

GT 4.0.5+ Contribution: GridWay

GT 4.0.5+ Contribution: GridWay

Overview

This component is a **contribution** for the GT 4.0 final distribution. Contributions are not supported by GT and have very limited GT documentation.

Important

This contribution has been added as of GT 4.0.5. In addition to the Release Notes and Admin Guide below, the following links take you to additional documentation available on the GridWay website.

- [User's Guide](#)¹
- [Application Developer's Guide](#)²

Starting with version 4.0.5, GT includes the GridWay Metascheduler, which enables large-scale, reliable and efficient sharing of computing resources (clusters, computing farms, servers, supercomputers...), managed by different LRM (Local Resource Management) systems, such as PBS, SGE, LSF, Condor..., within a single organization (enterprise grid) or scattered across several administrative domains (partner or supply-chain grid).

There exist a number of commercial and open source workload management and scheduling systems available today, each one suitable for different underlying computer infrastructures and execution profiles. GridWay stands out from other metascheduling systems because it has been specifically designed to work on top of Globus services, offering the highest functionality, quality of service and reliability on this kind of infrastructure, namely:

- **For project and infrastructure directors:** GridWay is an open-source community project, adhering to Globus philosophy and guidelines for collaborative development.
- **For system integrators:** GridWay is highly modular, allowing adaptation to different grid infrastructures, and supports several OGF standards.
- **For system managers:** GridWay provides a scheduling framework similar to that found on local LRM systems, supporting resource accounting and the definition of state-of-the-art scheduling policies.
- **For application developers:** GridWay implements the OGF standard DRMAA API (C and JAVA bindings), assuring compatibility between applications with LRM systems that implement the standard, such as SGE, Condor, Torque...etc.
- **For end users:** GridWay provides an LRM-like CLI for submitting, monitoring, synchronizing and controlling jobs, that could be described using the OGF standard JSDL.

GridWay gives end users, application developers and managers of Globus infrastructures a scheduling functionality similar to that found on local DRM systems.

¹ <http://www.gridway.org/documentation/stable/userguide/>

² <http://www.gridway.org/documentation/stable/appdeveloperguide/>

Table of Contents

1. System Administrator's Guide	1
1. Introduction	1
2. Building and installing	3
3. GridWay Core Configuration Guide	5
4. Scheduler Configuration Guide	9
5. MAD Configuration Guide	19
6. Deploying	26
7. Testing	26
8. Security considerations	29
9. Troubleshooting	29
2. Fact Sheet	30
1. Brief component overview	30
2. Summary of features	30
3. Backward compatibility summary	31
4. Technology dependencies	32
5. Tested platforms	32
6. Associated standards	33
7. For More Information	33
3. 4.0.8 Release Notes	34
1. Introduction	34
2. Changes Summary	34
3. Bug Fixes	34
4. Known Problems	34
5. For More Information	35
4. 4.0.7 Release Notes	36
1. Introduction	36
2. Changes Summary	36
3. Bug Fixes	36
4. Known Problems	36
5. For More Information	37
5. 4.0.6 Release Notes	38
1. Introduction	38
2. Changes Summary	38
3. Bug Fixes	38
4. Known Problems	39
5. For More Information	40
6. 4.0.5 Release Notes	41
1. Component Overview	41
2. Feature Summary	41
3. Technology Dependencies	42
4. Supported Platforms	43
5. Backward Compatibility Summary	43
6. Changes Summary	44
7. Bug Fixes	44
8. Known Problems	45
9. For More Information	45

List of Figures

1.1. Components of the GridWay Meta-scheduler	1
1.2. Site-level Meta-scheduling Infrastructure.	3
1.3. Enterprise Grid deployment with multiple-user mode GridWay.	3
1.4. Job Scheduling in GridWay	10
1.5. Job and resource prioritization policies in GridWay.	11
1.6. Dispatch priority of a job	13
1.7. Estimated execution time of a job on a resource	15
1.8. Banned time formula	15
1.9. Suitability priority of a resource	16

List of Tables

1.1. GWD Configuration File Options.	7
1.2. Built-in Scheduler Configuration File Options.	17
1.3. GridWay tests description.	28

Chapter 1. GT 4.0.5+ GridWay: System Administrator's Guide

1. Introduction

This guide contains installation and configuration information for system administrators installing GridWay. It explains how to install, configure and test the installation.

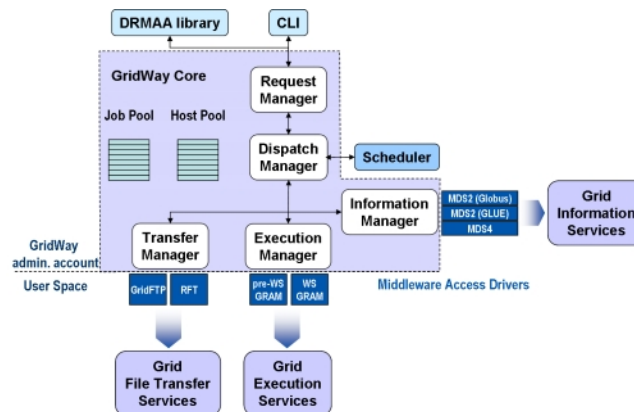
! Important

This information is in addition to the basic Globus Toolkit prerequisite, overview, installation, security configuration instructions in the [GT 4.0 System Administrator's Guide](#)¹. Read through this guide before continuing!

1.1. GridWay Architecture

In GridWay 4.0.2, we introduced an architecture for the execution manager module based on a MAD (Middleware Access Driver) to interface with Grid execution services. In this release we have taken advantage of this architecture to implement an information manager module with a MAD that interfaces with Grid information services, and a transfer manager module with a MAD that interfaces with Grid data services. Moreover, we have decoupled the scheduling process from the dispatch manager through the use of an external and selectable scheduler module.

Figure 1.1. Components of the GridWay Meta-scheduler.



GridWay 5 architecture consists of the following components:

- *User Interface* provides the end user with DRM-like commands to submit, kill, migrate, monitor and synchronize jobs and includes DRMAA (Distributed Resource Management Application API) GGF (Global Grid Forum) standard support to develop distributed applications (C and JAVA bindings).
- *GridWay core* is responsible for job execution management and resource brokering, providing advanced scheduling, and job failure & recovery capabilities. The Dispatch Manager performs all submission stages and watches over the efficient execution of the job. The Information Manager, through its MADs (Middleware Access Driver), is responsible for host discovery and monitoring. The Execution Manager, through its MADs, is responsible for job exe-

¹ admin/admin/admin/docbook/

cution and management. The Transfer Manager, through its MADs, is responsible for file staging, remote working directory set-up and remote host clean-up.

- *Scheduler* makes scheduling decisions for jobs on available resources.
- *Information Manager MAD* interfaces with the monitoring and discovering services available in the Grid infrastructure.
- *Execution Manager MAD* interfaces with the Job Management Services available in the Grid resources.
- *Transfer Manager MAD* interfaces with the Data Management Services available in the Grid resources.

1.2. Meta-scheduling Infrastructures with GridWay

1.2.1. Partner Grid Infrastructures

Partner grid infrastructures of several scales are being deployed within the context of different research projects whose final goal is to provide large-scale, secure and reliable sharing of resources among partner organizations and supply-chain participants. Such partner grids allow access to higher computing performance to satisfy peak demands and also provide support to interface with collaborative projects. The multiple administration domains existing in a partner grid infrastructure prevent the deployment of centralized meta-schedulers, with total control over client requests and resource status.

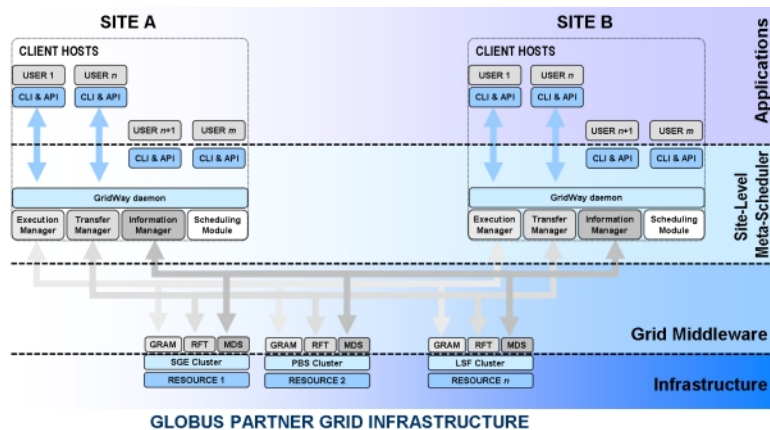
1.2.1.1. User-level Meta-scheduling Infrastructures

Most current meta-schedulers are user-level tools that provide scheduling and job management functionality. This means that there is one scheduling instance for each user, and all scheduling instances compete with each other for the available resources. In this scenario, GridWay is installed in single-user mode by each end-user on his client host.

1.2.1.2. Organization-level Meta- scheduling Infrastructures

User-level meta-scheduling deployment may exhibit technical and socio-political difficulties within an organization, namely: the end users are responsible for the installation and administration of the meta-scheduler, a Globus client installation is required in each end-user system, and firewall configuration must allow traffic between grid resources and end user systems. Undoubtedly, an approach that gives administrators full control of meta-scheduling deployment could help overcome these difficulties. Organization-level meta-schedulers provide support for multiple intra-organization users in each scheduling instance. This means that there is one scheduling instance for each organization, and all scheduling instances compete with each other for the available resources.

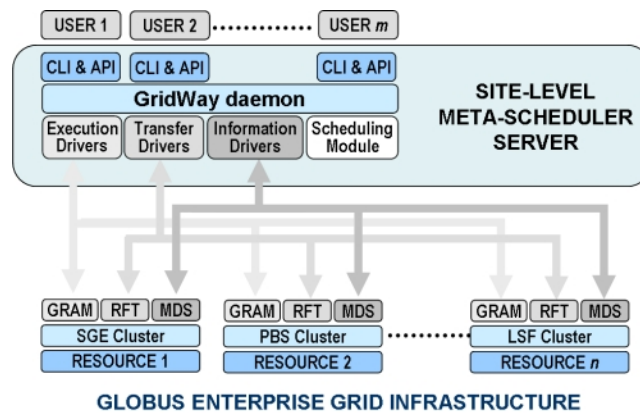
In this scenario, GridWay is configured in multiple-user mode. This way the installation and configuration of GridWay will be performed by each system manager and the users could submit, control and monitor their jobs from a front-end or from submission hosts (that do not require GridWay and Globus installation).

Figure 1.2. Site-level Meta-scheduling Infrastructure.

1.2.2. Enterprise Grid Infrastructures

Enterprise grids enable diverse resource sharing to improve internal collaboration and achieve a better return from information technology investment. Available resources within a company are better exploited and the administrative overhead is minimized by using Grid technology. The resources are part of the same administrative domain. These infrastructures require a centralized approach for scheduling and accounting. The administrator must be able to apply centralized usage policies and access to global reporting and accounting. Enterprise grid infrastructures require meta-schedulers to provide support for multiple users in a single scheduling instance.

In this scenario, GridWay is installed in multiple-user mode. This way the installation and configuration of GridWay will be performed by the system manager and the users could submit, control and monitor their jobs from a front-end or from submission hosts (that do not require GridWay and Globus installation).

Figure 1.3. Enterprise Grid deployment with multiple-user mode GridWay.

2. Building and installing

The full source [installation](http://www.globus.org/toolkit/docs/4.0/admin/docbook/)² of GT 4.0.5+ will build the prerequisite libraries with the correct flavor.

² <http://www.globus.org/toolkit/docs/4.0/admin/docbook/>

2.1. Required Software

GridWay is distributed as a source package; the following software is required to compile it:

- C compiler: Tested versions gcc 3.4.2, 3.4.4, 4.0.3, 4.0.3 and 4.1.2
- Globus C libraries: globus_gram_client, globus_ftp_client and globus_gass_copy
- Globus JAVA development libraries
- J2SE versions 1.4.2_10+ (Builds higher than 10) or 1.5.0+
- GNU Make
- Sudo command (only required for multiple-user mode)
- Berkeley Database library version 4.4.20 (only required to compile the accounting module)

2.2. Building

To build GridWay alongside Globus you have two useful options you can pass to the Globus configure script. Here is an example:

```
$ ./configure --enable-gridway --with-gridwayargs="<gridway args>"
```

These two options are described in this table:

Option	Description
--enable-gridway	Tells globus to build GridWay. This option is disabled by default.
--with-gridwayargs	This option is used to pass parameters to the GridWay configuration interface. These parameters should be separated by spaces. The parameters you can use and their descriptions can be found in the table below.

Specific GridWay building parameters:

Option	Description
--disable-drmad	Don't build DRMAA support. Default is enabled.
--disable-prews	Don't build pre-web-services support. Default is enabled.
--disable-ws	Don't build web-services support. Default is enabled.
--enable-globus-scheme	Adds GridWay subdirectories to etc and var. This parameter is meant to be used within a gpt-build process, not directly. Default is disabled.
--enable-jsdl	Does compile JSDL support. Default is disabled.
--disable-gridftp	Does not compile GridFTP MAD. Default is enabled.
--with-db=path_to_db	Specify the Berkeley Database path to build accounting support.
--with-doc	Install GridWay documentation
--with-tests	Install tests

3. GridWay Core Configuration Guide

GridWay requires that the environment variables `GLOBUS_LOCATION` and `GW_LOCATION` are set. These are set to the base of your Globus installation and GridWay installation. In GT 4.0.5+, GridWay is installed in the same place as Globus, so you can set both of these environment variables to the same location.

Important

Note that the GridWay daemon **SHOULD NOT** be run as root. Only part of the installation will require privileged access.

Login as root account and follow the next steps:

1. All of the GridWay users must be members of the same UNIX group, `<gwgroup>`. We recommend creating a new group (called `gwusers`, for example), and make sure that all GridWay user accounts are members of this new group.
2. The GridWay administrator account, `<adminuser>`, can be an existing administrative login or a new login. We recommend using the Globus account for the GridWay administration user. This account will own all of the files in the GridWay installation plus all of the daemons in the GridWay execution and it can be used to configure GridWay once it is installed. Primary group of `<adminuser>` should be `<gwgroup>`.

DO NOT use root account for the GridWay administrator account.

3. The `sudoers` file of the **sudo** command should include the following

```
...
# User alias specification
...
Runas_Alias      GW_USERS = %<gwgroup>
...
# GridWay entries
globus ALL=(GW_USERS)      NOPASSWD: /home/gwadmin/gw/bin/gw_em_mad_prews *
globus ALL=(GW_USERS)      NOPASSWD: /home/gwadmin/gw/bin/gw_em_mad_ws *
globus ALL=(GW_USERS)      NOPASSWD: /home/gwadmin/gw/bin/gw_tm_mad_ftp *
```

Usually **sudo** clears all environment variables for security reasons. However MADs need the `GW_LOCATION` and `GLOBUS_LOCATION` variables to be set. To preserve those variables in the MAD environment, add the following line to your `sudoers` file:

```
Defaults>GW_USERS env_keep="GW_LOCATION GLOBUS_LOCATION"
```

Please refer to the **sudo** manual page for more information.

To test the **sudo** command configuration try to execute a MAD as a user in the `<gwgroup>` group, for example:

```
$ sudo -u <gw_user> /home/gwadmin/gw/bin/gw_em_mad_prews
```

3.1. Configuration Interface

The configuration files for GridWay are read from the following locations:

- `$GW_LOCATION/etc/gridway/gwd.conf`: Configuration options for the GridWay daemon (GWD).
- `$GW_LOCATION/etc/gridway/sched.conf`: Configuration options for the GridWay built-in scheduling policies (see [Section 4.6, “Scheduler Configuration”](#) for more information).
- `$GW_LOCATION/etc/gridway/job_template.default`: Default values for job's templates (i.e. job definition files).
- `$GW_LOCATION/etc/gridway/gwrc`: Default environment variables for MADs.

Options are defined one per line, with the following format:

```
<option> = [value]
```

If the value is missing the option will fall back to its default. Blank lines and any character after a '#' are ignored. Note: Job template options can use job or host variables to define their value, these variables are substituted at run time with their corresponding values (see the [GridWay user guide](#)³).

3.2. GridWay Daemon (GWD) Configuration

The GridWay daemon (GWD) configuration options are defined in `$GW_LOCATION/etc/gridway/gwd.conf`. The table below summarizes the configuration file options, their description and default values. Note that blank lines and any character after a '#' are ignored.

³ <http://www.gridway.org/documentation/stable/userguide/>

Table 1.1. GWD Configuration File Options.

Option	Description	Default
Connection Options		
GWD_PORT	TCP/IP Port where GWD will listen for client requests. If this port is in use, GWD will try to bind to the next port until it finds a free one. The TCP/IP port used by GWD can be found in <code>\$GW_LOCATION/var/gridway/gwd.port</code>	6725
MAX_NUMBER_OF_CLIENTS	Maximum number of simultaneous client connections. Note that only blocking client requests keeps its connection open.	20
Pool Options		
NUMBER_OF_JOBS	The maximum number of jobs that will be handled by the GridWay system	200
NUMBER_OF_ARRAYS	The maximum number of array-jobs that will be handled by the GridWay system	20
NUMBER_OF_HOSTS	The maximum number of hosts that will be handled by the GridWay system	100
NUMBER_OF_USERS	The maximum number of different users in the GridWay system	30
Intervals		
SCHEDULING_INTERVAL	Period (in seconds) between two scheduling actions	30
DISCOVERY_INTERVAL	How often (in seconds) the information manager searches the Grid for new hosts	300
MONITORING_INTERVAL	How often (in seconds) the information manager updates the information of each host	120
POLL_INTERVAL	How often (in seconds) the underlying grid middleware is queried about the state of a job.	60
Middleware Access Driver (MAD) Options		
IM_MAD	Information Manager MADs, see Section 5.4, "Information Driver Configuration"	-
TM_MAD	Transfer Manager MADs, see Section 5.3, "File Transfer Driver Configuration"	-
EM_MAD	Execution Manager MADs, see Section 5.2, "Execution Driver Configuration"	-
MAX_ACTIVE_IM_QUERIES	Maximum number (soft limit) of active IM queries (each query spawns one process)	4
Scheduler Options		
DM_SCHED	Scheduling module, see Section 4.6, "Scheduler Configuration"	-

Here is an example of a GWD configuration file:

```
#-----
# Example: GWD Configuration File
#-----
GWD_PORT           = 6725
MAX_NUMBER_OF_CLIENTS = 20
```

```
NUMBER_OF_ARRAYS = 20
NUMBER_OF_JOBS   = 200
NUMBER_OF_HOSTS  = 100
NUMBER_OF_USERS  = 30
JOBS_PER_SCHED  = 10
JOBS_PER_HOST    = 10
JOBS_PER_USER    = 30
SCHEDULING_INTERVAL = 30
DISCOVERY_INTERVAL = 300
MONITORING_INTERVAL = 120
POLL_INTERVAL    = 60
IM_MAD = mds4:gw_im_mad_mds4:-s hydrus.dacya.ucm.es::gridftp:ws
TM_MAD = gridftp:gw_tm_mad_ftp:
EM_MAD = ws:gw_em_mad_ws:rsl2
DM_SCHED = flood:gw_flood_scheduler:-h 10 -u 30 -c 5
```

3.3. Job Template Default Values

Default values for *every* job template option can be set in `$GW_LOCATION/etc/gridway/job_template.default`. You can use this file to set the value of advanced job configuration options and use them for all your jobs. Note that the values set in a job template file override those defined in `job_template.default`. See the [GridWay user guide](#)⁴ for a detailed description of each job option.

3.4. Daemon Failure Recovery

Since GridWay 4.9, when you start the daemon, **gwd** tries to recover its previous state. This is, any submitted job is stored in a persistent pool, and in case of a **gwd** (or client machine) crash these jobs are recovered. This includes, for jobs in wrapper state, contacting with the remote jobmanager.

Recovery actions are performed by default, if you do not want to recover the previous submitted jobs use the **-c** option.

For example, to start **gwd** in multi-user mode and clear its previous state, use:

```
$ gwd -c -m
```

3.5. Logging

GridWay reporting and accounting facilities provide information about overall performance and help troubleshoot configuration problems. GWD generates the following logs under the `$GW_LOCATION/var` directory:

- `$GW_LOCATION/var/gwd.log`: System level log. You can find log information of the activity of the middleware access drivers; and a coarse-grain log information about jobs.
- `$GW_LOCATION/var/sched.log`: Scheduler log. You can find log information to fit the scheduler policies to your organization needs.
- `$GW_LOCATION/var/$JOB_ID/job.log`: Detailed log information for each job, it includes details of job resource usage and performance.
- `$GW_LOCATION/var/acct`: Accounting information. Use the **gwacct** command to access the data bases. Note that you need Berkeley DB library (version 4.4.20).

⁴ <http://www.gridway.org/documentation/stable/userguide/>

- `$GW_LOCATION/var/.lock`: Used to prevent from running more than one instance of the daemon.
- `$GW_LOCATION/var/gwd.port`: TCP/IP port GWD is listening for client connection requests.
- `$GW_LOCATION/var/globus-gw.log`: Used to encapsulate GridWay in a GRAM service (please refer to the Grid4Utility project web page for more information). This log file follows the globus fork job starter's format (based on SEG, Scheduler Event Generator messages):

```
001;TIMESTAMP;JOBID;STATE;EXIT_CODE
```

4. Scheduler Configuration Guide

Grid scheduling consists of finding a suitable (in terms of a given target) assignment between a computational workload (jobs) and computational resources. The scheduling problem has been thoroughly studied in the past and efficient algorithms have been devised for different computing platforms. Although some of the experience gained in scheduling can be applied to the Grid, it presents some characteristics that differ dramatically from classical computing platforms (i.e. clusters or MPPs), namely: different administration domains, limited control over resources, heterogeneity and dynamism.

Grid scheduling is an active research area. The Grid scheduling problem is better understood today and several heuristics, performance models and algorithms have been proposed and evaluated with the aid of simulation tools. However, current working Grid schedulers are only based on match-making, and barely consider multi-user environments.

In this section, we describe the state-of-the-art scheduling policies implemented in the GridWay system. The contents of this guide reflect the experience obtained since GridWay version 4, and a strong feedback from the GridWay user community.

4.1. GridWay Scheduling Architecture

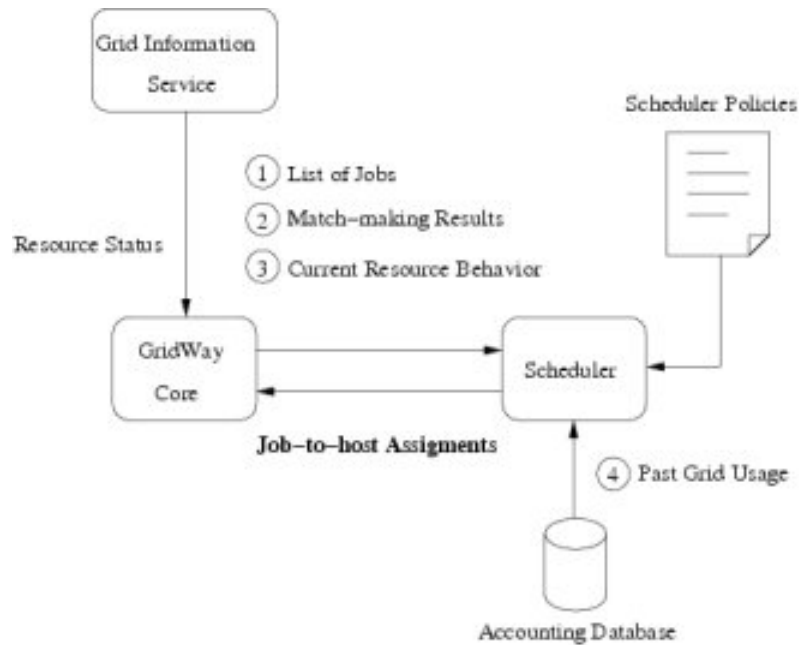
The scheduler is responsible for assigning jobs to Grid resources; therefore, it decides when and where to run a job. These decisions are made periodically in an endless loop. The frequency of the scheduler interventions can be adjusted with the `SCHEDULER_INTERVAL` configuration parameter (see [Section 3.2, “GridWay Daemon \(GWD\) Configuration”](#)).

In order to make job to resource assignments the scheduler receives information from the following sources (see [Figure 1.4, “Job Scheduling in GridWay”](#)):

1. *List of jobs in the system*, which includes pending jobs as well as running jobs (those in wrapper state). Those jobs that cannot be started are filtered out from the list, i.e., jobs with unmet dependencies, stopped or held.
2. *Match-making results*: The Information Manager drivers query the Grid information services to track the availability and status of Grid resources. The discovery and monitoring processes can both be configured as static or dynamic, see [Section 5.4, “Information Driver Configuration”](#). This information is used by the GridWay core to build a list of suitable resources for each job, i.e., resources meeting the job requirements, and to compute their rank.
3. *Current resource behavior*: The scheduler considers the way a resource is behaving when making its decisions. In particular, it evaluates the migration and failure rates and execution statistics (transfer, execution and queue wait times).
4. *Past Grid Usage*: The scheduler also considers the past history (behavior) of Grid resources to issue schedules. Note that database support needs to be compiled in GridWay for this feature.

The information gathered from the previous sources is combined with a given scheduling policy to prioritize jobs and resources. Then, the scheduler dispatches the highest priority job to the best resource for it. The process continues until all jobs are dispatched, and those that could not be assigned wait for the next scheduling interval.

Figure 1.4. Job Scheduling in GridWay

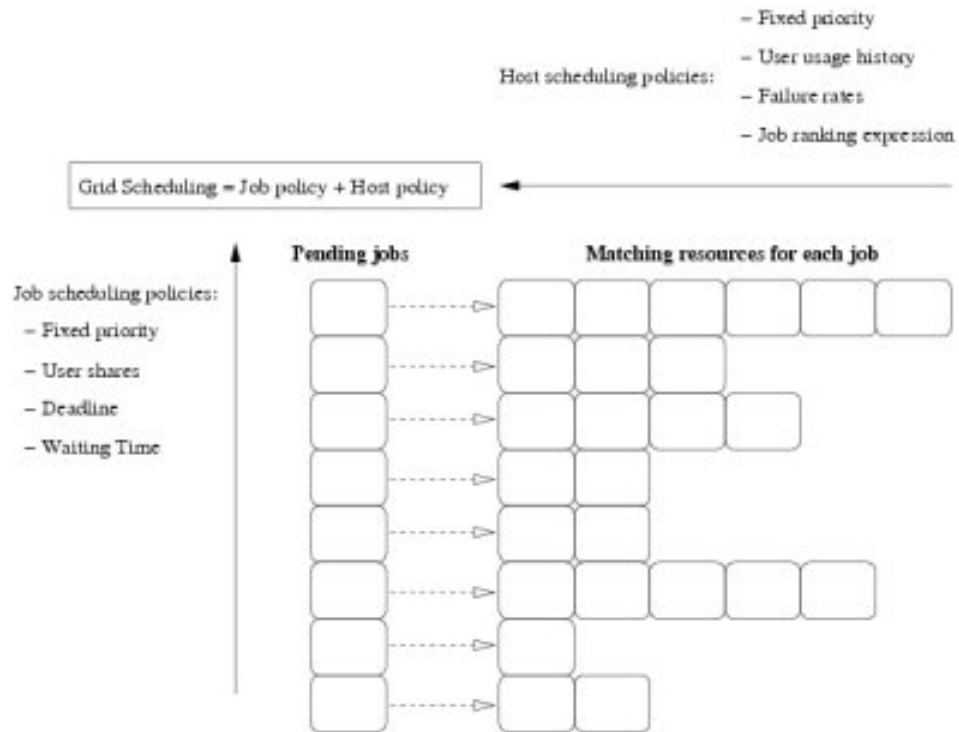


4.2. Scheduling Policies

A scheduling policy is used to assign a *dispatch priority* to each job and a *suitability priority* to each resource. Therefore, a Grid scheduling policy comprises two components:

- *Job prioritization policies.* Pending jobs are prioritized according to four policies: fixed, share, deadline and waiting-time. The job policies are used to sort the jobs of the users of a given scheduling domain (GridWay instance). Note that these policies are only enforced in the scheduling domain and not for the whole Grid infrastructure as discussed above.
- *Resource prioritization policies.* A given job can be executed on those resources that match its requirements. The resource policies are used to sort the matching resource list of each job. The matching resources are prioritized according to four policies: fixed, usage, failure and rank. Note that these policies do not only depend on the Grid resource but also on the job owner, as each Grid user (or VO member) has its own access rights and usage history.

These two top-level policies can be combined to implement a wide range of scheduling schemes (see [Figure 1.5, “Job and resource prioritization policies in GridWay.”](#)). The above scheduling policies are described in the following sections.

Figure 1.5. Job and resource prioritization policies in GridWay.

4.3. Job Prioritization Policies

The job prioritization policies allow Grid administrators to influence the dispatch order of the jobs, that is, to decide which job is sent to the Grid. Traditionally, DRMS implement different policies based on the owner of the job, the resources consumed by each user or the requirements of the job. Some of these scheduling strategies can be directly applied in a Grid, while others must be adapted because of their unique characteristics: dynamism, heterogeneity, high fault rate and site autonomy.

4.3.1. Fixed Priority Policy (FP)

This policy assigns a fixed priority to each job. The fixed priority ranges from 00 (lowest priority) to 19 (highest priority), so jobs with a higher priority will be dispatched first. The default priority values are assigned, by the Grid administrator, using the following criteria:

- *User.* All jobs of a User are given a fixed priority.
- *Group.* All jobs of a user belonging to a given Group get a fixed priority.

The user priority prevails over the group one. Also there is a special user (DEFAULT) to define the default priority value when no criteria apply.

The users can set the priority of their own jobs (**gws submit -p**) but without exceeding their limit set by the administrator in the scheduler configuration file.

Here is an example configuration for the fixed priority (see also [Section 4.5, "Built-in Scheduler Configuration File"](#)):

```
# Weight for the Fixed priority policy
```

```
FP_WEIGHT = 1

# Fixed priority values for David's and Alice's jobs
FP_USER[david] = 2
FP_USER[alice] = 12

# Fixed priority for every body in the staff group
FP_GROUP[staff] = 5

# Anyone else gets a default priority 3
FP_USER[DEFAULT] = 3
```

4.3.2. Urgent Job Policy

The Grid administrator can also set the fixed priority of a job to 20. When a job gets a fixed priority of 20, it becomes an *urgent job*. Urgent jobs are dispatched as soon as possible, bypassing all the scheduling policies.

4.3.3. Fair-Share Policy (SH)

The fair-share policy allows you to establish a dispatching ratio among the users of a scheduling domain. For example, a fair-share policy could establish that jobs from David and Alice must be dispatched to the Grid in a 2:5 ratio. In this case, the scheduler tracks the jobs submitted to the Grid by these two users and dispatches the jobs so they target a 2:5 ratio of job submissions.

This policy resembles the well-known fair-share of traditional LRMS. However, note that what GridWay users share is the ability to submit a job to the Grid and not resource usage. Resource usage share cannot be imposed at a Grid level, as Grid resources are shared with other Grid users and with local users from the remote organization. In addition, the set of resources that can be potentially used by each user is not homogeneous, as each user may belong to a different VO.

GridWay tracks the jobs submitted to the Grid by the users over time. Grid administrators can specify a timeframe over which user submissions are evaluated. The amount of time considered by GridWay is defined by a number of time intervals (`SH_WINDOW_DEPTH`) and the duration of each one (`SH_WINDOW_SIZE`, in days). The effective number of submissions in a given window is exponentially damped, so present events become more relevant.

Here is an example configuration for the share policy (see also [Section 4.5, “Built-in Scheduler Configuration File”](#)):

```
# Weight for the Fair-share policy
SH_WEIGHT = 1

# Shared values for David's and Alice's submissions
SH_USER[david] = 2
SH_USER[alice] = 5

# Anyone else gets a default share value of 1
SH_USER[DEFAULT] = 1

# Consider submissions in the last 5 days
SH_WINDOW_SIZE = 1
SH_WINDOW_DEPTH= 5
```

4.3.4. Waiting-time Policy (WT)

The goal of this policy is to prevent low-priority jobs from starving. So jobs in the pending state long enough will be eventually submitted to the Grid. This policy can be found in most of the DRMS today. In GridWay, the priority of a job is increased linearly with the waiting time.

Here is an example configuration for this policy:

```
# Weight for the Waiting-time policy
WT_WEIGHT = 1
```

4.3.5. Deadline Policy (DL)

GridWay includes support for specifying deadlines at job submission. The scheduler will increase the priority of a job as its deadline approaches.

Important

Note that this policy does not guarantee that a job is completed before the deadline.

Grid administrators should provide a way to qualify the remaining time to reach the job deadline by defining when a job should get half of the maximum priority assigned by this policy (DL_HALF, in days).

Here is an example configuration for the deadline policy (see also [Section 4.5, “Built-in Scheduler Configuration File”](#)):

```
# Weight of the Deadline Policy
DL_WEIGHT = 1

# Assign half of the priority two days before the deadline
DL_HALF = 2
```

4.3.6. The Overall Dispatch Priority of a Job

The list of all pending jobs is sorted by the *dispatch priority*, which is computed as a weighted sum of the contribution from the previous policies. In this way, the Grid administrator can implement different scheduling schemes by adjusting the policy weights.

The *dispatch priority* of a job is therefore computed with:

Figure 1.6. Dispatch priority of a job

$$P_j = \sum_i w_i \cdot p_i \text{ where } i = \{\text{fixed, share, wait-time, deadline}\}.$$

where w_i is the weight for each policy (integer value) and p_i is the priority (normalized) contribution from each policy.

4.4. Resource Prioritization Policies

The resource prioritization policies allow Grid administrators to influence the usage of resources made by the users, that is, decide where to run a job. Usually, in classical DRMS, this resource usage is administered by means of the queue concept.

In GridWay, the scheduler builds a *meta-queue* (a queue consisting of the local queues of the Grid resources) for each job based on its requirements (e.g., operating system or architecture). Note that this *meta-queue* is not only built in terms of resource properties but is also based upon the owner of the job, (as each Grid user may belong to a different VO with its own access rights and usage privileges).

The *meta-queue* for a job consists of the queues of those resources that meet the job requirements specified in the job template and have at least one free slot. By default, this queue is sorted in a first-discovered first-used fashion. This order can be influenced by means of the subsequent resource prioritization policies.

4.4.1. Fixed Resource Priority Policy (RP)

Usually, GridWay is configured with several Information Managers (IM). Grid administrators can prioritize resources based upon the IM that discovered the resource. Grid administrators can also assign priorities to individual resources. For example, a fixed priority policy can specify that resources from the intranet (managed by an IM driver tagged **intranet**) should always be used before resources from other sites (managed by an IM driver tagged **grid**).

The priority of a resource ranges from 01 (lowest priority) to 99 (highest priority), so resources with a higher priority will be used first. Grid administrators can also prioritize individual resources based on business decisions.

When a resource gets the priority value 00, it becomes a **banned resource**, and will not be used for any job. So Grid administrators can virtually *unplug* resources from their scheduling domain.

Example configuration for the resource Fixed Priority Policy:

```
# Weight for the Resource fixed priority policy
RP_WEIGHT = 1

# Fixed priority values for specific resources
RP_HOST[my.cluster.com] = 12
RP_HOST[slow.machine.com] = 02

# Fixed priority for every resource in the intranet
RP_IM[intranet] = 65

# Fixed priority for every resource discovered by the grid IM
RP_IM[grid] = 05

# Anyone else gets a default priority 04 (i.e. other IM)
RP_IM[DEFAULT] = 01
```

4.4.2. Rank Policy (RA)

The goal of this policy is to prioritize those resources more suitable for the job, from its own point of view. For example, the rank policy for a job can state that resources with faster CPUs should be used first. This policy is configured through the RANK attribute in the job template, please refer to the [GridWay user guide](http://www.gridway.org/documentation/stable/userguide/)⁵.

Example configuration for the Rank policy:

```
# Weight of the Rank policy
RA_WEIGHT = 1
```

⁵ <http://www.gridway.org/documentation/stable/userguide/>

4.4.3. Usage Policy (UG)

This policy reflects the behavior of Grid resources based on job execution statistics. So, crucial performance variables, like the average queue wait time or network transfer times, are considered when scheduling a job. This policy is derived from the sum of two contributions: history and current.

- *History contribution.* Execution statistics on a given period of time (for example, average values in the last 3 days). This information is obtained from the accounting database, so GridWay must be compiled with the Berkeley DB libraries.
- *Last job contribution.* Execution statistics of the last job on that resource.

These values are used to compute an estimated execution time of a job on a given resource for a given user:

Figure 1.7. Estimated execution time of a job on a resource

$$T = (1 - w) \cdot (T_{exe}^h + T_{zfr}^h + T_{que}^h) + w \cdot (T_{exe}^c + T_{zfr}^c + T_{que}^c)$$

where T^c are the execution statistics of the last job (execution, transfer and queue wait-time), T^h are the execution statistics based on the history data; and w is the history ratio. Those resources with a lower estimated time are used first to execute a job.

The Usage policy can be configured with:

- UG_HISTORY_WINDOW. Number of days used to compute the execution statistics from the History contribution.
- UG_HISTORY_RATIO. The value of w , use 0 to use only data from the accounting database, and 1 to use only results from the last execution.

Example configuration for Usage policy:

```
# Weight of the Usage policy
UG_WEIGHT = 1

# Number of days in the history window
UG_HISTORY_WINDOW = 3

# Accounting database to last execution ratio
UG_HISTORY_RATIO = 0.25
```

4.4.4. Failure Rate Policy (FR)

When a resource fails, GridWay implements an exponential linear back-off strategy at resource level (and per each user); henceforth, resources with persistent failures are discarded (for a given user).

In particular, when a failure occurs a resource is *banned* for T seconds:

Figure 1.8. Banned time formula

$$T = T_{\infty} \cdot (1 - e^{-\frac{\Delta t}{\sigma}})$$

where T_∞ is the maximum time that a resource can be *banned*, Δt is the time since last failure, and C is a constant that determines how fast the T_∞ limit is reached.

The failure rate policy can be configured with the following parameters:

- **FR_MAX_BANNED_TIME**. The value of T_∞ , use 0 to disable this policy.
- **FR_BANNED_C**. The value of the C constant in the above equation.

Example configuration for the Failure Rate policy:

```
# Maximum time that a resource will not be used, in seconds
FR_MAX_BANNED_TIME = 3600
# Exponential constant
FR_BANNED_C          = 650
```

4.4.5. The Overall Suitability Priority of a Resource

The list of all candidate resources is sorted by the *suitability priority*, which is computed as a weighted sum of the contribution from the previous policies. The *suitability priority* resource is therefore computed with:

Figure 1.9. Suitability priority of a resource

$$P_h = \sum_i w_i \cdot p_i \text{ where } i = \{\text{fixed, usage, rank}\}.$$

where w_i is the weight for each policy (integer value) and p_i is the priority (normalized) contribution from each policy.

4.4.6. Re-scheduling Policies

Also, the scheduler can migrate running jobs in the following situations:

- A better resource is discovered.
- A job has been waiting in the remote queue system more than a given threshold.
- The application changes its requirements.
- A performance degradation is detected.

See Section 3.2, “GridWay Daemon (GWD) Configuration” and the [GridWay user guide](#)⁶, for information on configuring these policies.

4.5. Built-in Scheduler Configuration File

The built-in scheduler configuration options are defined in `$GW_LOCATION/etc/sched.conf`. The table below summarizes the configuration file options, their description and default values. Note that blank lines and any character after a '#' are ignored.

⁶ <http://www.gridway.org/documentation/stable/userguide/>

Table 1.2. Built-in Scheduler Configuration File Options.

Option	Description	Default
Job Scheduling Policies. Pending jobs are prioritized according to four policies:fixed (FP), share(SH), deadline (DL) and waiting-time (WT). The dispatch priority of a job is computed as a weighted sum of the contribution of each policy (normalized to one).		
DISPATCH_CHUNK	The maximum number of jobs that will be dispatched for each scheduling action	15 (0 to dispatch as many jobs as possible)
MAX_RUNNING_USER	The maximum number of simultaneous running jobs per user.	30 (0 to dispatch as many jobs as possible)
<i>Fixed Priority (FP) Policy:</i> Assigns a fixed priority to each job		
FP_WEIGHT	Weight for the policy (real numbers allowed).	1
FP_USER[<username>]	Priority for jobs owned by <username>. Use the special username DEFAULT to set default priorities. Priority range [0,19]	
FP_GROUP[<groupname>]	Priority for jobs owned by users in group <groupname>. Priority range [0,19]	
<i>Share (SH) Policy:</i> Allows you to establish a dispatch ratio among users.		
SH_WEIGHT	Weight for the policy (real numbers allowed).	
SH_USER[<username>]	Share for jobs owned by <username>. Use the special username DEFAULT to set default shares.	
SH_WINDOW_DEPTH	Number of intervals (windows) to "remember" each user's dispatching history. The submissions of each window are exponentially "forgotten".	5, the maximum value is 10.
SH_WINDOW_SIZE	The size of each interval in days (real numbers allowed).	1
<i>Waiting-time (WT) Policy:</i> The priority of a job is increased linearly with the waiting time to prevent job starvation		
WT_WEIGHT	Weight for the policy (real numbers allowed).	0
<i>Deadline (DL) Policy:</i> The priority of a job is increased exponentially as its deadline approaches.		
DL_WEIGHT	Weight for the policy (real numbers allowed).	1
DL_HALF	Number of days before the deadline when the job should get half of the maximum priority.	1
Resource Scheduling Policies. The resource policies allows grid administrators to influence the usage of resources made by the users, according to: fixed (FP), rank (RA), failure rate (FR), and usage (UG). The suitability priority of a resource is computed as a weighted sum of the contribution of each policy (normalized to one).		
MAX_RUNNING_RESOURCE	The maximum number of jobs that the scheduler submits to a given resource	10
<i>Fixed Priority (RP) Policy:</i> Assigns a fixed priority (range [01,99]) to each resource		
RP_WEIGHT	Weight for the policy (real numbers allowed).	1 (real numbers allowed)
RP_HOST[<FQDN>]	Priority for resource <FQDN>. Those resources with priority 00 WILL NOT be used to dispatch jobs.	

RP_IM[<im_tag>]	Priority for ALL resources discovered by the IM <im_tag> (as set in <code>gwd.conf</code> , see Section 3.2, “GridWay Daemon (GWD) Configuration”). Use the special tag <code>DEFAULT</code> to set default priorities for resources.	
<i>Usage (UG) Policy:</i> Resources are prioritized based on the estimated execution time of a job (on each resource).		
UG_WEIGHT	Weight for the policy (real numbers allowed).	1 (real numbers allowed)
UG_HISTORY_WINDOW	Number of days used to compute the history contribution.	3 (real numbers allowed)
UG_HISTORY_RATIO	Weight to compute the estimated execution time on a given resource.	0.25
<i>Rank (RA) Policy:</i> Prioritize resources based on their RANK (as defined in the job template)		
RA_WEIGHT	Weight for the policy.	0 (real numbers allowed)
<i>Failure Rate (FR) Policy:</i> Resources with persistent failures are banned		
FR_MAX_BANNED	The maximum time a resource is banned, in seconds. Use 0 TO DISABLE this policy.	3600
FR_BANNED_C	Exponential constant to compute banned time	650

4.6. Scheduler Configuration

GridWay uses an external and selectable scheduler module to schedule jobs. The following schedulers are distributed with GridWay:

- Built-in Scheduler (default), which implements the above policies.
- Round-robin/flood Scheduler. This is a simple scheduling algorithm. It maximizes the number of jobs submitted to the Grid. Available resources are flooded with user jobs in a round-robin fashion.



Important

The flood (user round-robin) scheduler is included as an example, and should not be used in production environments.

The schedulers are configured with the `DM_SCHED` option in the `gwd.conf` file, with the format:

```
DM_SCHED = <sched_name>:<path_to_sched>:[args]
```

where:

- **sched_name:** is a tag to further refer to this scheduler.
- **path_to_sched:** is the name of the Scheduler executable. Use an absolute path or include the Scheduler executable directory in the `PATH` variable (such directory is `$GW_LOCATION/bin` by default).
- **arg:** Additional arguments to be passed to the Scheduler executable.

4.6.1. Built-in Scheduler

By default, GridWay is configured to use the built-in policy engine described in the previous sections. If for any reason you need to recover this configuration, add the following line to `$GW_LOCATION/etc/gwd.conf`:

```
DM_SCHED = builtin:gw_sched:
```

Do not forget to adjust the scheduler policies to your needs by editing the `$GW_LOCATION/etc/sched.conf` file.

4.6.2. Flood Scheduler

To configure the round-robin/flood scheduler, first disable the built-in engine policy in the `$GW_LOCATION/etc/sched.conf` configuration file by adding the following line:

```
DISABLE = yes
```

Then add the following line to `$GW_LOCATION/etc/gwd.conf`:

```
DM_SCHED = flood:gw_flood_scheduler:-h 10 -u 30 -c 5 -s 15
```

where:

- **-h**: The max number of jobs that the scheduler submits to a given host. Default value is 10; use 0 to dispatch to each host as many jobs as possible.
- **-u**: The maximum number of simultaneous running jobs per user. Default value is 30; use 0 to dispatch as many jobs as possible.
- **-c**: Scheduling Chunk. Jobs of the same user are submitted in a round-robin fashion with the given chunk. Default value is 5.
- **-s**: Dispatch Chunk. The maximum number of jobs that will be dispatched each scheduling action. Default value is 15; use 0 to dispatch as many jobs as possible.

5. MAD Configuration Guide

GridWay uses several Middleware Access Drivers (MAD) to interface with different Grid services. The following MADs are part of the GridWay distribution:

- Execution Managers to interface with both pre-WS GRAM and WS GRAM services.
- Information Managers to interface with both MDS2 (MDS and GLUE schemas) and MDS4 services.
- Transfer Managers to interface with GridFTP servers.

These drivers are configured and selected via the GWD configuration interface described in [Section 3.2, “GridWay Daemon \(GWD\) Configuration”](#). Additionally you may need to configure your environment (see [Section 7, “Testing”](#)) in order to successfully load the MADs into the GWD core. To do so, you can also use global and per user environment configuration files (`gwr.c`).

5.1. MAD Environment Configuration

There is one global config file and per user configuration files that can be used to set environment variables for MADs. These files are standard shell scripts that are sourced into the MAD environment before it is loaded. It can be used, for example, to set the variable `X509_USER_PROXY` so you can have it located elsewhere instead of the standard place (`/tmp/x509_u<uid>`). Other variables can be set and you can even source other shell scripts, for instance, you can prepare another globus environment for MADs for some users, like this:

```
X509_USER_PROXY=$HOME/.globus/proxy.pem

GLOBUS_LOCATION=/opt/globus-4.0
. $GLOBUS_LOCATION/etc/globus-user-env.sh
```

The file for global MAD environment configuration is `$GW_LOCATION/etc/gridway/gwrc` and the user specific one is `$HOME/.gwrc`.

You have to take into account a couple of things:

- The global environment file is loaded *before* the user one, so the variables set by the user file take precedence over the global ones.
- The files are sourced so you need to export the variables to make them visible in the environment of the called MAD. Right now there is a mechanism so variables set as `VARIABLENAME=VALUE` are automatically exported (without spaces preceding the variable name). If you are sourcing other files or you put variables inside an indented block (for example, in an if statement) you have to explicitly export them. For example:

```
if [ -d /opt/globus-devel ]; then
    export GLOBUS_LOCATION=/opt/globus-devel
```

5.2. Execution Driver Configuration

The Execution Driver interfaces with Grid Execution Services and is responsible for low-level job execution and management. The GridWay distribution includes the following Execution MADs:

- Pre-WS GRAM (Globus Toolkit 2.4 and above)
- WS GRAM (Globus Toolkit 4.0)

Note that the use of these MADs requires a valid proxy.

Execution MADs are configured with the `EM_MAD` option in the `$GW_LOCATION/etc/gwd.conf` file, with the following format:

```
EM_MAD = <mad_name>:<path_to_mad>:<args>:<rs1|rs1_nsh|rs12>
```

where:

- **mad_name**: is a tag to further refer to this Execution Driver, and it is also useful for logging purposes.
- **path_to_mad**: is the name of the Execution Driver executable, which *must* be placed in the `$GW_LOCATION/bin` directory.

- **args:** Parameters passed to the mad when it is executed.
- **rsl|rsl_nsh|rsl2:** Selects the language that GWD will use to describe job requests. It can be *rsl* (intended to be used with pre-WS drivers), *rsl_nsh* (intended to be used with pre-WS drivers over resources with non-shared home directories, like in LCG) and *rsl2* (intended to be used with WS drivers).

For example, the following line will configure GridWay to use the Execution Driver **gw_em_mad_prews** using RSL syntax with name *prews*:

```
EM_MAD = prews:gw_em_mad_prews::rsl
```

To use WS-GRAM services, you can include the following line in your `$GW_LOCATION/etc/gwd.conf` file:

```
EM_MAD = ws:gw_em_mad_ws::rsl2
```



Note

You can simultaneously use as many Execution Drivers as you need (up to 10). So GridWay allows you to simultaneously use pre-WS and WS Globus Services.

5.2.1. Port configuration in WS EM MAD

Now it is possible to specify a different gatekeeper port than the standard one (8443) in the Web Service driver. The line to configure EM MADs in `gwd.conf` has changed so you can add parameters to it. The parameter to change the port is the `-p` followed by the port number. For example:

```
EM_MAD = osg_ws:gw_em_mad_ws:-p 9443:rsl2
```

This line tells the EM MAD to use port 9443 to connect to the GT4 Gatekeeper.

5.3. File Transfer Driver Configuration

The File Transfer Driver interfaces with Grid Data Management Services and is responsible for file staging, remote working directory set-up and remote host clean up. The GridWay distribution includes:

- GridFTP server (version 1.1.2 and above)
- Dummy Transfer driver (to be used with clusters without shared home)

The use of this driver requires a valid Proxy.

File Transfer Managers are configured with the `TM_MAD` option in the `gwd.conf` file, with the format:

```
TM_MAD = <mad_name>:<path_to_mad>:[arg]
```

where:

- **mad_name:** is a tag to further refer to this File Transfer Driver, and it is also useful for logging purposes.
- **path_to_mad:** is the name of the File Transfer Driver executable, which *must* be placed in the `$GW_LOCATION/bin` directory.
- **arg:** Additional arguments to be passed to the File Transfer executable.

To configure the Transfer Driver, add a line to `$GW_LOATION/etc/gwd.conf`, with the following format:

```
TM_MAD = <mad_name>:<path_to_mad>:[arguments]>
```

5.3.1. Configuring the GridFTP Transfer Driver

The GridFTP driver does not require any command line arguments. So to configure the driver, add the following line to `$GW_LOCATION/etc/gwd.conf`:

```
TM_MAD = gridftp:gw_tm_mad_ftp:
```

The name of the driver will be later used to specify the transfer mechanisms with Grid resource.

5.3.2. Configuring the Dummy Transfer Driver

The Dummy driver should be used with those resources (clusters) which do not have a shared home. In this case, transfer and execution are performed as follows:

- The Dummy Transfer MAD performs data movements from the cluster worker node and the client using a reverse server model.
- The `rsl_nsh` RSL generation function is used, which transfers the wrapper along with its stdout and stderr streams.
- The wrapper executing in the worker node automatically transfers `job.env` and input/output files from the client.

The following servers can be configured to access files on the client machine:

- *GASS Server*, started for each user.
- *GridFTP*, specified by its URL running on the GridWay server.

The Dummy driver behavior is specified with the following command line arguments:

- **-u <URL>**: URL of the GridFTP server.
- **-g**: Use a user GASS server to transfer files.

Sample configuration to use a GridFTP server:

```
TM_MAD = dummy:gw_tm_mad_dummy:-u gsiftp\://hostname
```

Important

You MUST escape the colon character in `gsiftp` URL. Also, `hostname` should be the host running the GridWay instance.

Sample configuration to use GASS servers:

```
TM_MAD = dummy:gw_tm_mad_dummy:-g
```

5.4. Information Driver Configuration

The Information Driver interfaces with Grid Monitoring Services and is responsible for host discovery and monitoring. The following Information Drivers are included in GridWay:

- Static host information data
- MDS2 with MDS schema (Globus Toolkit 2.4)

- MDS2 with GLUE schema (Globus Toolkit 2.4 and LCG middleware)
- MDS4 (Globus Toolkit 4.0)

To configure an Information Driver, add a line to `$GW_LOCATION/etc/gwd.conf`, with the following format:

```
IM_MAD = <mad_name>:<path_to_mad>:[args]:[nice]:<tm_mad_name>:<em_mad_name>
```

where:

- **mad_name**: is a tag to further refer to this Information Driver.
- **path_to_mad**: is the name of the Information Driver executable. Use an absolute path or include the Information Driver directory in the `PATH` variable (such directory is `$GW_LOCATION/bin` by default).
- **arg**: Additional arguments to be passed to the Information Driver executable.
- **nice**: Integer value that will be added to the rank calculated for the hosts managed by this Information Driver. So you can prioritize, at a coarse level, hosts from different Information Drivers (or Grids).
- **tm_mad_name**: File Transfer Driver to be used with the hosts managed by this Information Driver.
- **em_mad_name**: Execution Driver to be used with the hosts managed by this Information Driver.

For example, to configure GWD to access a MDS4 hierarchical information service:

```
IM_MAD = mds4:gw_im_mad_mds4:-s hydrus.dacya.ucm.es::gridftp:ws
```

All the Information Drivers provided with GridWay use a common interface to configure their operation mode. The arguments used by the Information Drivers are:

- **-s <server>**: The information server in a hierarchical configuration, i.e. MDS2 GIIS or MDS4 root IndexService.
- **-l <host list>**: A host list file to be used by GridWay, only relevant for static discovery and monitoring. See the Information Driver operation mode below (Relative path to `$GW_LOCATION`).
- **-b <base>**: The Virtual Organization name in the DN of the LDIF entries, i.e. the `Mds-Vo-name` attribute, only relevant for MDS2.
- **-f <filter>**: Additional requirements to be imposed on all the hosts managed by this Information Driver, in LDIF format.

These options allow you to configure your Information Drivers in the three operation modes, described below.

5.4.1. Static Discovery and Monitoring (SS mode)

In this mode, hosts are statically discovered by reading a host list file (note: each time it is read). Also the attributes of each host are read from files. Hint: Use this mode for testing purposes and not in a production environment. To configure a Information Driver in SS mode use the host list option, for example:

```
IM_MAD = static:gw_im_mad_static:-l examples/im/host.static::gridftp:ws
```

The host list file contains one host per line, with format:

```
FQDN    attribute_file
```

where:

- **FQDN**: is the Full Qualified Domain Name of the host.
- **attribute_file**: is the name of the file with the static attributes of this host. Relative to the `GW_LOCATION` directory.

For example (you can find this file, `host.list`, in `$GW_LOCATION/examples/im/`)

```
hydrus.dacya.ucm.es examples/im/hydrus.attr
draco.dacya.ucm.es examples/im/draco.attr
```

The *attribute_file* includes a *single line* with the host information and *other lines* with the information of each queue (one line per queue). Use the examples below as templates for your hosts.

Example of attribute file for a PBS cluster (you can find this file in `$GW_LOCATION/examples/im/`):

```
HOSTNAME="hydrus.dacya.ucm.es" ARCH="i686" OS_NAME="Linux" OS_VERSION="2.6.4"
CPU_MODEL="Intel(R) Pentium(R) 4 CPU 2" CPU_MHZ=2539 CPU_FREE=098 CPU_SMP=1
NODECOUNT=4 SIZE_MEM_MB=503 FREE_MEM_MB=188 SIZE_DISK_MB=55570
FREE_DISK_MB=39193 FORK_NAME="jobmanager-fork" LRMS_NAME="jobmanager-pbs"
LRMS_TYPE="pbs" QUEUE_NAME[0]="q4small" QUEUE_NODECOUNT[0]=1
QUEUE_FREENODECOUNT[0]=4 QUEUE_MAXTIME[0]=0 QUEUE_MAXCPUPTIME[0]=20
QUEUE_MAXCOUNT[0]=4 QUEUE_MAXRUNNINGJOBS[0]=0 QUEUE_MAXJOBSINQUEUE[0]=1
QUEUE_STATUS[0]="enabled" QUEUE_DISPATCHCTYPE[0]="batch"
QUEUE_NAME[1]="q4medium" QUEUE_NODECOUNT[1]=4 QUEUE_FREENODECOUNT[1]=4
QUEUE_MAXTIME[1]=0 QUEUE_MAXCPUPTIME[1]=120 QUEUE_MAXCOUNT[1]=4
QUEUE_MAXRUNNINGJOBS[1]=0 QUEUE_MAXJOBSINQUEUE[1]=1
QUEUE_STATUS[1]="enabled" QUEUE_DISPATCHCTYPE[1]="batch"
```

Example of attribute file for a Fork Desktop (you can find this file in `$GW_LOCATION/examples/im/`):

```
HOSTNAME="draco.dacya.ucm.es" ARCH="i686" OS_NAME="Linux" OS_VERSION="2.6-xen"
CPU_MODEL="Intel(R) Pentium(R) 4 CPU 3" CPU_MHZ=3201 CPU_FREE=185 CPU_SMP=2
NODECOUNT=2 SIZE_MEM_MB=431 FREE_MEM_MB=180 SIZE_DISK_MB=74312
FREE_DISK_MB=40461 FORK_NAME="jobmanager-fork" LRMS_NAME="jobmanager-fork"
LRMS_TYPE="fork" QUEUE_NAME[0]="default" QUEUE_NODECOUNT[0]=1
QUEUE_FREENODECOUNT[0]=1 QUEUE_MAXTIME[0]=0 QUEUE_MAXCPUPTIME[0]=0
QUEUE_MAXCOUNT[0]=0 QUEUE_MAXRUNNINGJOBS[0]=0 QUEUE_MAXJOBSINQUEUE[0]=0
QUEUE_STATUS[0]="0" QUEUE_DISPATCHCTYPE[0]="Immediate"
```

To use the WS version of these files just change `jobmanager-fork` with `Fork` and `jobmanager-pbs` with `PBS`.

5.4.2. Static Discovery and Dynamic Monitoring (SD mode)

Hosts are discovered by reading a host list file. However, the information of each host is gathered by querying its information service (GRIS in MDS2 or the `DefaultIndexService` in MDS4). Hint: Use this mode if the resources in your Grid does not vary too much, i.e. resource are not added or removed very often. To configure an Information Driver in SD mode, use the host list option, for example:

```
IM_MAD = glue:gw_im_mad_mds2_glue:-1 examples/im/host.list::gridftp:prews
```

In this case the host list file contains one host per line, in the format:

FQDN

...

FQDN

where:

- **FQDN**: is the Full Qualified Domain Name of the host.

For example (you can find this file in `$GW_LOCATION/examples/im/`)

`hydrus.dacya.ucm.es`

`ursa.dacya.ucm.es`

`draco.dacya.ucm.es`



Note

The information services of each host (GRIS or/and DefaultIndexServices) must be properly configured to use this mode.



Important

You can configure your IMs to work in a dynamic monitoring mode but get some static information from an attributes file (as described in the SS mode). This configuration is useful when you want to add some host attributes missing from the IndexService (like software availability, special hardware devices...). You can see a useful use of this mode in section [Troubleshooting](#).

5.4.3. Dynamic Discovery and Monitoring (DD mode)

In this mode, hosts are discovered and monitored by directly accessing the Grid Information Service. Hint: Use this mode if the resources in your Grid does vary too much, i.e. resource are added or removed very often. To configure a Information Driver in SD mode, use the server option, for example:

```
IM_MAD = mds4:gw_im_mad_mds4:-s hydrus.dacya.ucm.es::gridftp:ws
```



Note

A hierarchical information service (GIIS or/and DefaultIndexService) must be properly configured to use this mode.

If you are using an MDS2 information service you may need to specify the Virtual Organization name in the DN of the LDIF entries (*Mds-vo-name*) with the base option described above.



Note

You can simultaneously use as many Information Drivers as you need (up to 10). So GridWay allows you to simultaneously use MDS2 and MDS4 Services. You can also use resources from different Grids at the same time.



Note

You can mix SS, SD and DD modes in the same Information Driver.

5.4.4. Separate Storage and Computing Element

There is a way to specify a different machine to be used as gsiftp endpoint than the one that has the gatekeeper installed. This is useful when the CE machine does not have gsiftp server configured but there is another machine that works as a Storage Element. Right now, this information could be set statically but the rest of the information can be updated dynamically. To use this feature you have to create a file for each host you want to configure with extra information and another file with pairs of host and file name (as described above for the SS mode). The filename can be a full path or a relative path to `GW_LOCATION`. Then in the IM MAD you must specify the list file with `-l`, like this (in `gwd.conf`):

```
IM_MAD = mds4:gw_im_mad_mds4:-l etc/gridway/host.list:gridftp:ws
```

The file list should look like this:

```
wsgram-host1.domain.com etc/gridway/wsgram-host1.attr  
wsgram-host2.domain.com etc/gridway/wsgram-host2.attr
```

And the attributes file for each node should look like this:

```
SE_HOSTNAME="gridftp-host1.domain.com"
```

6. Deploying

GridWay is installed in the Globus directory when you issue the **make install**.

7. Testing

In order to test the GridWay installation, login as the `<gwadmin>` account and follow the steps listed below:

1. Set up the environment variables `GW_LOCATION` and `PATH`:

```
$ export GW_LOCATION=$GLOBUS_LOCATION  
$ export PATH=$PATH:$GW_LOCATION/bin
```

or

```
$ setenv GW_LOCATION $GLOBUS_LOCATION  
$ setenv PATH $PATH:$GW_LOCATION/bin
```

depending on the shell you are using.

2. Generate a valid proxy

```
$ grid-proxy-init  
Your identity: /O=Grid/OU=GRIDWAY/CN=GRIDWAY User  
Enter GRID pass phrase for this identity:  
Creating proxy ..... Done  
Your proxy is valid until: Mon Oct 29 03:29:17 2005
```

3. Show the GridWay license:

```
$ gwd -v
Copyright 2002-2007 GridWay Team, Distributed Systems Architecture
Group, Universidad Complutense de Madrid

GridWay 5.2 is distributed and licensed for use under the terms of the
Apache License, Version 2.0 (http://www.apache.org/licenses/LICENSE-2.0).
```

4. Start the GridWay daemon (GWD) in multiple users mode:

```
$ gwd -m
```

5. Check the connection to GWD:

```
$ gwps
USER          JID DM   EM   START   END     EXEC   XFER   EXIT NAME          HOST
$ gwghost
HID PRIO  OS                ARCH   MHZ  %CPU  MEM(F/T)   DISK(F/T)   N(U/F/T) LRMS
```

6. Stop GWD:

```
$ pkill gwd
```

7.1. GridWay Test Suite

GridWay is shipped with a test suite, available in the test directory. The test suite exercises different parts of GridWay, and can be used to track functionality bugs. However you need a working GridWay installation and testbed to execute the suite. Usage information is available with "gwtest -h". Tests can be performed individually (using the test id) or all together automatically.

Table 1.3. GridWay tests description.

Test #	Test Name	Description
1	Normal Execution (SINGLE)	Submits a single job and verifies it is executed correctly
2	Normal Execution (BULK)	Submits an array of 5 jobs and verifies that all of them are executed correctly.
3	Pre Wrapper	Verifies that GridWay is able to execute the pre wrapper functionality.
4	Prolog Fail (Fake Stdin) No Reschedule	Submits a single job that fails in the prolog state due to a wrong input file for stdin.
5	Prolog Fail (Fake Stdin) Reschedule	Equal to the previous one, but GridWay tries to reschedule the job up to 2 times.
6	Prolog Fail (Fake Input File) No Reschedule	Same as #4 with a wrong input file for the executable.
7	Prolog Fail (Fake Executable) No Reschedule	Same as #4 with a wrong filename for the executable.
8	Prolog Fail (Fake Executable) No Reschedule	Same as #4 with a wrong filename for the executable.
9	Prolog Fail (Fake Stdin) No Reschedule (BULK)	Same as #4 submitting an array of 5 jobs.
10	Execution Fail No Reschedule	Submits a single job designed to fail (bad exit code) and verifies the correctness of the final state (failed).
11	Execution Fail Reschedule	Same as #9 but GridWay tries to reschedule the job up to 2 times.
12	Hold Release	Submits a single job on hold, releases it and verifies that it is executed correctly.
13	Stop Resume	Submits a single job, stops it (in Wrapper state), resumes it and verifies that it is executed correctly.
14	Kill Sync	Submits a job and kills it using a synchronous signal.
15	Kill Async	Submits a job and kills it using an asynchronous signal.
16	Kill Hard	Submits a job and hard kills it.
17	Migrate	Submits a job and sends a migrate signal when it reaches the Wrapper state. It then verifies the correct execution of the job.
18	Checkpoint local	Submits a job which creates a checkpoint file and verifies the correct execution of the job and the correct creation of the checkpoint file.
19	Checkpoint remote server	Same as #17 but the checkpoint file is created in a remote gsiftp server.
20	Wait Timeout	Submits a job and waits for it repeatedly using short timeouts until it finishes correctly.
21	Wait Zerotimeout	Same as #19 but with zero timeout (effectively, an asynchronous wait).
22	Input Output files	Tests the different methods GridWay offers to stage files (both input and output).
23	Epilog Fail (Fake Output) No Reschedule	Submits a single job that fails in the epilog state due to a wrong output filename.

24	Epilog Fail (Fake Output) Resched- ule	Same as #22 but GridWay tries to reschedule the job up to 2 times.
25	Epilog Fail (Fake Output) No Res- chedule (BULK)	Same as #22 but submitting an array of 5 jobs.

8. Security considerations

Access authorization to the GridWay server is done based on the Unix identity of the user (accessing GridWay directly or through a Web Services GRAM, as in [GridGateWay⁷](#)). Hence, security in GridWay has the same implications as the Unix accounts of their users.

Also, GridWay uses proxy certificates to use Globus services, so the security implications of managing certificates also must be taken into account

9. Troubleshooting

If you are having problems starting GWD, check the errors listed below. Also try checking the daemon logs, see [Section 3.5, “Logging”](#).

Lock file exists

GridWay finishes with the following message when you try to start it:

```
Error! Lock file <path_to_GridWay>/var/.lock exists.
```

Be sure that no other GWD is running, then remove the lock file and try again.

Error in MAD initialization

GridWay finishes with the following message, when you try to start it:

```
Error in Execution MAD prews initialization, exiting. Check path
you have a valid proxy...
```

Check that you have generated a valid proxy (for example with the `grid-proxy-info` command). Also, check that the directory `$GW_LOCATION/bin` is in your path, and the executable name of all the MADs is defined in `gwd.conf`.

Error contacting GWD

Client commands, like **gwps**, finish with the message:

```
connect(): Connection refused
Could not connect to gwd
```

Be sure that GWD is running (ex. `pgrep -l gwd`). If it is running, check that you can connect to GWD (ex. `telnet `cat $GW_LOCATION/var/gwd.port``)

⁷ <http://www.grid4utility.org>

Chapter 2. GT 4.0.5+ Component Fact Sheet: GridWay (Contribution)

1. Brief component overview

Starting with version 4.0.5, GT includes the GridWay Metascheduler, which enables large-scale, reliable and efficient sharing of computing resources (clusters, computing farms, servers, supercomputers...), managed by different LRM (Local Resource Management) systems, such as PBS, SGE, LSF, Condor..., within a single organization (enterprise grid) or scattered across several administrative domains (partner or supply-chain grid).

There exist a number of commercial and open source workload management and scheduling systems available today, each one suitable for different underlying computer infrastructures and execution profiles. GridWay stands out from other metascheduling systems because it has been specifically designed to work on top of Globus services, offering the highest functionality, quality of service and reliability on this kind of infrastructure, namely:

- **For project and infrastructure directors:** GridWay is an open-source community project, adhering to Globus philosophy and guidelines for collaborative development.
- **For system integrators:** GridWay is highly modular, allowing adaptation to different grid infrastructures, and supports several OGF standards.
- **For system managers:** GridWay provides a scheduling framework similar to that found on local LRM systems, supporting resource accounting and the definition of state-of-the-art scheduling policies.
- **For application developers:** GridWay implements the OGF standard DRMAA API (C and JAVA bindings), assuring compatibility between applications with LRM systems that implement the standard, such as SGE, Condor, Torque...etc.
- **For end users:** GridWay provides an LRM-like CLI for submitting, monitoring, synchronizing and controlling jobs, that could be described using the OGF standard JSDL.

GridWay gives end users, application developers and managers of Globus infrastructures a scheduling functionality similar to that found on local DRM systems.

2. Summary of features

Advanced Scheduling Capabilities GridWay implements several state-of-the-art Grid-aware scheduling policies, comprising *job prioritization* policies (fixed priority, urgency, share, deadline and waiting-time) and *resource prioritization* policies (fixed priority, usage, failure and rank).

These policies are combined with:

- *Adaptive Scheduling*, to periodically adapt the schedule considering applications' demands and Grid resource characteristics.
- *Adaptive Execution* to migrate running applications in terms of resource availability, capacity or cost, and new application requirements or preferences.

Transparent Grid Access	GridWay interfaces infrastructures with different middleware stacks. With GridWay, users can access heterogeneous resources in a transparent way. For example, it can access resources configured with both GRAM pre web services and GRAM web services. It also permits the use of different grids with different software stacks, for example, Teragrid with Globus and EGEE with gLite.
Flexible Deployment Capabilities	<p>GridWay supports multiple-user operation mode, and does not require additional middleware installation (apart from standard Globus services). Globus installation is not required in each end-user system.</p> <p>GridWay allows different Grid deployment strategies, like Enterprise Grids, Partner Grids or Utility Grids.</p>
Different Application Profiles	<p>GridWay executes different Grid application profiles:</p> <ul style="list-style-type: none"> • Array (Bulk) jobs, for parameter sweep applications • DAG Workflows • Single-site MPI applications
Fault Detection and Recovery	<p>GridWay is able to detect several problems that can occur when executing a remote job. It also implements mechanisms that make the execution more reliable. It can detect a remote system crash, a job failure (via the job exit code) or even a network disconnection (using the polling mechanism) and migrate the problematic job to another resource.</p> <p>GridWay also performs periodic saves of its state in order to recover from local failure.</p>
Reporting and Accounting	GridWay provides detailed statistics of Grid usage. In this way, the Grid administrator can properly plan usage policies and forecast workload. In addition, these statistics can be used by the scheduler to predict (per user) Grid resource response time.
Standard Compliance	GridWay is an open-source project, flexible and completely based on standards to leverage its usability and interoperability. For example, users can describe their jobs using JSDL. Similarly, programmers can build grid enabled applications using the DRMAA standard.
User Interface	GridWay provides end-users with a familiar environment similar to that found on classical LRM systems. So GridWay CLI eases the adoption of Grid technologies.

3. Backward compatibility summary

GridWay 5.2.1 is a new component in GT starting with this release. The following information regards compatibility with its previous version, 5.2:

API changes since GridWay 5.2:

- None

CLI changes since GridWay 5.2:

- None

Configuration interface changes since GridWay 5.2:

- Arguments can be specified for the EM and TM drivers. Beware that `gwd.conf` configuration files from previous versions of GridWay **will not work with version 5.2.1**

4. Technology dependencies

GridWay uses different Globus services to perform the tasks of information gathering, job execution and data transfer.

- GridWay depends on the following GT components for job execution:
 - GRAM: GridWay can interface with both GRAM 2 (pre web services) and GRAM 4 (web services)
- GridWay depends on the following GT components for data staging:
 - RFT
 - GridFTP
- GridWay depends on the following GT component for information gathering:
 - MDS
- GridWay relies on the Globus basic security infrastructure for authentication and authorization. On top of that, GridWay can use other Globus services and components to complement this infrastructure:
 - Delegation Service
 - MyProxy

5. Tested platforms

Tested platforms for GridWay:

- GridWay builds successfully for the following platforms:
 - Linux
 - Tru64
 - Mac OS X
 - Solaris
 - Aix

In addition, GridWay has been tested with the major Grid infrastructures. Click the following links to find more information on how to use GridWay with [EGEE¹](http://www.gridway.org/documentation/stable/egeehowto/), [TeraGrid²](http://www.gridway.org/documentation/stable/tghowto/) and [OSG³](http://www.gridway.org/documentation/stable/osghowto/).

¹ <http://www.gridway.org/documentation/stable/egeehowto/>

² <http://www.gridway.org/documentation/stable/tghowto/>

³ <http://www.gridway.org/documentation/stable/osghowto/>

6. Associated standards

GridWay is an open project, flexible and completely based on standards to leverage its usability and interoperability. GridWay supports several standards developed by the Open Grid Forum,⁴ namely:

Distributed Resource Management
Application API Working Group
(DRMAA-WG)

The DRMAA working group⁵ has developed an API specification for the submission and control of jobs to one or more Distributed Resource Management (DRM) systems. The goal is to facilitate the direct interfacing of applications to today's DRM systems by application's builders, portal builders, and Independent Software Vendors (ISVs). The Distributed Resource Management Application API (DRMAA) provides a generalized API to distributed resource management systems (DRMSs) in order to facilitate integration of application programs.

The scope of DRMAA is limited to job submission, job monitoring and control, and retrieval of the finished job status. DRMAA provides application developers and distributed resource management builders with a programming model that enables the development of distributed applications tightly coupled to an underlying DRMS. DRMAA preserves flexibility and choice in system design.

GridWay provides support for the DRMAA API (JAVA and C bindings) to develop distributed applications. OGF document GFD.104: GridWay DRMAA 1.0 Implementation - Experience Report⁶ provides details of the implementation.

Job Submission Description Language (JSDL-WG)

The goal of the JSDL working group⁷ is to produce a language that describes the requirements of jobs for submission to Grids. JSDL 1.0, published as OGF recommendation GFD-R-P.056⁸ is an XML-based language that focuses mainly on computational jobs. The JSDL-WG is working on extending this language to address a wider class of jobs, including Web service invocations.

Last GridWay release provides support for JSDL, the POSIX Application profile and HTC Profile schemas can be used to describe jobs.

7. For More Information

Click here⁹ for more information about this component.

⁴ <http://www.ogf.org>

⁵ <http://drmaa.org/wiki/>

⁶ <http://www.ogf.org/documents/GFD.104.pdf>

⁷ <http://forge.gridforum.org/sf/projects/jsdl-wg>

⁸ <http://www.ggf.org/documents/GFD.56.pdf>

⁹ <http://www.gridway.org>

Chapter 3. GT 4.0.8 Incremental Release Notes: GridWay

1. Introduction

These release notes are for the incremental release 4.0.8. It includes a summary of changes since 4.0.7, bug fixes since 4.0.7 and any known problems that still exist at the time of the 4.0.8 release. This page is in addition to the top-level 4.0.8 release notes at <http://www.globus.org/toolkit/releasenotes/4.0.8>.

GridWay was introduced in a previous release, 4.0.5. For those release notes (including feature summary, technology dependencies, etc) go to the [GridWay 4.0.5 Release Notes](#)¹.

2. Changes Summary

GridWay version shipped with GT4.0.8 is based on GridWay 5.2 and no changes have been made since 4.0.7

3. Bug Fixes

No bugs have been fixed since the previous release.

4. Known Problems

The following is a list of all of the bugs known at the time for the 4.0.8 release:

- [Bug 5592](#):² gwd blocks when misconfigured sudo
- [Bug 5724](#):³ Environment for Non-Wrapper based jobs
- [Bug 5308](#):⁴ gwd doesn't recover AIDs and TIDs
- [Bug 5641](#):⁵ Sudoers configuration not allowing sudo to be run from a program
- [Bug 5592](#):⁶ gwd blocks when misconfigured sudo
- [Bug 5724](#):⁷ Environment for Non-Wrapper based jobs
- [Bug 5734](#):⁸ Documentation doesn't reference the new DRMAA 1.0 binding
- [Bug 5898](#):⁹ JSDL-Support needs Java 1.5+

¹ http://www.globus.org/toolkit/docs/4.0/contributions/gridway/GridWay_Release_Notes_405.html

² http://bugzilla.globus.org/globus/show_bug.cgi?id=5592

³ http://bugzilla.globus.org/globus/show_bug.cgi?id=5724

⁴ http://bugzilla.globus.org/globus/show_bug.cgi?id=5308

⁵ http://bugzilla.globus.org/globus/show_bug.cgi?id=5641

⁶ http://bugzilla.globus.org/globus/show_bug.cgi?id=5592

⁷ http://bugzilla.globus.org/globus/show_bug.cgi?id=5724

⁸ http://bugzilla.globus.org/globus/show_bug.cgi?id=5734

⁹ http://bugzilla.globus.org/globus/show_bug.cgi?id=5898

5. For More Information

Click [here](#)¹⁰ for more information about this component.

¹⁰ <http://www.gridway.org>

Chapter 4. GT 4.0.7 Incremental Release Notes: GridWay

1. Introduction

These release notes are for the incremental release 4.0.7. It includes a summary of changes since 4.0.6, bug fixes since 4.0.6 and any known problems that still exist at the time of the 4.0.7 release. This page is in addition to the top-level 4.0.7 release notes at <http://www.globus.org/toolkit/releasenotes/4.0.7>.

GridWay was introduced in a previous release, 4.0.5. For those release notes (including feature summary, technology dependencies, etc) go to the [GridWay 4.0.5 Release Notes](#)¹.

2. Changes Summary

GridWay version shipped with GT4.0.7 is based on GridWay 5.2 as in the last version so just bugfixes are relevant between this version and the one shipped with 4.0.6

3. Bug Fixes

Here is the list of bugs fixed in Globus GridWay 4.0.7, follow the links for a description of each bug:

- [Bug 5850](#):² Bug in gwhost -f

4. Known Problems

The following is a list of all of the bugs known at the time for the 4.0.7 release:

- [Bug 5592](#):³ gwd blocks when misconfigured sudo
- [Bug 5724](#):⁴ Environment for Non-Wrapper based jobs
- [Bug 5308](#):⁵ gwd doesn't recover AIDs and TIDs
- [Bug 5641](#):⁶ Sudoers configuration not allowing sudo to be run from a program
- [Bug 5592](#):⁷ gwd blocks when misconfigured sudo
- [Bug 5724](#):⁸ Environment for Non-Wrapper based jobs
- [Bug 5734](#):⁹ Documentation doesn't reference the new DRMAA 1.0 binding

¹ http://www.globus.org/toolkit/docs/4.0/contributions/gridway/GridWay_Release_Notes_405.html

² http://bugzilla.globus.org/globus/show_bug.cgi?id=5850

³ http://bugzilla.globus.org/globus/show_bug.cgi?id=5592

⁴ http://bugzilla.globus.org/globus/show_bug.cgi?id=5724

⁵ http://bugzilla.globus.org/globus/show_bug.cgi?id=5308

⁶ http://bugzilla.globus.org/globus/show_bug.cgi?id=5641

⁷ http://bugzilla.globus.org/globus/show_bug.cgi?id=5592

⁸ http://bugzilla.globus.org/globus/show_bug.cgi?id=5724

⁹ http://bugzilla.globus.org/globus/show_bug.cgi?id=5734

- [Bug 5898](#).¹⁰ JSDL-Support needs Java 1.5+

5. For More Information

Click [here](#)¹¹ for more information about this component.

¹⁰ http://bugzilla.globus.org/globus/show_bug.cgi?id=5898

¹¹ <http://www.gridway.org>

Chapter 5. GT 4.0.6 Incremental Release Notes: GridWay

1. Introduction

These release notes are for the incremental release 4.0.6. It includes a summary of changes since 4.0.5, bug fixes since 4.0.5 and any known problems that still exist at the time of the 4.0.6 release. This page is in addition to the top-level 4.0.6 release notes at <http://www.globus.org/toolkit/releasenotes/4.0.6>.

GridWay was introduced in the previous release, 4.0.5. For those release notes (including feature summary, technology dependencies, etc) go to the [GridWay 4.0.5 Release Notes](#)¹.

2. Changes Summary

GridWay 5.2.3 mainly includes fixes for bugs reported by users and some performance improvements.

New to GridWay 5.2.3:

New Information MAD	Bug 5231 ² was reopened and, as a result, a new Information MAD for MDS4 has been developed. Since this new MAD has significant performance improvements, its use is highly recommended.
Better integration with GridGateWay	Some minor bugs has been solved to better integrate GridWay with the GridGateWay. In addition some bugs in the configure script have been solved. Therefore, this release is intended for those affected by the bugs below, and for the GridGateWay users.

3. Bug Fixes

Here is the list of bugs fixed in GridWay 5.2.3, follow the links for a description of each bug:

- [5400](#)³
- [5401](#)⁴
- [5412](#)⁵
- [5413](#)⁶
- [5416](#)⁷
- [5420](#)⁸

¹ http://www.globus.org/toolkit/docs/4.0/contributions/gridway/GridWay_Release_Notes_405.html

² http://bugzilla.mcs.anl.gov/globus/show_bug.cgi?id=5251

³ http://bugzilla.globus.org/globus/show_bug.cgi?id=5400

⁴ http://bugzilla.globus.org/globus/show_bug.cgi?id=5401

⁵ http://bugzilla.globus.org/globus/show_bug.cgi?id=5412

⁶ http://bugzilla.globus.org/globus/show_bug.cgi?id=5413

⁷ http://bugzilla.globus.org/globus/show_bug.cgi?id=5416

⁸ http://bugzilla.globus.org/globus/show_bug.cgi?id=5420

- [5422](#)⁹
- [4563](#)¹⁰
- [4789](#)¹¹
- [5404](#)¹²
- [5411](#)¹³
- [5413](#)¹⁴
- [5438](#)¹⁵
- [5448](#)¹⁶
- [5449](#)¹⁷
- [5458](#)¹⁸
- [5461](#)¹⁹
- [5462](#)²⁰
- [5476](#)²¹
- [5497](#)²²
- [5498](#)²³
- [5561](#)²⁴
- [5579](#)²⁵

4. Known Problems

The following is a list of all of the bugs known at the time for the 4.0.6 release:

- [Bug 5592](#).²⁶ gwd blocks when misconfigured sudo

⁹ http://bugzilla.globus.org/globus/show_bug.cgi?id=5422

¹⁰ http://bugzilla.globus.org/globus/show_bug.cgi?id=4563

¹¹ http://bugzilla.globus.org/globus/show_bug.cgi?id=4789

¹² http://bugzilla.globus.org/globus/show_bug.cgi?id=5404

¹³ http://bugzilla.globus.org/globus/show_bug.cgi?id=5411

¹⁴ http://bugzilla.globus.org/globus/show_bug.cgi?id=5413

¹⁵ http://bugzilla.globus.org/globus/show_bug.cgi?id=5438

¹⁶ http://bugzilla.globus.org/globus/show_bug.cgi?id=5448

¹⁷ http://bugzilla.globus.org/globus/show_bug.cgi?id=5449

¹⁸ http://bugzilla.globus.org/globus/show_bug.cgi?id=5458

¹⁹ http://bugzilla.globus.org/globus/show_bug.cgi?id=5461

²⁰ http://bugzilla.globus.org/globus/show_bug.cgi?id=5462

²¹ http://bugzilla.globus.org/globus/show_bug.cgi?id=5476

²² http://bugzilla.globus.org/globus/show_bug.cgi?id=5497

²³ http://bugzilla.globus.org/globus/show_bug.cgi?id=5498

²⁴ http://bugzilla.globus.org/globus/show_bug.cgi?id=5561

²⁵ http://bugzilla.globus.org/globus/show_bug.cgi?id=5579

²⁶ http://bugzilla.globus.org/globus/show_bug.cgi?id=5592

- [Bug 5724](#).²⁷ Environment for Non-Wrapper based jobs
- [Bug 5308](#).²⁸ gwd doesn't recover AIDs and TIDs

5. For More Information

Click [here](#)²⁹ for more information about this component.

²⁷ http://bugzilla.globus.org/globus/show_bug.cgi?id=5724

²⁸ http://bugzilla.globus.org/globus/show_bug.cgi?id=5308

²⁹ <http://www.gridway.org>

Chapter 6. GT 4.0.5 Incremental Release Notes: GridWay

1. Component Overview

Starting with version 4.0.5, GT includes the GridWay Metascheduler, which enables large-scale, reliable and efficient sharing of computing resources (clusters, computing farms, servers, supercomputers...), managed by different LRM (Local Resource Management) systems, such as PBS, SGE, LSF, Condor..., within a single organization (enterprise grid) or scattered across several administrative domains (partner or supply-chain grid).

There exist a number of commercial and open source workload management and scheduling systems available today, each one suitable for different underlying computer infrastructures and execution profiles. GridWay stands out from other metascheduling systems because it has been specifically designed to work on top of Globus services, offering the highest functionality, quality of service and reliability on this kind of infrastructure, namely:

- **For project and infrastructure directors:** GridWay is an open-source community project, adhering to Globus philosophy and guidelines for collaborative development.
- **For system integrators:** GridWay is highly modular, allowing adaptation to different grid infrastructures, and supports several OGF standards.
- **For system managers:** GridWay provides a scheduling framework similar to that found on local LRM systems, supporting resource accounting and the definition of state-of-the-art scheduling policies.
- **For application developers:** GridWay implements the OGF standard DRMAA API (C and JAVA bindings), assuring compatibility between applications with LRM systems that implement the standard, such as SGE, Condor, Torque...etc.
- **For end users:** GridWay provides an LRM-like CLI for submitting, monitoring, synchronizing and controlling jobs, that could be described using the OGF standard JSDL.

GridWay gives end users, application developers and managers of Globus infrastructures a scheduling functionality similar to that found on local DRM systems.

2. Feature Summary

Advanced Scheduling Capabilities GridWay implements several state-of-the-art Grid-aware scheduling policies, comprising *job prioritization* policies (fixed priority, urgency, share, deadline and waiting-time) and *resource prioritization* policies (fixed priority, usage, failure and rank).

These policies are combined with:

- *Adaptive Scheduling*, to periodically adapt the schedule considering applications' demands and Grid resource characteristics.
- *Adaptive Execution* to migrate running applications in terms of resource availability, capacity or cost, and new application requirements or preferences.

Transparent Grid Access	GridWay interfaces infrastructures with different middleware stacks. With GridWay, users can access heterogeneous resources in a transparent way. For example, it can access resources configured with both GRAM pre web services and GRAM web services. It also permits the use of different grids with different software stacks, for example, Teragrid with Globus and EGEE with gLite.
Flexible Deployment Capabilities	<p>GridWay supports multiple-user operation mode, and does not require additional middleware installation (apart from standard Globus services). Globus installation is not required in each end-user system.</p> <p>GridWay allows different Grid deployment strategies, like Enterprise Grids, Partner Grids or Utility Grids.</p>
Different Application Profiles	<p>GridWay executes different Grid application profiles:</p> <ul style="list-style-type: none">• Array (Bulk) jobs, for parameter sweep applications• DAG Workflows• Single-site MPI applications
Fault Detection and Recovery	<p>GridWay is able to detect several problems that can occur when executing a remote job. It also implements mechanisms that make the execution more reliable. It can detect a remote system crash, a job failure (via the job exit code) or even a network disconnection (using the polling mechanism) and migrate the problematic job to another resource.</p> <p>GridWay also performs periodic saves of its state in order to recover from local failure.</p>
Reporting and Accounting	GridWay provides detailed statistics of Grid usage. In this way, the Grid administrator can properly plan usage policies and forecast workload. In addition, these statistics can be used by the scheduler to predict (per user) Grid resource response time.
Standard Compliance	GridWay is an open-source project, flexible and completely based on standards to leverage its usability and interoperability. For example, users can describe their jobs using JSDL. Similarly, programmers can build grid enabled applications using the DRMAA standard.
User Interface	GridWay provides end-users with a familiar environment similar to that found on classical LRM systems. So GridWay CLI eases the adoption of Grid technologies.

3. Technology Dependencies

GridWay uses different Globus services to perform the tasks of information gathering, job execution and data transfer.

- GridWay depends on the following GT components for job execution:
 - GRAM: GridWay can interface with both GRAM 2 (pre web services) and GRAM 4 (web services)
- GridWay depends on the following GT components for data staging:
 - RFT

- GridFTP
- GridWay depends on the following GT component for information gathering:
 - MDS
- GridWay relies on the Globus basic security infrastructure for authentication and authorization. On top of that, GridWay can use other Globus services and components to complement this infrastructure:
 - Delegation Service
 - MyProxy

4. Supported Platforms

Tested platforms for GridWay:

- GridWay builds successfully for the following platforms:
 - Linux
 - Tru64
 - Mac OS X
 - Solaris
 - Aix

In addition, GridWay has been tested with the major Grid infrastructures. Click the following links to find more information on how to use GridWay with [EGEE¹](#), [TeraGrid²](#) and [OSG³](#).

5. Backward Compatibility Summary

GridWay 5.2.1 is a new component in GT starting with this release. The following information regards compatibility with its previous version, 5.2:

API changes since GridWay 5.2:

- None

CLI changes since GridWay 5.2:

- None

Configuration interface changes since GridWay 5.2:

- Arguments can be specified for the EM and TM drivers. Beware that `gwd.conf` configuration files from previous versions of GridWay **will not work with version 5.2.1**

¹ <http://www.gridway.org/documentation/stable/egeehowto>

² <http://www.gridway.org/documentation/stable/tghowto/>

³ <http://www.gridway.org/documentation/stable/osghowto/>

6. Changes Summary

Since GridWay 5.2, development activities have been focused on easing the integration of GridWay with the major Grid infrastructures: EGEE, TeraGrid and OSG. As a result, the flexibility of GridWay has been considerably improved. In addition, an important effort has been made to improve the reliability of GridWay's core.

Also, GridWay 5.2.1 is the first release shipped with the Globus Toolkit. Therefore, some modifications have been introduced to build and install GridWay in a GT tree: the directory layout has been slightly changed, and support to build GridWay with GPT has been added.

New to GridWay 5.2.1:

Integration with major Grid Infrastructures	GridWay 5.2.1 can be easily integrated with all the major Grid infrastructures. Its functionality has been extended to operate different Grid deployments, which includes support for different execution/transfer schemes, information models and service configurations.
Improved Reliability	Previous GridWay releases do not handle MAD crashes. GridWay 5.2.1 will reload a MAD process whenever a MAD is killed or crashed.
New Information model	The dynamic information of a host gathered from the Grid Information server can be mixed with custom variables defined by the GridWay administrator. This functionality is very useful when you need to extend the information scheme but have no access to the Grid server. For example, you can use this new feature to add software or license attributes to grid resources that can subsequently be used for resource requirement expressions.
Configuration Interface for MADs	Middleware Access Drivers need some environment variables to work properly. Usually, sudo must be configured to preserve these variables (e.g. GLOBUS_LOCATION or GW_LOCATION). GridWay 5.2.1 has a new configuration interface to ease this configuration. In this way, global (and per user) environment variables can be defined for MAD execution. This new feature allows GridWay to work with delegated credentials when configured with a GRAM interface.
Flexible definition of file transfer servers	A GridFTP server, different from the GRAM server, can be defined for file staging. The storage server must be defined (SE_HOSTNAME, attribute) for those resources operated on in this way. This can be done either by modifying the IM's MAD or using the new static-dynamic information model.
Support for JSDL HPC profile	Now job templates can be also defined using the OGF standard JSDL HPC profile.

7. Bug Fixes

- [Bug 5090](http://bugzilla.globus.org/globus/show_bug.cgi?id=5090):⁴ Autotools broken when --disable-jsdl and --disable-ws flags enabled
- [Bug 5113](http://bugzilla.globus.org/globus/show_bug.cgi?id=5113):⁵ GridWay does not execute custom monitor scripts
- [Bug 5333](http://bugzilla.globus.org/globus/show_bug.cgi?id=5333):⁶ An alternative wrapper can not be specified with relative paths

⁴ http://bugzilla.globus.org/globus/show_bug.cgi?id=5090

⁵ http://bugzilla.globus.org/globus/show_bug.cgi?id=5113

⁶ http://bugzilla.globus.org/globus/show_bug.cgi?id=5333

- [Bug 5231](#):⁷ Possible errors for the deadline policy
- [Bug 5114](#):⁸ Host information is not updated when a parse error occurs
- [Bug 5202](#):⁹ Buffer overflow in gwhost
- [Bug 5302](#):¹⁰ Scheduler dies when a user is reload
- [Bug 4922](#):¹¹ Middleware drivers are not reloaded when they crash
- [Bug 5251](#):¹² WS IM_MAD launches one java process per host for monitoring
- [Bug 5320](#):¹³ Control the number of active IM MADs

8. Known Problems

The following is a list of all of the bugs known at the time for the 4.0.5 release:

- [Bug 5308](#):¹⁴ gwd doesn't recover AIDs and TIDs

9. For More Information

Click [here](#)¹⁵ for more information about this component.

⁷ http://bugzilla.globus.org/globus/show_bug.cgi?id=5231

⁸ http://bugzilla.globus.org/globus/show_bug.cgi?id=5114

⁹ http://bugzilla.globus.org/globus/show_bug.cgi?id=5202

¹⁰ http://bugzilla.globus.org/globus/show_bug.cgi?id=5302

¹¹ http://bugzilla.globus.org/globus/show_bug.cgi?id=4922

¹² http://bugzilla.globus.org/globus/show_bug.cgi?id=5251

¹³ http://bugzilla.globus.org/globus/show_bug.cgi?id=5320

¹⁴ http://bugzilla.globus.org/globus/show_bug.cgi?id=5308

¹⁵ <http://www.gridway.org>