

# **GT 4.2.1 Component Guide to Public Interfaces: GridWay**

---

## GT 4.2.1 Component Guide to Public Interfaces: GridWay

Published May, 2008

Copyright © 2002-2008 GridWay Team, Distributed Systems Architecture Group, Universidad Complutense de Madrid (dsa-research.org).

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0><sup>1</sup>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Any academic report, publication, or other academic disclosure of results obtained with the GridWay Metascheduler will acknowledge GridWay's use by an appropriate citation to relevant papers by GridWay team members.

---

<sup>1</sup> <http://www.apache.org/licenses/LICENSE-2.0>

---

---

# Table of Contents

1. APIs .....	1
1. DRMAA bindings for C and Java .....	1
2. DRMAA bindings for scripting languages .....	1
2. Non-WSDL protocols .....	4
1. Information manager MAD .....	4
2. Execution manager MAD .....	6
3. Transfer manager MAD .....	6
4. Dispatch manager Scheduler .....	7
I. GridWay Commands .....	9
Job and Array Job submission Command .....	10
DAG Job submission Command .....	11
Job Monitoring Command .....	12
Job History Command .....	14
Host Monitoring Command .....	15
Job Control Command .....	17
Job Synchronization Command .....	18
User Monitoring Command .....	19
Accounting Command .....	20
JSDL To GridWay Job Template Parser Command .....	21
3. Configuration Guide .....	22
1. Core Configuration Guide .....	22
2. Scheduler Configuration Guide .....	26
3. MAD Configuration Guide .....	36
4. Integration Guides .....	43
4. Environment variable interface .....	44
1. Environment variables for GridWay .....	44
A. Errors .....	46

---

## List of Figures

3.1. Job Scheduling in GridWay .....	27
3.2. Job and resource prioritization policies in GridWay. ....	28
3.3. Dispatch priority of a job .....	30
3.4. Estimated execution time of a job on a resource .....	32
3.5. Banned time formula .....	32
3.6. Suitability priority of a resource .....	33

---

## List of Tables

1.1. Translation from C to Scripting language .....	2
2.1. Attributes that should be defined by the Information MADs. ....	5
3. Field options .....	13
4. Field information .....	14
5. Field information .....	15
6. Queue field information .....	16
7. Field information .....	19
8. Field information .....	20
3.1. GWD Configuration File Options. ....	24
3.2. Built-in Scheduler Configuration File Options. ....	34
A.1. Gridway Errors .....	46

---

# Chapter 1. APIs

## 1. DRMAA bindings for C and Java

The following links reference the API for the C and Java bindings of GridWay's implementation of DRMAA:

- [DRMAA C reference](#)<sup>1</sup>
- [DRMAA JAVA reference](#)<sup>2</sup>

## 2. DRMAA bindings for scripting languages.

Most functions in DRMAA C library use reference parameters to get results when calling them. Using scripting languages makes this way of getting information unfeasible. For example python strings are immutable so it is not possible to fill an empty string with the information needed from the extension. The way the functions are called are the same as Perl and Python SGE DRMAA bindings, reference variables are omitted and what functions return is an array of result code, reference variables and error string. For example in C this call:

```
result=drmaa_version(&major,&minor,error,DRMAA_ERROR_STRING_BUFFER-1);
```

is translated to ruby as:

```
(result, major, minor, error)=drmaa_version
```

---

<sup>1</sup> [http://www.gridway.org/documentation/stable/drmaa\\_c/](http://www.gridway.org/documentation/stable/drmaa_c/)

<sup>2</sup> [http://www.gridway.org/documentation/stable/drmaa\\_java/](http://www.gridway.org/documentation/stable/drmaa_java/)

**Table 1.1. Translation from C to Scripting language**

C	Scripting Language
result=drmaa_get_next_attr_name(values, &value, value_len)	(result, value)=drmaa_get_next_attr_name(values)
result=drmaa_get_next_attr_value(values, &value, value_len)	(result, value)=drmaa_get_next_attr_value(values)
result=drmaa_get_next_job_id(values, &value, value_len)	(result, value)=drmaa_get_next_job_id(values)
result=drmaa_get_num_attr_names(values, &size)	(result, size)=drmaa_get_num_attr_names(values)
result=drmaa_get_num_attr_values(values, &size)	(result, size)=drmaa_get_num_attr_values(values)
result=drmaa_get_num_job_ids(values, &size)	(result, size)=drmaa_get_num_job_ids(values)
drmaa_release_attr_names(values)	drmaa_release_attr_names(values)
drmaa_release_attr_values(values)	drmaa_release_attr_values(values)
drmaa_release_job_ids(values)	drmaa_release_job_ids(values)
result=drmaa_init(contact, error, error_len)	(result, error)=drmaa_init(nil)
result=drmaa_exit(error, error_len)	(result, error)=drmaa_exit()
result=drmaa_allocate_job_template(&jt, error, error_len)	(result, jt, error)=drmaa_allocate_job_template()
result=drmaa_delete_job_template(jt, error, error_len)	(result, error)=drmaa_delete_job_template(jt)
result=drmaa_set_attribute(jt, name, value, error, error_len)	(result, error)=drmaa_set_attribute(jt, name, value)
result=drmaa_get_attribute(jt, name, &value, error, error_len)	(result, value, error)=drmaa_get_attribute(jt, name)
result=drmaa_set_vector_attribute(jt, name, value, error, error_len)	(result, error)=drmaa_set_vector_attribute(jt, name, value)
result=drmaa_get_vector_attribute(jt, name, &values, error, error_len)	(result, values, error)=drmaa_get_vector_attribute(jt, name)
result=drmaa_get_attribute_names(&values, error, error_len)	(result, values, error)=drmaa_get_attribute_names()
result=drmaa_get_vector_attribute_names(&values, error, error_len)	(result, values, error)=drmaa_get_vector_attribute_names()
result=drmaa_run_job(job_id, job_id_len, jt, error, error_len)	(result, job_id, error)=drmaa_run_job(jt)
result=drmaa_run_bulk_jobs(&jobids, jt, start, end, incr, error, error_len)	(result, jobids, error)=drmaa_run_bulk_jobs(jt, start, end, incr)
result=drmaa_control(jobid, action, error, error_len)	(result, error)=drmaa_control(jobid, action)
result=drmaa_job_ps(job_id, &remote_ps, error, error_len)	(result, remote_ps, error)=drmaa_job_ps(job_id)
result=drmaa_synchronize(job_ids, timeout, dispose, error, error_len)	(result, error)=drmaa_synchronize(job_ids, timeout, dispose)
result=drmaa_wait(job_id, job_id_out, job_id_out_len, &stat, timeout, &rusage, error, error_len)	(result, job_id_out, stat, rusage, error)=drmaa_wait(job_id, timeout)
result=drmaa_wifexited(&exited, stat, error, error_len)	(result, exited, error)=drmaa_wifexited(stat)

C	Scripting Language
result=drmaa_wexitstatus(&exit_status, stat, error, error_len)	(result, exit_status, error)=drmaa_wexitstatus(stat)
result=drmaa_wifsignaled(&signaled, stat, error, error_len)	(result, signaled, error)=drmaa_wifsignaled(stat)
result=drmaa_wtermsig(signal, signal_len, stat, error, error_len)	(result, signal, error)=drmaa_wtermsig(stat)
result=drmaa_wcoredump(&core_dumped, stat, error, error_len)	(result, core_dumped, error)=drmaa_wcoredump(stat)
result=drmaa_wifaborted(&aborted, stat, error, error_len)	(result, aborted, error)=drmaa_wifaborted(stat)
error_string=drmaa_strerror(drmaa_errno)	error_string=drmaa_strerror(drmaa_errno)
result=drmaa_get_contact(contact, contact_len, error, error_len)	(result, contact, error)=drmaa_get_contact()
result=drmaa_version(&major, &minor, error, error_len)	(result, major, minor, error)=drmaa_version()
result=drmaa_get_DRM_system(drm_system, drm_system_len, error, error_len)	(result, drm_system, error)=drmaa_get_DRM_system()
result=drmaa_get_DRMAA_implementation(drmaa_impl, drmaa_impl_len, error, error_len)	(result, drmaa_impl, error)=drmaa_get_DRMAA_implementation()
str_status=drmaa_gw_strstatus(drmaa_state)	str_status=drmaa_gw_strstatus(drmaa_state)



## Note

In **drmaa\_init** call we have to pass a **NULL** argument as GridWay DRMAA library requires it. Here as an example I used **nil** as it is the ruby object describing **NULL** but in perl you have to use **undef** and in python **None**.

---

# Chapter 2. Non-WSDL protocols

## 1. Information manager MAD

In order to provide an abstraction with the monitoring and discovery middleware layer (or Grid Information System), GridWay uses a Middleware Access Driver (MAD) module to discover and monitor hosts. This module provides basic operations with the monitoring and discovery middleware.

The format to send a request to the Execution MAD, through its standard input, is:

```
OPERATION HID HOST -
```

Where:

- **OPERATION:** Can be one of the following:
  - **INIT:** Initializes the MAD.
  - **DISCOVER:** Discovers hosts.
  - **MONITOR:** Monitors a host.
  - **FINALIZE:** Finalizes the MAD.
- **HID:** If the operation is **MONITOR**, it is a host identifier, chosen by GridWay. Otherwise it is ignored.
- **HOST:** If the operation is **MONITOR** it specifies the host to monitor. Otherwise it is ignored.

On the other side, the format to receive a response from the MAD, through its standard output, is:

```
OPERATION HID RESULT INFO
```

Where:

- **OPERATION:** Is the operation specified in the request that originated the response.
- **HID:** It is the host identifier, as provided in the submission request.
- **RESULT:** It is the result of the operation. Could be **SUCCESS** or **FAILURE**.
- **INFO:** If **RESULT** is **FAILURE**, it contains the cause of failure. Otherwise, if **OPERATION** is **DISCOVER**, it contains a list of discovered host, or if **OPERATION** is **MONITOR**, it contains a list of host attributes.

**Table 2.1. Attributes that should be defined by the Information MADs.**

HOSTNAME	FQDN (Fully Qualified Domain Name) of the execution host (e.g. "hydrus.dacya.ucm.es")
ARCH	Architecture of the execution host (e.g. "i686", "alpha")
OS_NAME	Operating System name of the execution host (e.g. "Linux", "SL")
OS_VERSION	Operating System version of the execution host (e.g. "2.6.9-1.66", "3")
CPU_MODEL	CPU model of the execution host (e.g. "Intel(R) Pentium(R) 4 CPU 2", "PIV")
CPU_MHZ	CPU speed in MHz of the execution host
CPU_FREE	Percentage of free CPU of the execution host
CPU_SMP	CPU SMP size of the execution host
NODECOUNT	Total number of nodes of the execution host
SIZE_MEM_MB	Total memory size in MB of the execution host
FREE_MEM_MB	Free memory in MB of the execution hosts
SIZE_DISK_MB	Total disk space in MB of the execution hosts
FREE_DISK_MB	Free disk space in MB of the execution hosts
LRMS_NAME	Name of local DRM system (job manager) for execution, usually not fork (e.g. "jobmanager-pbs", "Pbs", "jobmanager-sge", "SGE")
LRMS_TYPE	Type of local DRM system for execution (e.g. "PBS", "SGE")
QUEUE_NAME[i]	Name of queue i (e.g. "default", "short", "dteam")
QUEUE_NODECOUNT[i]	Total node count of queue i
QUEUE_FREENODECOUNT[i]	Free node count of queue i
QUEUE_MAXTIME[i]	Maximum wall time of jobs in queue i
QUEUE_MAXCPU[i]	Maximum CPU time of jobs in queue i
QUEUE_MAXCOUNT[i]	Maximum count of jobs that can be submitted in one request to queue i
QUEUE_MAXRUNNINGJOBS[i]	Maximum number of running jobs in queue i
QUEUE_MAXJOBSINQUEUE[i]	Maximum number of queued jobs in queue i
QUEUE_DISPATCHTYPE[i]	Dispatch type of queue i (e.g. "batch", "immediate")
QUEUE_PRIORITY[i]	Priority of queue i
QUEUE_STATUS[i]	Status of queue i (e.g. "active", "production")

The information drivers interface to the grid information services to collect the resource attributes. These attributes can be used by the end-user to set requirement and rank expressions (job template), for filtering, prioritizing and selecting the candidate hosts. GridWay can simultaneously use as many Information drivers as needed. For example, GridWay allows you to simultaneously use MDS2 and MDS4 services, so you can also use resources from different Grids at the same time. Drivers for MDS 2 and MDS 4 provide the variables described in Table 2-1. However, the information manager is able to receive from the driver other parameters. The GridWay team has used other information parameters that could be very important to improve application efficiency (HTC apps) and for job migration: BANDWIDTH, LATENCY, SPEC\_INT, SPEC\_FLOAT...

## 2. Execution manager MAD

In order to provide an abstraction with the resource management middleware layer, GridWay uses a Middleware Access Driver (MAD) module to submit, control and monitor the execution of jobs. This module provides basic operations with the resource management middleware.

The format to send a request to the Execution MAD, through its standard input, is:

```
OPERATION JID HOST/JM RSL
```

Where:

- **OPERATION:** Can be one of the following:
  - **INIT:** Initializes the MAD.
  - **SUBMIT:** Submits a job.
  - **POLL:** Polls a job to obtain its state.
  - **CANCEL:** Cancels a job.
  - **FINALIZE:** Finalizes the MAD.
- **JID:** Is a job identifier, chosen by GridWay.
- **HOST:** If the operation is **SUBMIT**, it specifies the resource contact to submit the job. Otherwise it is ignored.
- **JM:** If the operation is **SUBMIT**, it specifies the job manager to submit the job. Otherwise it is ignored.
- **RSL:** If the operation is **SUBMIT**, it specifies the resource specification to submit the job. Otherwise it is ignored.

On the other side, the format to receive a response from the MAD, through its standard output, is:

```
OPERATION JID RESULT INFO
```

Where:

- **OPERATION:** Is the operation specified in the request that originated the response or **CALLBACK**, in the case of an asynchronous notification of a state change.
- **JID:** It is the job identifier, as provided in the submission request.
- **RESULT:** It is the result of the operation. Could be **SUCCESS** or **FAILURE**.
- **INFO:** If **RESULT** is **FAILURE**, it contains the cause of failure. Otherwise, if **OPERATION** is **POLL** or **CALLBACK**, it contains the state of the job.

## 3. Transfer manager MAD

In order to provide an abstraction with the file transfer management middleware layer, GridWay uses a Middleware Access Driver (MAD) module to transfer job files. This module provides basic operations with the file transfer middleware.

The format to send a request to the Transfer MAD, through its standard input, is:

OPERATION JID TID EXE\_MODE SRC\_URL DST\_URL

Where:

- OPERATION: Can be one of the following:
  - INIT: Initializes the MAD, JID should be max number of jobs.
  - START: Init transfer associated with job JID
  - END: Finish transfer associated with job JID
  - MKDIR: Creates directory SRC\_URL
  - RMDIR: Removes directory SRC\_URL
  - CP: start a copy of SRC\_URL to DST\_URL, with identification TID, and associated with job JID.
  - FINALIZE: Finalizes the MAD.
- JID: Is a job identifier, chosen by GridWay.
- TID: Transfer identifier, only relevant for command CP.
- EXE\_MODE: If equal to 'X' file will be given execution permissions, only relevant for command CP.

On the other side, the format to receive a response from the MAD, through its standard output, is:

OPERATION JID TID RESULT INFO

Where:

- OPERATION: Is the operation specified in the request that originated the response or CALLBACK, in the case of an asynchronous notification of a state change.
- JID: It is the job identifier, as provided in the START request.
- TID: It is the transfer identifier, as provided in the CP request.
- RESULT: It is the result of the operation. Could be SUCCESS or FAILURE.
- INFO: If RESULT is FAILURE, it contains the cause of failure.

## 4. Dispatch manager Scheduler

In order to decouple the scheduling process, GridWay uses a Scheduler module to schedule jobs.

The format to send a scheduling request to the Scheduler, through its standard input, is:

SCHEDULE - - -

On the other side, the format to receive a response from the Scheduler, through its standard output, is:

OPERATION JID RESULT INFO

Where:

- **OPERATION:** Is the operation requested to the Dispatch Manager. The Dispatch Manager only supports the `SCHEDULE_JOB` operation.
- **JID:** It is a job identifier.
- **RESULT:** It is the result of the operation. Could be `SUCCESS` or `FAILURE`.
- **INFO:** If **RESULT** is `FAILURE`, it contains the cause of failure. Otherwise, if **OPERATION** is `SCHEDULE_JOB` it contains a resource specification in the form `HID:QNAME:RANK`, where:
  - **HID:** It is the host identifier, as provided by `gwhosts` command.
  - **QNAME:** It is the queue name.
  - **RANK:** It is the rank of the host.

GridWay includes a scheduler template (`gw_scheduler_skel.c`) to develop custom schedulers. As an example a Round-Robin/Flooding scheduler can be found in the GridWay distribution (`gw_flood_scheduler.c`, in `$GW_LOCATION/src/sched/`). This is a very simple scheduling algorithm, which maximizes the number of jobs submitted to the Grid.

---

# GridWay Commands

---

# Name

Job and Array Job submission Command -- job submission utility for the GridWay system

```
gws submit <-t template> [-n tasks] [-h] [-v] [-o] [-s start] [-i increment] [-d  
"id1 id2 ..."]
```

# Description

Submit a job or an array job (if the number of tasks is defined) to gwd

# Command options

-h	Prints help.
-t <template>	The template file describing the job.
-n <tasks>	Submit an array job with the given number of tasks. All the jobs in the array will use the same template.
-s <start>	Start value for custom param in array jobs. Default 0.
-i <increment>	Increment value for custom param in array jobs. Each task has associated the value $PARAM=start + increment * TASK\_ID$ , and $MAX\_PARAM = start+increment*(tasks-1)$ . Default 1.
-d <"id1 id2...">	Job dependencies. Submit the job on hold state, and release it once jobs with id1,id2,.. have successfully finished.
-v	Print to stdout the job ids returned by gwd.
-o	Hold job on submission.
-p <priority>	Initial priority for the job.

---

## Name

DAG Job submission Command -- dag job submission utility for the GridWay system

```
gwdag [-h] [-d] <DAG description file>
```

## Description

Submit a dag job to gwd

## Command options

-h Prints help.

-d Writes to STDOUT a DOT description for the specified DAG job.

---

# Name

Job Monitoring Command -- report a snapshot of the current jobs

```
gwps [-h] [-u user] [-r host] [-A AID] [-s job_state] [-o output_format] [-c  
delay] [-n] [job_id]
```

# Description

Prints information about all the jobs in the GridWay system (default)

# Command options

-h Prints help.

-u user Monitor only jobs owned by user.

-r host Monitor only jobs executed in host.

-A AID Monitor only jobs part of the array AID.

-s job\_state Monitor only jobs in states matching that of job\_state.

-o output\_format Formats output information, allowing the selection of which fields to display.

-c <delay> This will cause gwps to print job information every <delay> seconds continuously (similar to top command).

-n Do not print the header.

job\_id Only monitor this job\_id.

## Output field description

**Table 3. Field options**

FIELD NAME	FIELD OPTION	DESCRIPTION	
USER	u	owner of this job	
JID	J	job unique identification assigned by the Gridway system	
AID	i	array unique identification, only relevant for array jobs	
TID	i	task identification, ranges from 0 to TOTAL_TASKS -1, only relevant for array jobs	
FP	p	fixed priority of the job	
TYPE	y	type of job (simple, multiple or mpi)	
NP	n	number of processors	
DM	s	dispatch Manager state, one of: pend, hold, prol, prew, wrap, epil, canl, stop, migr, done, fail	
EM	e	execution Manager state (Globus state): pend, susp, actv, fail, done	
RWS	f	flags:	
		R	times this job has been restarted
		W	number of processes waiting for this job
		S	re-schedule flag
START	t T	the time the job entered the system	
END	t T	the time the job reached a final state (fail or done)	
EXEC	t T	total execution time, includes suspension time in the remote queue system	
XFER	t T	total file transfer time, includes stage-in and stage-out phases	
EXIT	x	job exit code	
TEMPLATE	j	filename of the job template used for this job	
HOST	h	hostname where the job is being executed	

---

# Name

Job History Command -- shows history of a job

```
gwhistory [-h] [-n] <job_id>
```

# Description

Prints information about the execution history of a job

# Command options

-h Prints help.

-n Do not print the header lines

job\_id Job identification as provided by gwps.

# Output field description

**Table 4. Field information**

NAME	DESCRIPTION
HID	host unique identification assigned by the Gridway system.
START	the time the job start its execution on this host.
END	the time the job left this host, because it finished or it was migrated.
PROLOG	total prolog (file stage-in phase) time.
WRAPPER	total wrapper (execution phase) time.
EPILOG	total epilog (file stage-out phase) time.
MIGR	total migration time.
REASON	the reason why the job left this host.
QUEUE	name of the queue.
HOST	FQDN of the host.

---

# Name

Host Monitoring Command -- shows hosts information

```
gwhost [-h] [-c delay] [-nf] [-m job_id] [host_id]
```

# Description

Prints information about all the hosts in the GridWay system (default)

# Command options

-h Prints help.

-c <delay> This will cause gwhost to print job information every <delay> seconds continuously (similar to top command)

-n Do not print the header.

-f Full format.

-m <job\_id> Prints hosts matching the requirements of a given job.

host\_id Only monitor this host\_id, also prints queue information.

# Output field description

**Table 5. Field information**

FIELD	DESCRIPTION
HID	host unique identification assigned by the Gridway system
PRIO	priority assigned to the host
OS	operating system
ARCH	architecture
MHZ	CPU speed in MHZ
%CPU	free CPU ratio
MEM(F/T)	system memory: F = Free, T = Total
DISK(F/T)	secondary storage: F = Free, T = Total
N(U/F/T)	number of slots: U = used by GridWay, F = free, T = total
LRMS	local resource management system, the jobmanager name
HOSTNAME	FQDN of this host

**Table 6. Queue field information**

<b>FIELD</b>	<b>DESCRIPTION</b>
QUEUENAME	name of this queue
SL(F/T)	slots: F = Free, T = Total
WALLT	queue wall time
CPUT	queue cpu time
COUNT	queue count number
MAXR	max. running jobs
MAXQ	max. queued jobs
STATUS	queue status
DISPATCH	queue dispatch type
PRIORITY	queue priority

---

# Name

Job Control Command -- controls job execution

```
gkill [-h] [-a] [-k | -t | -o | -s | -r | -l | -9] <job_id [job_id2 ...] | -A  
array_id>
```

# Description

Sends a signal to a job or array job

# Command options

- h Prints help.
  - a Asynchronous signal, only relevant for KILL and STOP.
  - k Kill (default, if no signal specified).
  - t Stop job.
  - r Resume job.
  - o Hold job.
  - l Release job.
  - s Re-schedule job.
  - 9 Hard kill, removes the job from the system without synchronizing remote job execution or cleaning remote host.
- job\_id [job\_id2 ...]            Job identification as provided by gwps. You can specify a blank space separated list of job ids.
- A <array\_id>            Array identification as provided by gwps.

---

# Name

Job Synchronization Command -- synchronize a job

```
gwwait [-h] [-a] [-v] [-k] <job_id ...| -A array_id>
```

# Description

Waits for a job or array job

# Command options

-h Prints help.

-a Any, returns when the first job of the list or array finishes.

-v Prints job exit code.

-k Keep jobs, they remain in fail or done states in the GridWay system. By default, jobs are killed and their resources freed.

-A <array\_id> Array identification as provided by gwps.

job\_id ... Job ids list (blank space separated).

---

# Name

User Monitoring Command -- monitors users in GridWay

```
gwuser [-h] [-n]
```

# Description

Prints information about users in the GridWay system

# Command options

-h Prints help.

-n Do not print the header.

# Output field description

**Table 7. Field information**

<b>FIELD</b>	<b>DESCRIPTION</b>
UID	user unique identification assigned by the Gridway system
NAME	name of this user
JOBS	number of Jobs in the GridWay system
RUN	number of running jobs
IDLE	idle time, (time with JOBS = 0)
EM	execution manager drivers loaded for this user
TM	transfer manager drivers loaded for this user
PID	process identification of driver processes

---

# Name

Accounting Command -- prints accounting information

```
gwacct [-h] [-n] [<-d n | -w n | -m n | -t s>] <-u user|-r host>
```

# Description

Prints usage statistics per user or resource. Note: accounting statistics are updated once a job is killed.

# Command options

-h Prints help.

-n Do not print the header.

<-d n | -w n | -m n | -t s> Take into account jobs submitted after certain date, specified in number of days (-d), weeks (-w), months (-m) or an epoch (-t).

-u user Print usage statistics for user.

-r hostname Print usage statistics for host.

# Output field description

**Table 8. Field information**

FIELD	DESCRIPTION
HOST/USER	host/user usage summary for this user/host
XFR	total transfer time on this host (for this user)
EXE	total execution time on this host (for this user), without suspension time
SUSP	total suspension (queue) time on this host (for this user)
TOTS	total executions on this host (for this user). Termination reasons: <ul style="list-style-type: none"><li>• SUCC success</li><li>• ERR error</li><li>• KILL kill</li><li>• USER user requested</li><li>• SUSP suspension timeout</li><li>• DISC discovery timeout</li><li>• SELF self migration</li><li>• PERF performance degradation</li><li>• S/R stop/resume</li></ul>

---

## Name

JSDL To GridWay Job Template Parser Command -- parser to translate JSDL file into GridWay Job Template file

```
jsdl2gw [-h] input_jsdl [output_gwjt]
```

## Description

Converts a jsdl document into a gridway job template. If no output file is defined, it defaults to the standard output. This enables the use of pipes with gws submit in the following fashion:

```
jsdl2gw jsdl-job.xml | gws submit
```

## Command options

-h Prints help.

input\_jsdl Reads the jsdl document from the input\_jsdl

output\_gwjt Stores the GridWay Job Template specification in the output\_gwjt.jt file

---

# Chapter 3. Configuration Guide

## 1. 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 4.2.1, 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
```

## 1.1. 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 2.6, “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](#)<sup>1</sup>).

## 1.2. 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.

---

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

**Table 3.1. 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 \$GW_LOCATION/var/gridway/gwd.port	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 <a href="#">Section 3.4, “Information Driver Configuration”</a>	-
TM_MAD	Transfer Manager MADs, see <a href="#">Section 3.3, “File Transfer Driver Configuration”</a>	-
EM_MAD	Execution Manager MADs, see <a href="#">Section 3.2, “Execution Driver Configuration”</a>	-
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 <a href="#">Section 2.6, “Scheduler Configuration”</a>	-

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

```

## 1.3. 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](#)<sup>2</sup> for a detailed description of each job option.

## 1.4. Running gwd

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.
- `$GW_LOCATION/var/acct`: Accounting information. Use the **gwacct** command to access the data bases. Note that you need Berkeley DB library (version 4.4.20).
- `$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
```

---

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

## 1.5. 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
```

## 2. 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.

### 2.1. 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 1.2, “GridWay Daemon \(GWD\) Configuration”](#)).

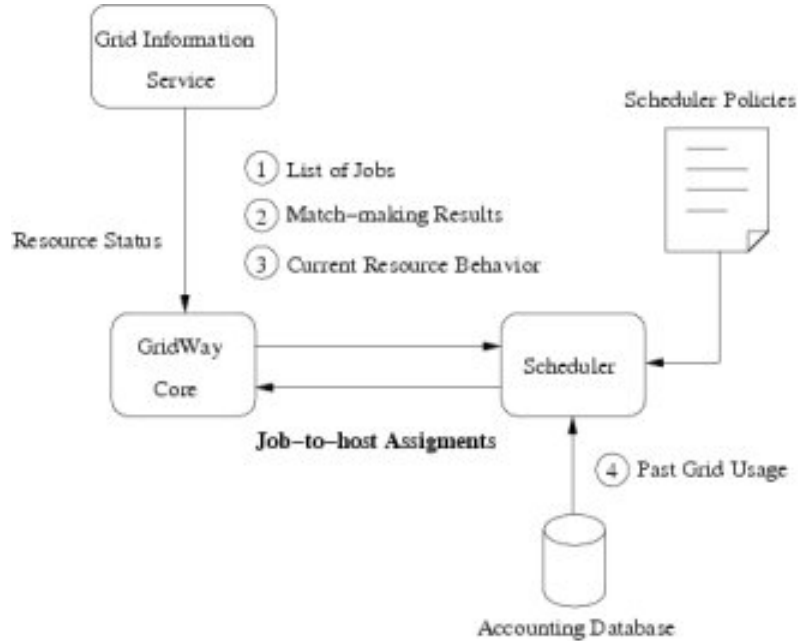
In order to make job to resource assignments the scheduler receives information from the following sources (see [Figure 3.1, “Job Scheduling in GridWay”](#)):

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 3.4, “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 3.1. Job Scheduling in GridWay**



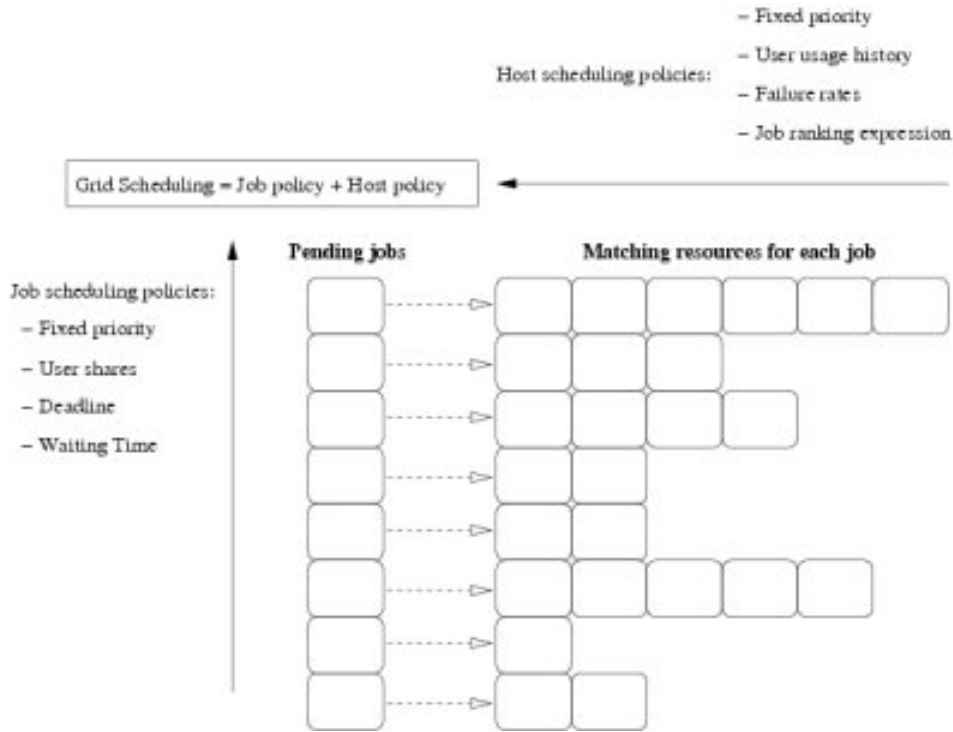
## 2.2. Scheduling Policies

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 3.2, “Job and resource prioritization policies in GridWay.”](#)). The above scheduling policies are described in the following sections.

**Figure 3.2. Job and resource prioritization policies in GridWay.**



## 2.3. 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.

### 2.3.1. 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 2.5, "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

```

### 2.3.2. 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.

### 2.3.3. 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 2.5, “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

```

### 2.3.4. 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
```

### 2.3.5. 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 2.5, “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
```

### 2.3.6. 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 3.3. 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.

## 2.4. 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.

### 2.4.1. 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
```

### 2.4.2. 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/)<sup>3</sup>.

Example configuration for the Rank policy:

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

---

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

### 2.4.3. 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 3.4. 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
```

### 2.4.4. 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 3.5. Banned time formula**

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

where  $T_\infty$  is the maximum time that a resource can be *banned*,  $\Delta t$  is the time since last failure, and  $C$  is a constant that determines how fast the  $T_\infty$  limit is reached.

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

- **FR\_MAX\_BANNED\_TIME**. The value of  $T_\infty$ , 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
```

### 2.4.5. 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 3.6. 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.

### 2.4.6. 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 1.2, “GridWay Daemon (GWD) Configuration” and the [GridWay user guide](#)<sup>4</sup>, for information on configuring these policies.

## 2.5. 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.

---

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

**Table 3.2. Built-in Scheduler Configuration File Options.**

Option	Description	Default
<b>Job Scheduling Policies.</b> 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
<b>Resource Scheduling Policies.</b> 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 <a href="#">Section 1.2, “GridWay Daemon (GWD) Configuration”</a> ). 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

## 2.6. 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.

## 2.6.1. 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.

## 2.6.2. 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.

# 3. 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 1.2, “GridWay Daemon \(GWD\) Configuration”](#). Additionally you may need to configure your environment (see [Chapter 5, 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`).

## 3.1. 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-4.2
. $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
```

## 3.2. 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:

- GRAM2 (Globus Toolkit 2.4 and above)
- GRAM4 (Globus Toolkit 4.0 and above)

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.

### 3.2.1. 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.

## 3.3. 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]>
```

### 3.3.1. 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.

### 3.3.2. 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
```

## 3.4. 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 and above)

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]:<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.
- **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.

### 3.4.1. 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_MAXCPU[0]=20
QUEUE_MAXCOUNT[0]=4 QUEUE_MAXRUNNINGJOBS[0]=0 QUEUE_MAXJOBSINQUEUE[0]=1
QUEUE_STATUS[0]="enabled" QUEUE_DISPATCHTYPE[0]="batch"
QUEUE_NAME[1]="q4medium" QUEUE_NODECOUNT[1]=4 QUEUE_FREENODECOUNT[1]=4
QUEUE_MAXTIME[1]=0 QUEUE_MAXCPU[1]=120 QUEUE_MAXCOUNT[1]=4
QUEUE_MAXRUNNINGJOBS[1]=0 QUEUE_MAXJOBSINQUEUE[1]=1
QUEUE_STATUS[1]="enabled" QUEUE_DISPATCHTYPE[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_MAXCPU[0]=0
QUEUE_MAXCOUNT[0]=0 QUEUE_MAXRUNNINGJOBS[0]=0 QUEUE_MAXJOBSINQUEUE[0]=0
QUEUE_STATUS[0]="0" QUEUE_DISPATCHTYPE[0]="Immediate"
```

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

### 3.4.2. 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, with 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 [Chapter 8, Troubleshooting](#).

## 3.4.3. 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.

### 3.4.4. 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"
```

## 4. Integration Guides

GridWay architecture flexibility allows it to interoperate with grids based on different middleware stacks. The following documents states how to configure GridWay for the following infrastructures:

- Integration of GridWay within the EGEE infrastructure[\[HTML\]](#)<sup>5</sup>[\[PDF\]](#)<sup>6</sup>
- Integration of GridWay within the OSG infrastructure[\[HTML\]](#)<sup>7</sup>[\[PDF\]](#)<sup>8</sup>
- Integration of GridWay within the Teragrid infrastructure[\[HTML\]](#)<sup>9</sup>[\[PDF\]](#)<sup>10</sup>
- Integration of GridWay within the Nordugrid infrastructure[\[HTML\]](#)<sup>11</sup>[\[PDF\]](#)<sup>12</sup>
- Creating MADs: SSH Example[\[HTML\]](#)<sup>13</sup>[\[PDF\]](#)<sup>14</sup>

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

<sup>6</sup> <http://www.gridway.org/documentation/stable/egeehowto.pdf>

<sup>7</sup> <http://www.gridway.org/documentation/stable/osghowto/>

<sup>8</sup> <http://www.gridway.org/documentation/stable/osghowto.pdf>

<sup>9</sup> <http://www.gridway.org/documentation/stable/tghowto/>

<sup>10</sup> <http://www.gridway.org/documentation/stable/tghowto.pdf>

<sup>11</sup> <http://www.gridway.org/documentation/stable/norduhowto>

<sup>12</sup> <http://www.gridway.org/documentation/stable/norduhowto.pdf>

<sup>13</sup> <http://www.gridway.org/documentation/stable/sshowto>

<sup>14</sup> <http://www.gridway.org/documentation/stable/sshowto.pdf>

---

# Chapter 4. Environment variable interface

## 1. Environment variables for GridWay

### Important

You should include the following environment variables in your shell configuration file. (example `$HOME/.bashrc`)

In order to set the user environment, follow these steps:

1. Set up Globus user environment:

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

or

```
$ . $GLOBUS_LOCATION/etc/globus-user-env.csh
```

depending on the shell you are using.

2. Set up the GridWay user environment:

```
$ export GW_LOCATION=<path_to_GridWay_installation>
$ export PATH=$PATH:$GW_LOCATION/bin
```

or

```
$ setenv GW_LOCATION <path_to_GW_location>
$ setenv PATH $PATH:$GW_LOCATION/bin
```

depending on the shell you are using.

3. Optionally, you can set up your environment to use the GridWay DRMAA library:

```
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$GW_LOCATION/lib
```

or:

```
$ setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH:$GW_LOCATION/lib
```

4. If GridWay has been compiled with accounting support, you may need to set up the DB library. For example, if DB library has been installed in `/usr/local/BerkeleyDB.4.4`:

```
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/BerkeleyDB.4.4/lib
```

**Note**

This step is only needed if your environment has not been configured, ask your administrator.

5. DRMAA extensions for all the languages use the dynamic drmaa libraries provided by GridWay. To use this libraries it is needed to tell the operating system where to look for them. Here are described the steps needed to do this in Linux and MacOS X.

- 1. In linux we have two ways to do this, one is using environment variables and the other one is modifying systemwide library path configuration. You only need to use one of this methods. If you do not have root access to the machine you are using or you do not want to setup it for every user in your system you have to use the environment variable method.
- • 1.1 The environment variable you have to set so the extensions find the required DRMAA library is `LD_LIBRARY_PATH` with a line similar to:

```
export LD_LIBRARY_PATH=$GW_LOCATION/lib
```

If you want to setup this systemwide you can put this line alongside `GW_LOCATION` setup into `/etc/profile`. If you do not have root access or you want to do it per user the best place to do it is in the user's `.bashrc`.

You can also do this steps in the console before launching your scripts as it will have the same effect.

- Systems that use GNU/libc (GNU/Linux is one of them) do have a systemwide configuration file with the paths where to look for dynamic libraries. You have to add this line to `/etc/ld.so.conf`:

```
<path_to_gridway_installation>/lib
```

After doing this you have to rebuild the library cache issuing this command:

```
# ldconfig
```

- In MacOS X you have to use the environment variable method described for Linux but this time the name of the variable is `DYLD_LIBRARY_PATH`.

---

# Appendix A. Errors

**Table A.1. Gridway Errors**

Error Code	Definition	Possible Solutions
Lock file exists	Another GWD may be running.	Be sure that no other GWD is running, then remove the lock file and try again.
Error in MAD initialization	There may be problems with the proxy certificate, bin directory, or the executable name of a MAD may not be in the correct location.	Check that you have generated a valid proxy (for example with the <b>grid-proxy-info</b> command). Also, check that the directory <code>\$GW_LOCATION/bin</code> is in your path, and the executable name of all the MADs is defined in <code>gwd.conf</code> .
Could not connect to gwd	GridWay may not be running or there may be something wrong with the connection.	Be sure that GWD is running; for example: <pre>pgrep -l gwd</pre> If it is running, check that you can connect to GWD; for example: <pre>telnet `cat \$GW_LOCATION/var/gwd.port`</pre>