote server that the file has been staged and is ready to transfer. The remote server will start a new client using `gdmp_replicate_get` to transfer this file.

9.3 The Interface

The staging script itself is not provided by GDMP. We only provide a plug-in mechanism for staging scripts (to and from the MSS) since different sites may have different Mass Storage Systems and thus require specific procedures. The staging script should have the following command line interface:

```
GDMP_STAGE_FROM_MSS <relative_path_name> <root_directory_on_disk>
```

where `relative_path_name` is a file path on disk and `root_directory_on_disk` is the storage root directory on disk (either for files or Objectivity files).

For instance, a possible script could be invoked as follows:

```
GDMP_STAGE_FROM_MSS dir1/dir2/file1 /data/directory
```

where the second argument would be mapped to the following physical file path on disk:

```
/data/directory/dir1/dir2/file1
```

It is required that the MSS itself has its internal catalogue about all files listed. Thus, it needs to map this file information here to its internal file location.

It is required that the staging process is atomic. GDMP partly takes care of this internally. GDMP has an internal method that checks the remote file size on disk and compares it with the one registered in the local file catalogue. Only if both file sizes are the same, GDMP starts the data transfer process. Thus, GDMP never transfers files that are partially staged to disk.

When the staging is completed, the script is expected to call the tool `gdmp_stage_complete` which would notify the client that the file is ready to be transferred.

The second staging script for transfers from disk to MSS should have the following interface:

```
GDMP_STAGE_TO_MSS <relative_path_name> <root_directory_on_disk>
```

The script should also have knowledge about the directory inside the MSS where the file should be copied to.

9.4 Staging States

We define the following states for processing of files that reside on tape at the remote site. Let us consider an example of a staging operation. A client requests the file `filename1` from a remote site with `gdmf_replicate_get`. Locally, this request is logged and the file `filename1.stat` is created in the GDMP directory `var`. If the file is on the remote disk, the local client carries on with the transfer, and this status file contains the progress of the transfer. However, if the file is not present on the remote disk, the local client sends a request to stage the file at the remote end and produces a file `filename1.req` in `GDMP_INSTALL_DIR/var`. The remote server calls the staging script. The script starts the executable `gdmf_stage_complete` when the staging has completed. This notifies the server on the requesting site that staging has been completed on the remote end. The server then starts the `gdmf_replicate_get` which starts the file transfer of `filename1` and creates the file `filename1.stat` indicating the transfer status.

Note the `GDMP_STAGE_TO_MSS` is called in `gdmf_register_local_file`.

9.5 Interface to HRM

GDMP has a plug-in for the Hierarchical Storage Manager (HRM) [1] APIs, which provide a common interface to be used to access different Mass Storage Systems. The implementation of HRM is based on CORBA communication mechanisms. Some initial integration tests have been performed, with promising results.

The C++ plug-in and the CORBA IDL exist in the source directories under: `StagingPlugins` and `HRMIDL`. We do not provide a production version of this now and thus the code is not included into the installation tree. For details on the IDL refer to:

http://cdcvs.fnal.gov/cgi-bin/public-cvs/cvsweb-public.cgi/ppdg_idl/

10 Replica Catalogue C++ API and Command Line Tools

10.1 Description of the Programming Interface

The Replica Catalogue API is a generic C++ API to the Globus replica catalogue. It implements most of the methods for the Replica Catalog as stated in the Data Management Architecture document, “Section Replica Catalog” [9]. For background on replica catalogues, logical and physical filenames please refer to this [9].

Note: for a C++ API to RLS, please refer to the edg-replica-manager [5].

The C++ replica catalogue class (to be found in the GDMP installation tree under `include/ReplicaCatalog/ReplicaCatalog.h`) is defined as follows and allows for insertion, deletion and search of replica information:

```
class ReplicaCatalog {
public:
    ReplicaCatalog(      RC_Url      contact_string,
                       string      manager_dn,
                       string      manager_pw,
                       string      collection_url);
    ~ReplicaCatalog(void);

    vector<string> getPhysicalFileNames(string lfn);
    string getLogicalFileName(string pfn);
    RC_Result addLogicalFileName(string lfn);
    RC_Result addPhysicalFileName(string lfn, string pfn);
    RC_Result deleteLogicalFileName(string lfn);
    RC_Result deletePhysicalFileName(string lfn, string pfn);
    RC_Result addLogicalFileAttribute(string lfn, string attrnam, string attrval);
    RC_Result deleteLogicalFileAttribute(string lfn, string attrnam, string attrval)
    deque<GDMP_AttrVal_Pair> getLogicalFileAttributes(string lfn);

private:
    RC_Url      rc_url_;
    string      collection_url_;
    GDMP_Rep_Catalogue* rep_catalog_;
};
```

Note that the variables in the constructor need LDAP specific information and need to be checked with the replica catalogue administrator.

10.2 Usage Instructions

The replica catalogue API does not depend on GDMP and can be compiled and linked without the GDMP source code. In order to use the API in an application, the ReplicaCatalogue library as well as a few Globus and LDAP libraries need to be linked to the application (see example makefile below). Note that the ReplicaCatalog library is available as a static and shared library. If the shared library is used, then the directory `lib` of the GDMP installation tree needs to be added to the `LD_LIBRARY_PATH`.

A test program for the replica catalogue can be found in the directory `test` in the GDMP source code tree which is also available at the CVS repository at the address below. Please make sure that `gdmp install` is called before `rc_addtest` is executed. If not, you need to include “`lib/.libs`” to the library path.

<http://cdcvs.fnal.gov/cgi-bin/public-cvs/cvsweb-public.cgi/gdmp/>

A possible makefile looks like follows:

```
g++ rc_addtest.C -g -o rc_addtest -I$GLOBUS_LOCATION/include \  
-I../ReplicaCatalogue -I$GLOBUS_LOCATION/include/gcc32dbgpthr \  
-L../lib/.libs -L$GLOBUS_LOCATION/lib \  
-lReplicaCatalog -lglobus_replica_catalog_gcc32dbgpthr \  
-lglobus_common_gcc32dbgpthr -lldap_gcc32dbgpthr -llber_gcc32dbgpthr
```

10.3 Usage of the Replica Catalogue Command Line Tools

For the replica catalogue API we also have the equivalent command line tools in the bin directory of the GDMP installation tree:

```
edg-rc  
edg_rc_addLogicalFileAttribute  
edg_rc_addLogicalFileName  
edg_rc_addPhysicalFileName  
edg_rc_deleteLogicalFileAttribute  
edg_rc_deleteLogicalFileName  
edg_rc_deletePhysicalFileName  
edg_rc_getLogicalFileAttributes  
edg_rc_getLogicalFileName  
edg_rc_getPhysicalFileNames
```

`edg-rc` is the main executable but cannot be used. All other programs have symbolic links to the main executable and must be used instead with the correct command-line arguments.

Since the client applications need information about the replica catalogue, a Replica Catalogue configuration file `rc.conf` with the following entries needs to be available:

```
RC_REP_CAT_MANAGER_DN = cn=RCManager, dc=host2, dc=cern, dc=ch  
RC_REP_CAT_MANAGER_PWD = secret  
RC_REP_CAT_URL=ldap://host2.cern.ch:2010/rc=TESTRC, dc=host2, dc=cern, dc=ch  
RC_LOGICAL_COLLECTION = ldap://host2.cern.ch:2010/lc=file collection, \  
rc=TESTRC, dc=host2, dc=cern, dc=ch
```

The client commands line tools have exactly the same attributes as the methods specified in the C++ interface. For instance:

```
edg_rc_addLogicalFileAttribute -l <logical file name> -n <attribute name>  
-v <attribute value> [-c <config file name>] [-d] [-C]
```

where `-l` corresponds to the logical filename, `-n` is the attribute name, `-v` the attribute value.

One can optionally specify the `rc.conf` file for each of the client commands with the option `-c`. Alternatively, one can set the environment variable `RC_CONFIG_FILE` which needs to point to the location of the configuration file.

The option `-d` is used to suppress some output.

By default, if the patched Globus Replica Catalogue libraries are used, the command-line tools try to use GSI authentication for access to the Replica Catalogue server. In this case, the pass word entry in

rc.conf can be empty. With the option `-C` one can pass the pass word in clear text to the Replica Catalogue server and thus it needs to be read from the configuration file rc.conf.

In summary, the command line arguments are like follows:

- l: logical filename
- p: physical filename
- n: attribute name
- v: attribute value
- c: config filename
- d: don't show all output
- C: clear text pass word

Example

Add a logical file called filename1 and the corresponding physical file to the replica catalogue:

```
[hst@tbed0079] edg_rc_addLogicalFileName -l filename1 \  
-p tbed0079.cern.ch/home/edg-replica-manager/testfiles/filename1 -C  
configuration file:      /home/edg-replica-manager/Heinz/rc.conf.new  
logical file name:      filename1  
physical file name:      tbed0079.cern.ch/home/edg-replica-manager/testfiles/filename1  
Message: Opening logical collection... [ Fri Mar 29 21:22:20 2002 ]  
Message: Adding collection attributes [ Fri Mar 29 21:22:20 2002 ]  
Message: Adding logical file attributes [ Fri Mar 29 21:22:20 2002 ]  
Message: Adding location attributes [ Fri Mar 29 21:22:21 2002 ]  
Message: Closing collection [ Fri Mar 29 21:22:21 2002 ]  
OK
```

11 BrokerInfo API

11.1 Description of the Programming Interface

The BrokerInfo C++ API provides the user with an interface to job information that comes from the Job Scheduler. The Scheduler selects a Computing Element to send a job for execution and writes info about the selected resources in the BrokerInfo file which is shipped together with the job. Reading the BrokerInfo file through this interface, the application may retrieve info on the Computing Element where the job is running, the close Storage Elements, etc. The API provides also an implementation for methods that will be provided by the WP2 Replica Manager interface in the future.

A more detailed description of the BrokerInfo library and its functionality can be found in the BrokerInfo document provided by WP1 [10].

The C++ BrokerInfo and ReplicaCatalogB classes (to be found in the GDMP installation tree under `include/BrokerInfo/BrokerInfoB.h` and `include/BrokerInfo/ReplicaCatalogB.h`) are defined as follows:

```
class BrokerInfoEx {
};

class BrokerInfo {
public:

    ~BrokerInfo(void);
    static BrokerInfo* instance(void);
    BI_Result getCE(string& CE) const;
    BI_Result getDataAccessProtocol(vector<string>& DAPs) const;
    BI_Result getInputPFNs(vector<string>& PFNs) const;
    BI_Result getLFN2PFN(string LFN, vector<string>& PFNs) const;
    BI_Result getSEs(vector<string>& SEs) const;
    BI_Result getSEProto(string SE, vector<string>& SEProtos) const;
    BI_Result getSEPort(string SE, string SEProtocol, string& SEPort) const;
    BI_Result getCloseSEs(vector<string>& SEs) const;
    BI_Result getSEMountPoint(string CloseSE, string& SEMount) const;
    BI_Result getPFNs(vector<string>& PFNs) const;

private:

    BrokerInfo(void);
    BI_Result vSearch(const char* searchstr, vector<string>& retvect) const;
    BI_Result sSearch(const char* searcharg, const string searchstr,
        int& position) const;
    BI_Result svIndexBuild(const char* sarg, const string sstr,
        const string varg, vector<string>& retvect) const;
    void vBuild(const string buildstr, vector<string>& retvect) const;

    string BrokerInfoFile_;
    ifstream fbrokerinfo_;
    strstream mbrokerinfo_;
    ClassAd* ad_;
```

```

    static BrokerInfo* instance_;
}

class ReplicaCatalogBEx {
};

class ReplicaCatalogB {
public:

    ~ReplicaCatalogB(void);
    ReplicaCatalogB(void);
    vector<string> ReplicaCatalogB::getPhysicalFileName(string LFN);
    string ReplicaCatalogB::getBestPhysicalFileName(vector<string> PFN,
vector<string> Protocols);
    string ReplicaCatalogB::getTransportFileName(string PFN, string Protocol);
    string ReplicaCatalogB::getPosixFileName(string TFN);
    BI_Result ReplicaCatalogB::getSelectedFile(string LFN, string Protocol,
string TFN, string FileName);

private:

    BrokerInfo *brokerinfo_;

};

};

```

11.2 Usage Instructions

The BrokerInfo API does not depend on GDMP and can be compiled and linked without the GDMP source code. In order to use the API in an application, BrokerInfo library as well as the Condor ClassAd library need to be linked to the application (see example makefile below). Note that the BrokerInfo library is available as a shared and static library. If you use the shared library, the directory `lib` of the GDMP installation tree needs to be added to the `LD_LIBRARY_PATH`.

A test program for the BrokerInfo can be found in the directory `test` in the GDMP source code tree which is also available at the CVS repository at:

<http://cdcvs.fnal.gov/cgi-bin/public-cvs/cvsweb-public.cgi/gdmp/>

A possible makefile looks like follows:

```

g++ BrokerTester.C -g -o BrokerTester \
-I../BrokerInfo \
-I/usr/local/grid/ClassAd/include \
-L../lib/.libs \
-L/usr/local/grid/ClassAd/lib \
-lBrokerInfo -lclassads

```

12 Appendix A: Configuring GDMP with inetd

This appendix gives a short introduction to inetd and explains which system files are edited locally doing the GDMP installation process.

12.1 Inetd Background

Simply put, the inetd provides Internet service management for a networked computer. It listens on certain ports and calls other servers or daemons to service request. As regards GDMP, we register a certain port, e.g. port 2000, with the inetd daemon and when a GDMP client connects to the machine via a socket connection, the inetd daemon takes the request on port 2000, starts the GDMP server via the script `GDMP_INSTALL_DIR/sbin/gdmp_server_start` and passes all the socket information to the GDMP server. The GDMP server in turn then handles the client request.

In more detail, the inetd daemon starts by default each time the system is started. When the daemon starts, it reads its configuration information from the configuration files `/etc/services` and `/etc/inetd.conf`. GDMP is thus regarded as a service that is started by inetd.

12.2 Configuration Steps done by GDMP Installation Program

All the following steps are done automatically but we state them in some details for possible problem detection. In detail, the installation program inserts information into the files `/etc/services` and `/etc/inetd.conf` to set up a service.

The following steps are done by root:

1. Edit `inetd.conf` to add in one line:

```
gdmp-server stream tcp nowait "username"  
"GDMP_INSTALL_DIR"/sbin/gdmp_server_start
```

The terms in “ “ have to be replaced by the respective fields and “username” needs to be the userID of the user running the GDMP server.

2. Edit the `/etc/services` file to add

```
gdmp-server "port"/tcp
```

again “port” is the port of your choice; we use 2000 mostly.

3. Send the HUP signal to the inetd server: Linux users can get the inetd process id (pid) from `/var/run/inetd.pid`. On Solaris one can get the pid by doing `ps -e | grep inetd`.

To send the HUP signal to inetd you need to do:

```
kill -HUP <inetd_pid>
```

In principle, `gdmp_server_start` contains the environment variables `PATH`, `LD_LIBRARY_PATH` plus additional variables for GDMP, Objectivity and Orbacus.

Since inetd does not see the `PATH` and `LD_LIBRARY_PATH`, you can set them here but it depends on the system. Just do an `echo PATH` and `echo LD_LIBRARY_PATH` and add whatever you get in this file. Make sure you do not have any “echo”s in this file for debugging or whatever as the stdout is connected to the socket when the server is started with inetd.

12.3 Example Configuration for inetd

An example for a `/etc/inetd.conf` file entry is given here. GDMP is assigned the service name `gdmp-server` and the `gdmp_server_start` script will be called when `inetd` receives a request on the port 2000. We assume now that is the is default GDMP port. The port itself is configured in the file `/etc/services` (see below). As for all examples in this User Guide we assume that the GDMP installation directory is `/usr/local/grid/gdmp`. Note that the entire sequence has to be added in one single line and the server is started as user “username”:

```
gdmp-server stream tcp nowait username \  
/usr/local/grid/gdmp/sbin/gdmp_server_start
```

In the file `/etc/services` the GDMP server port is assigned:

```
gdmp-server      2000/tcp      # GDMP 3.0
```

13 Appendix B: Usage of Grid Security Infrastructure

This is a short introduction into the usage of Grid security in Globus and should be sufficient for using GDMP. It is intended for people new to Globus. For exact details, please refer to the Security section at the Globus web page: <http://www.globus.org/Security/>

GDMP assumes that Globus is installed at the local machine and requires the authentication and authorisation method of Globus. In particular, the Globus command `grid-proxy-init`. Please check if this command is available on your system.

In order to use Globus, one needs to have a special certificate and a key which are required for the authentication procedure. For certificate requests and further details on certificates refer to the Testbed and Integration web page of DataGrid at: <http://marianne.in2p3.fr/> and check out the side bar “Certification Authorities”.

We now assume that you have requested such a certificate and the key. The files `usercert.pem` and `userkey.pem` by default are stored in:

```
user_home_directory/.globus
```

Please make sure that the two files have the correct access permissions:

```
-r----- 1 username      group           963 Mar  8 2001 userkey.pem
-r--r--r-- 1 username      group          3873 Mar  8 2001 usercert.pem
```

Once all the files are correctly in place, a “proxy” can be gained via `grid-proxy-init` which provides a single log on to several machines where you are registered as a Grid user. Only when the proxy is valid (it expires after some time) GDMP tools can be used. The lifetime of the proxy can be checked with the Globus command `grid-proxy-info` like follows:

```
host1>grid-proxy-info -all
subject   : /O=Grid/O=CERN/OU=cern.ch/CN=Firstname Surname/CN=proxy
issuer    : /O=Grid/O=CERN/OU=cern.ch/CN=Firstname Surname
type      : full
strength  : 512 bits
timeleft  : 25:59:59 (1.0 days)
```

The subject name “/O=Grid/O=CERN/OU=cern.ch/CN=Firstname Surname” uniquely identifies a Grid user. Before you want to access a remote machine, e.g. set up a GDMP transfer between `host1.infn.it` and `host1.cern.ch`, you need to request the remote administrator to enter your subject name to the machines “grid-mapfile” that holds all authenticated users. Note that this only needs to be done once before the GDMP installation process.

14 Appendix C: Trouble Shooting for GDMP Server Configuration

In principle, the GDMP server is completely configured and set up after `configure_gdmp` is executed and the configuration file `gdmp.conf` is filled in correctly. However, no installation is perfect and below is a list of possible errors.

In order to test if the server is installed correctly, run `gdmp_ping -r hostname -p portnumber`. Note, in order to do that, you need to be in the GDMP grid-mapfile of the GDMP host. We now assume that the GDMP server is installed on host `testbed008.cern.ch` and is running on port 2000.

```
[userid@testbed008] gdmp_ping -r testbed008.cern.ch -p2002
Try to get a connection testbed008.cern.ch:2002
The GDMP server testbed008.cern.ch:2002 is listening [ Wed Jan 30 16:30:14 2002 ]
```

- **Wrong compiler is installed**

GDMP requires the compiler `gcc-2.95.2` and thus the shared library `libstdc++-libc6.1-2.so.3`:

```
[userid@testbed008] gdmp_ping -r testbed008 -p2000
gdmp_ping: error in loading shared libraries: libstdc++-libc6.1-2.so.3: \
cannot open shared object file: No such file or directory
```

Make sure you have the correct compiler libraries in the `LD_LIBRARY_PATH`. For instance, the compiler can be installed in `/usr/local/lib` and then add this path to your path. e.g. in `tcsh`:

```
setenv LD_LIBRARY_PATH /usr/local/lib:${LD_LIBRARY_PATH}
```

If you try `gdmp_ping` again, you might see the following error:

```
[userid@testbed008] gdmp_ping -r testbed001 -p2000
Try to get a connection testbed008:2000
Communication Error: the buffer size is not valid! [ Wed Jan 30 16:41:38 2002 ]
Security Error: receiving server message [ Wed Jan 30 16:41:38 2002 ]
GDMP_Req_Manager::check_authorization(): Error: establishing context [ Wed Jan 30 16:41:38 2002 ]
```

Note: If this is the case *also* the script `gdmp_server_start` needs to be changed and the compiler path needs to be added. Thus, add the following line into `gdmp_server_start`:

```
setenv LD_LIBRARY_PATH /usr/local/lib
```

and it should look like follows

```
#!/bin/csh
#
setenv LD_LIBRARY_PATH /usr/local/lib
#
# Identify GDMP_INSTALL_PATH.
#
```

- GDMP server does not respond:

```
[userid@testbed008] gdm_ping -r testbed008 -p2000
Try to get a connection testbed001:20003
.....
The server testbed008 did not respond on the port 2000 ... [ Wed Jan 30 16:48:42 2002 ]
```

First, check if the server is listening on the port with:

```
[userid@testbed008] netstat -an | grep 2000
tcp          0      0 0.0.0.0:2000          0.0.0.0:*              LISTEN
```

This means that the server is running. Check now `sbin/gdmp_server_start` and run it:

```
[userid@testbed008] gdmp_server_start
/bin/gdmp_server: Command not found.
```

It looks like the `GDMP_INSTALL_PATH` is not correct in the file `gdmp_server_start`. Modify it manually. Let's assume GDMP is installed in `/opt/edg`. Add the absolute path for the GDMP installation like follows:

```
#!/bin/csh
#
setenv LD_LIBRARY_PATH /usr/local/lib
#
# Identify GDMP_INSTALL_PATH.
#
set gdm_serpath='echo $0 | awk -F/ '{print substr($0,1,index($0,$NF)-2)}'
set gdm_path=/opt/edg
```

15 Appendix D: Special Instructions for Usage in the EDG Testbed

In the EDG testbed, people do not have access to the SE directly but have to submit GDMP jobs via the User Interface (WP1 software). This also means that *for each client command* the options **-S (server)** **-P (port)** have to be used as described in Section 6.2. For completeness, we state the entire procedure again and explain it by using `gdmp_ping`:

1. log on to the User Interface machine
2. get a proxy via `grid-proxy-init`
3. write a jdl file that contains your GDMP command (see further comments below)
4. submit your jdl file to the RB via `dg-job-submit`

A JDL script (`gdmp_ping.jdl`) for calling the JDL file looks like follows:

```
Executable      = "gdmp_ping.csh";
StdOutput       = "gdmp_ping.out";
StdError        = "gdmp_ping.err";
InputSandbox    = {"temp/gdmp_ping.csh"};
OutputSandbox  = {"gdmp_ping.out", "gdmp_ping.err"};
Requirements    = other.LRMSType == "PBS" && other.OpSys=="RH 6.2";
Rank            = other.MaxCpuTime;
```

The shell script (`gdmp_ping.csh`) file looks as follows. Note that our server we are connecting to is `lxshare0222` but we want to ping `lxshare0219`. Note the usage of the parameters `-P` and `-S`:

```
#!/bin/csh
#
gdmp_ping -S lxshare0222.cern.ch -P 2000 -r lxshare0219.cern.ch -p 2000
#
```

15.1 Additional Hints

15.1.1 `gdmp_register_local_file`

For instance, on `lxshare0222` we want to register files in the directory `/flatfiles/SE1/cms` and need to do that as follows:

```
gdmp_register_local_file -S lxshare0222.cern.ch -P 2000 -d /flatfiles/SE1/cms
```

Note again the options `-P` and `-S`.

To sum up, users always have to refer to the directory structure on the SE !!! GDMP assumes that the file is already there and it is up to the user to copy it to that location with e.g. `globus-url-copy`.

15.2 `gdmp_replicate_get` and others

The correct usage is as follows:

```
gdmp_replicate_get -S lxshare0222.cern.ch -P 2000
```

additional options can be provided but `-S` and `-P` are essential. This is also true for:

- `gdmp_publish_catalogue`
- `gdmp_get_catalogue`
- `gdmp_ping`
- `gdmp_remove_local_file`
- `gdmp_replicate_put`

15.3 Miscellaneous

Additional EDG testbed specific hints and instructions can be found at:

<http://cmsdoc.cern.ch/cms/grid/edg-testbed/>

where we can also put up new questions and answers.

Acknowledgements

The GDMP project (originally called Grid Data Management Pilot) was started in early 2000 as a pilot project by Heinz Stockinger and Asad Samar to evaluate the Globus toolkit, take useful features for a file replication system and produce a prototype to be evaluated in a real production environment.

The software development process is already well advanced and the project is now a collaboration between the European DataGrid (in particular the Data Management work package) and the Particle Physics Data Grid (PPDG). Thus, the GDMP team was increased and we got lots of constructive feedback from our colleagues in DataGrid and PPDG. In particular, we want to thank the WP2 team (the bigger subset that is not already in the GDMP team): Wolfgang Hoeschek, Peter Kunszt, Javier Jaen-Martinez, Ben Segal, Kurt Stockinger and Brian Tierney.

From the US side, we want to thank the Globus team: in particular Bill Allcock, John Bresnahan, Ann Chevernak, Ian Foster, Carl Kesselman, Darcy Quesnel and Mike Wilde (we definitely have forgotten a few names (sorry about that) but these are the people we have most contacts with). Thanks also to Miron Livny (Univ. of Wisconsin) for good discussions which led to several useful additions to GDMP.

Within the DataGrid testbed we got constructive feedback from several people like Stephen Burke and Jeffrey Templon.

Versions 1.0 until 1.2.2

We keep the acknowledgement of older GDMP versions too since the following people gave constructive feedback and discussion during the first project phase:

We want to thank Tony Wildish (Princeton University) for initial work on a replication tool written in Perl. This tool [14] and personal discussions have provided us with important input for the architecture of GDMP. Furthermore, thanks to Koen Holtman (Caltech) who is re-using and testing parts of our code and has pointed out some bugs in the software. Thank you also to Shahzad Muzaffar (Fermi National Lab.) for taking part in the transatlantic replication tests and fixing bugs in GDMP. Thanks to Flavia Donno (INFN) for very valuable input and providing an installation procedure for GDMP within the INFN Installation. Finally we want to thank Luciano Barone (INFN), Dominique Boutigny (IN2P3), Johannes Gutleber (CERN), Mehnaz Hafeez (CERN), Koen Holtman (Caltech), Wolfgang Hoeschek (CERN), Bob Jacobson (LBL), Werner Jank (CERN), Javier Jaen-Martinez (CERN), Veronique Lefebure (CERN), Harvey Newman (Caltech), Ben Segal (CERN), Arie Shoshani (LBL), and Kurt Stockinger (CERN) for their input and valuable discussions. We are also thankful to the Globus team for providing support and technical advice on various issues.

References

- [1] L. M. Bernardo, A. Shoshani, A. Sim, H. Nordberg. Access Coordination of Tertiary Storage for High Energy Physics Application, *17th IEEE Symposium on Mass Storage Systems and 8th NASA Goddard Conference on Mass Storage Systems and Technologies*, Maryland, USA, March 27-30, 2000.
- [2] Ann Chervenak, Ewa Deelman, Ian Foster, Wolfgang Hoeschek, Adriana Iamnitchi, Carl Kesselman, Peter Kunszt, Matei Ripeanu, Heinz Stockinger, Kurt Stockinger, and Brian Tierney. Giggle: A Framework for Constructing Scalable Replica Location Services. In Proc. of the Int'l. IEEE Supercomputing Conference (SC 2002) (to appear), Baltimore, USA, November 2002.
- [3] European DataGrid Project: <http://www.eu-datagrid.org>
- [4] Data Management Work Package in EDG: <http://grid-data-management.web.cern.ch/grid-data-management>

- [5] edg-replica-manager: <http://grid-data-management.web.cern.ch/grid-data-management/edg-replica-manager>
- [6] Andrea Domenici, Notes on the Usage of an experimental Replica Catalog for the CERN DataGrid Testbed, 14 August 2001. <http://www.cern.ch/grid-data-management/docs/ldapuse.ps>
- [7] Globus Project: Getting Started with the Globus Replica Catalog, <http://www.globus.org/datagrid/deliverables/replicaGettingStarted.pdf>
- [8] Mehnaz Hafeez, Asad Samar, Heinz Stockinger. A DataGrid Prototype for Distributed Data Production in CMS, *VII International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT2000)*, October 2000.
- [9] Wolfgang Hoschek, Javier Jean-Martinez, Peter Kunszt, Ben Segal, Heinz Stockinger, Kurt Stockinger, Brian Tierney. Data Management (WP2) Architecture Report - Design, Requirements and Valuation Criteria, *DataGrid-02-D2.2-0103-1_2*, http://grid-data-management.web.cern.ch/grid-data-management/docs/DataGrid-02-D2.2-0103-1_2.pdf, Geneva, Sept 19, 2001.
- [10] Flavia Donno, Salvo Monforte, Francesco Prelz, Livio Salconi, Massimo Sgaravatto. The Resource Broker Info File, *DataGrid-01-NOT-0113* http://www.pd.infn.it/~sgaravat/Grid/datagrid-01-not-0113-1_2.pdf Pisa, Sept 28, 2001.
- [11] Particle Physics Data Grid project (PPDG): <http://www.ppdg.net>
- [12] Asad Samar, Heinz Stockinger. Grid Data Management Pilot (GDMP): A Tool for Wide Area Replication, *IASTED International Conference on Applied Informatics (AI2001)*, Innsbruck, Austria, February 19-22, 2001.
- [13] Heinz Stockinger, Asad Samar, Bill Allcock, Ian Foster, Koen Holtman, Brian Tierney. File and Object Replication in Data Grids, *10th IEEE International Symposium on High Performance and Distributed Computing (HPDC-10)*, San Francisco, California, August 7-9, 2001.
- [14] <http://www.cern.ch/wildish>
- [15] Wengyik Yeong, Tim Howes, Steve Kille. Lightweight Directory Access Protocol, Request For Comments (RFC) 1777, March 1995.