Administering Guide (Printable)

The information presented here is also available in individual pages in the Administering section. This page is designed to enable you to easily export the information to a PDF or Word document.

To print a version of this document, log in to wikis.sun.com, click Tools, then select Export to PDF or Export to Word.

Contents

Administering Sun Grid Engine

Administering Guide (Printable)

Managing Your Cluster

Topic	Description
Interacting With Sun Grid Engine as an Administrator	Learn how you can use the command line interface, the graphical user interface (QMON), and the Distributed Resource Management Application API (DRMAA) to interact with the Sun Grid Engine system.
Understanding the Bootstrap File	The bootstrap file contains parameters that are needed for starting up the Grid Engine components. The bootstrap file is created during the `sge_qmaster` installation. You cannot modify the bootstrap file in a running system.
Using qconf	Learn how you can use `qconf` to manage most components of the Sun Grid Engine system. The qconf Cheat Sheet lists the most commonly-used `qconf` commands.
Inspect	Learn how to use Inspect to monitor your cluster(s).
Configuring Clusters	Learn how to display, add, modify and delete cluster configurations. (Note: Adapt the global configuration and local host configurations to your site's needs immediately after installation.)

Configuring Hosts	Learn how to configure the master host, the shadow master host, execution hosts, administration hosts, submit hosts, and host groups.
Configuring Queues	Learn how to configure queues.
Configuring Queue Calendars	Learn how to configure queue calendars.
Managing User Access	Learn how to configure user access, configure projects, configure default requests, and use path aliasing.
Using Job Submission Verifiers	Learn how to use Job Submission Verifiers to verify, modify or reject a job during the time of job submission.

Managing Resource Allocation and Scheduling

Topic	Description
Configuring Resource Attributes	Learn how to configure complex resource attributes to match the requirements of your environment.
Configuring Load Parameters	Learn how to add site-specific load parameters and write your own load sensors.
Managing Policies	Learn about the types of user policies that are available and how to match these policies to your scheduling and resource allocation needs.
Managing the Scheduler	Learn how to modify the scheduler.
Managing Advance Reservations	Learn how to maximize your computing resources by using advance reservations.
Managing Resource Quotas	Learn how to use the resource quotas feature to limit resources by user, project, host, cluster queue, or parallel environment.

Managing Special Environments

Topic	Description
Managing Parallel Environments	Learn how to set up concurrent computing on parallel platforms in networked environments using Sun Grid Engine.
Managing Checkpointing Environments	Learn how to use user-level checkpointing or kernel-level transparent checkpointing to manage your environment.

Other Administrative Tasks

Topic	Description
Monitoring and Controlling SMF Services	Learn how to monitor and Control SMF Services.
Generating Accounting Statistics	Learn how to generate accounting information according to real time, user time, or system time.
Backing Up and Restoring System Configuration	Learn how to back up your Grid Engine system configuration files automatically.
Fine-Tuning Your Environment	Learn about how you can fine tune your environment.
Using DTrace for Performance Tuning	Learn how to use DTrace for performance tuning.

To print this section, see the Administering Guide (Printable).

Interacting With Sun Grid Engine as an Administrator

As an administrator, you can choose to interact with the Sun Grid Engine system using the command line interface, the graphical user interface (QMON), and the Distributed Resource Management Application API (DRMAA).

The Command Line Interface

The command line interface provides more flexibility than the graphical user interface in configuring, monitoring, and controlling the Grid Engine system. Many experienced administrators find that using files and scripts is a more flexible, quicker, and more powerful way to change settings.

The following commands are central to Sun Grid Engine administration:

- `qconf` – Add, delete, and modify the current Grid Engine configuration. For more information, see Using qconf.
- `qhost` – View current status of the available Grid Engine hosts, the queues, and the jobs associated with the queues. For more information, see the `qhost(1)` man page.
- `qalter` and `qsub` – Submit jobs. For more information, see the `submit(1)` man page.
- `qstat` – Show the status of Grid Engine jobs and queues. For more information, see the `qstat(1)` man page.
- `qquota` – List each resource quota that is being used at least once or that defines a static limit. For more information, see the `qquota(1)` man page.

For information on the ancillary programs that Sun Grid Engine provides and which users have access to these commands, see Command Line Interface Ancillary Programs.

QMON - The Graphical User Interface

You can use QMON, the graphical user interface (GUI) tool, to accomplish most Grid Engine system tasks. Only administrators can configure QMON using the specifically designed resource file. Reasonable defaults are compiled in *$SGE_ROOT*`/qmon/Qmon`. This file also includes a sample resource file. Refer to the comment lines in the sample `Qmon` file for detailed information on the possible customizations.

As the cluster administrator, you can do any of the following:

- Install site-specific defaults in standard locations such as `/usr/lib/X11/app-defaults/Qmon`.
- Include QMON-specific resource definitions in the standard `.Xdefaults` or `.Xresources` files.
- Put a site-specific `Qmon` file in a location referenced by standard search paths such as `XAPPLRESDIR`.

The Distributed Resource Management Application API (DRMAA)

You can automate Sun Grid Engine functions by writing scripts that run Sun Grid Engine commands and parse the results. However, for more consistent and efficient results, you can use the Distributed Resource Management Application API (DRMAA). For more information about the DRMAA concept and how to use it with the C and Java TM languages, see Automating Grid Engine Functions Through DRMAA.

For general information about these administration tools, see Choosing a User Interface.

Understanding the Bootstrap File

The bootstrap file contains parameters that are needed for starting up the Grid Engine components. The bootstrap file is created during the `sge_qmaster` installation. You cannot modify the bootstrap file in a running system.

Note
Any changes made to the bootstrap file become effective only after restarting the qmaster.

Bootstrap File Parameters

The following section provides a brief description of the individual parameters that compose the bootstrap configuration for a Sun Grid Engine cluster.

admin_user

The admin_user parameter is the administrative user account used by Sun Grid Engine for all internal file handling operations like status spooling, message logging, and so on. This parameter can be used in cases where the root user account does not have the corresponding file access permissions. For example, on a shared file system without global root read/write access.

As the admin_user parameter is set at installation time, you cannot change the parameter in a running system. You can manually change the admin_user parameter on a shutdown cluster. However, if access to the Sun Grid Engine spooling area is interrupted, it will result in unpredictable behavior.

The admin_user parameter has no default value. The default value can be defined during the master installation procedure.

default_domain

The default_domain parameter is needed if your Sun Grid Engine cluster covers hosts belonging to more than a single DNS domain. In this case, it can be used if your host name resolving yields both qualified and unqualified host names for the hosts in one of the DNS domains. The value of the default_domain parameter is appended to the unqualified host name to define a fully qualified host name.

The default_domain parameter will have no effect if the ignore_fqdn parameter is set to True.

As the default_domain parameter is set at installation time, you cannot change the parameter in a running system.

The default value for the default_domain parameter is None.

ignore_fqdn

The ignore_fqdn parameter is used to ignore the fully qualified domain name component of host names. This parameter should be set if all hosts belonging to a Sun Grid Engine cluster are part of a single DNS domain. The ignore_fqdn parameter is enabled if it is set to either True or 1. Enabling the ignore_fqdn parameter can solve problems with load reports caused due to different host name resolutions across the cluster.

As the ignore_fqdn parameter is set at installation time, you cannot change the parameter in a running system.

The default value for the ignore_fqdn parameter is True.

spooling_method

The spooling_method parameter defines how sge_qmaster(8) writes its configuration and the status information of a running cluster. The available spooling methods are berkeleydb and classic.

spooling_lib

The name of a shared library containing the spooling_method parameter to be loaded at sge_qmaster(8) initialization time. The extension characterizing a shared library like .so, .sl, or .dylib is not contained in the spooling_lib parameter.

If the spooling_method parameter is set to berkeleydb during installation, the spooling_lib parameter is set to libspoolb. If the classic option is chosen as spooling_method during installation, the spooling_lib parameter is set to libspoolc.

You should note that not all operating systems allow the dynamic loading of libraries. On such operating systems a certain spooling method with the default value berkeleydb is compiled into the binaries and the spooling_lib parameter will be ignored.

spooling_params

The spooling_params parameter defines parameters to the chosen spooling method. These parameters are required to initialize the spooling framework. For example, you can define parameters to open database files or to connect to a certain database server.

The spooling parameter value for the berkeleydb spooling method is [rpc_server:]database directory. For example, /sge_local/default/spool/qmaster/spooldb for spooling to a local file system or myhost:sge for spooling over a Berkeley DB RPC server.

The spooling parameter value for the classic spooling method is <common_dir>;<qmaster spool dir>. For example, /sge/default/common;/sge/default/spool/qmaster.

binary_path

The directory path where the Sun Grid Engine binaries reside. It is used within the Sun Grid Engine components to locate and start up other Sun Grid Engine programs.

The path name given here is searched for binaries as well as any directory below with a directory name equal to the current operating system architecture. Therefore, `/usr/SGE/bin` will work for all architectures, if the corresponding binaries are located in subdirectories named aix43, cray, lx24-x86, hp11, irix65, tru64, sol-sparc, and so on.

The default location for the binary path is <sge_root>/bin

qmaster_spool_dir

The location where the master spool directory resides. sge_qmaster(8) and sge_shadowd(8) need to have access to this directory. The master spool directory, in particular the job_scripts directory and the messages log file, may become quite large depending on the size of the cluster and the number of jobs. Ensure that you allocate enough disk space and regularly clean up the log files. For example, you can achieve this with the help of a cron(8) job.

As the `qmaster_spool_dir` parameter is set at installation time, you cannot change the parameter in a running system.

The default location for the `qmaster_spool_directory` parameter is <sge_root>/<cell>/spool/qmaster.

security_mode

The `security_mode` parameter defines the set of security features used by the Sun Grid Engine cluster. The possible security mode settings are None, AFS, DCE, KERBEROS, and CSP.

> Note
> Modifying the `security_mode` parameter generally will require you to reinstall the Sun Grid Engine cluster. For more information on editing the `security_mode` parameter, contact a Sun Grid Engine support specialist.

listener_threads

The `listener_threads` parameter defines the number of listener threads. The default number is set during installation.

worker_threads

The `worker_threads` parameter defines the number of worker threads. The default number is set during installation.

scheduler_threads

The `scheduler_threads` parameter defines the number of scheduler threads. The allowed values for this parameter are 0 and 1. The default value (1) is set during installation. For more information, refer qconf(1) -kt/-at option.

jvm_threads

The `jvm_threads` parameter defines the number of JVM threads. The allowed values are 0 and 1. The default value is set during installation.

> Note
> The bootstrap file parameters `admin_user`, `default_domain`, `ignore_fqdn`, and `binary_path` also effect the behavior of the execution daemon execd and require that execd be restarted at the same time as qmaster.

Using qconf

`qconf` is the central configuration command. As a system administrator, you can use `qconf` to add, delete, and modify the current Grid Engine configuration.

To learn how to most effectively use `qconf`, familiarize yourself with the following:

- `qconf` Syntax

- qconf Command Execution Methods
- Automating the qconf Command Using the Editor Environment Variable

See the qconf Cheat Sheet for a reference list of the most commonly used `qconf` commands. For more information, see the `qconf(1)` man page.

`qconf` Syntax

Note
While the syntax for `qconf` is fairly consistent, there are a few exceptions. For example, `qconf -aprj` does not take an argument. See the qconf Cheat Sheet for more info.

The following options consistently apply when using `qconf`:

Function	Syntax
Add	`a`
Modify a value setting	`m`
Delete	`d`
Show	`s`
Show complete list of currently-configured objects	`s<obj>l`
Replace the entire list of settings	`r`
Add from file	`A`
Modify a value setting from file	`M`

The following object specifications are used:

object	Specification
Administrative Host	`h`
Calendar	`cal`
Checkpointing Environment	`ckpt`
Cluster	`conf`
Complex	`c`
Execution Host	`e`
Host Group	`hgrp`
Manager	`m`
Operator	`o`
Parallel Environment	`p`
Project	`prj`
Queue	`q`
Submit Host	`s`
User	`user`
User Access List	`u`

See the qconf Cheat Sheet for a reference list of commonly used commands.

qconf Command Execution Methods

The qconf command can be used to add new objects or modify existing objects from the specification in a file.

You can use `qconf` to do the following:

- Add new objects

```
qconf -a<obj_spec> <object name>
```

- Add new objects from file

```
qconf -A<obj_spec> <filename>
```

- Modify existing objects using an editor

```
qconf -m<obj_spec> <object name>
```

- Modify existing objects from file

```
qconf M<obj_spec> <filename>
```

- Modify many objects at once (or modify an object non-interactively)

```
qconf -{a,m,r,d}attr <obj_spec> <attrib> <value> <queue_list>|<host_list>
```

- Modify many objects at once from file (or modify an object non-interactively)

```
qconf -{A,M,R,D}attr <obj_spec> <filename>
```

Automating the qconf Command Using the Editor Environment Variable

You can use the `EDITOR` environment variable to automate the behavior of the `qconf` command. Change the value of this variable to point to an editor program that modifies a file whose name is given by the first argument. After the editor modifies the temporary file and exits, the system reads in the modifications, which take effect immediately.

Note
If the modification time of the file does not change after the edit operation, the system sometimes incorrectly assumes that the file was not modified. Therefore you should insert a `sleep 1` instruction before writing the file, to ensure a different modification time.

You can use this technique with any `qconf -m...` command. However, the technique is especially useful for administration of the scheduler and the global configuration, as you cannot automate the procedure in any other way.

qconf Cheat Sheet

- Complex Configuration
- Calendar Configuration
- Checkpointing Environment Configuration

- Cluster Configuration
- Event Client Configuration
- Host Configuration
- Host Group Configuration
- Parallel Environment Configuration
- Project Configuration
- Queue Configuration
- Scheduler Configuration
- Sharetree Configuration
- User Configuration
- User Access List Configuration

For a detailed list of available `qconf` commands, see the `qconf(1)` man page.

Complex Configuration

Command	Description
`qconf -sc`	Show the current complex configuration.
`qconf -mc`	Modify the complex configuration using an editor.
`qconf -Mc` filename	Modify the current complex configuration by overwriting the current configuration as specified by the file filename.

For more information, see How to Configure the Complex From the Command Line.

Calendar Configuration

Command	Description
`qconf -scal` calendarname	Show the configuration for the specified calendar.
`qconf -scall`	Show a list of all currently configured calendars.
`qconf -acal` calendarname	Add a new calendar definition to the Sun Grid Engine environment.
`qconf -Acal` filename	Add a new calendar definition to the Sun Grid Engine environment from the file filename.
`qconf -mcal` calendarname	Modify the specified calendar configuration using an editor.
`qconf -Mcal` filename	Modify the current calendar configuration by overwriting the current configuration as specified by filename.
`qconf -dcal` calendarname	Delete a calendar.

For more information, see How to Configure Queue Calendars From the Command Line.

Checkpointing Environment Configuration

Command	Description
`qconf -sckpt` checkpointname	Show the configuration of the specified checkpointing environment.
`qconf -sckptl`	Show a list of all currently configured checkpointing environments.
`qconf -ackpt` checkpointname	Add a checkpointing environment to the Sun Grid engine environment.
`qconf -Ackpt` filename	Add a checkpointing environment from the file filename.
`qconf -mckpt` checkpointname	Modify the specified checkpointing environment using an editor.
`qconf -Mckpt` filename	Modify a checkpointing environment from file.
`qconf -dckpt` checkpointname	Delete a checkpointing environment.

For more information, see How to Configure Checkpointing Environments From the Command Line.

Cluster Configuration

Command	Description
`qconf -sconf` [host,...|global]	Show a local host configuration or the global cluster configuration.
`qconf -sconfl`	Show a list of all currently configured hosts.
`qconf -aconf` host,...	Add a host configuration.
`qconf -Aconf` filelist	Add the configuration specified in the files enlisted in the comma-separated filelist.
`qconf -mconf` [host,...|global]	Modify the one or more local host configurations or the global cluster configuration.
`qconf -Mconf` filelist	Modify the configurations specified in the files enlisted in the comma separated filelist.
`qconf -dconf` host,...	Delete one or more hosts from the configuration list.

For more information, see How to Configure Clusters From the Command Line.

Event Client Configuration

Command	Description
`qconf -secl`	Show event client list.
`qconf -kec` [eventclientname,...|all]	Shuts down event clients registered at the master daemon.

Host Configuration

Command	Description
`qconf -se` host	Show the configuration for the specified execution host.
`qconf -sel`	Show a list of all currently-configured execution hosts.
`qconf -ae`	Add an execution host.
`qconf -Ae` filename	Add an execution host from file)filename.
`qconf -me` host	Modify the specified execution host using an editor.
`qconf -Me` filename	Modify an execution host from file filename.
`qconf -de` host,...	Delete one ore more execution hosts.
`qconf -sh`	Show a list of all currently-configured administrative hosts.
`qconf -ah` host,...	Add one ore more administrative hosts.
`qconf -dh` host,...	Delete one ore more administrative hosts from the administrative host list.
`qconf -ss`	Show a list of all currently-configured submit hosts.
`qconf -as` host,...	Add one more submit hosts.
`qconf -ds` host,...	Delete a submit host.

For more information, see Configuring Hosts.

Host Group Configuration

Command	Description
`qconf -shgrp` groupname	Show the configuration for the specified host group.
`qconf -shgrpl`	Show a list of all currently configured host groups.

`qconf -ahgrp` hostgroupname	Add a host group.
`qconf -Ahgrp` filename	Add a host group configuration from file filename.
`qconf -mhgrp` hostgroupname	Modify the specified host group using an editor.
`qconf -Mhgrp` filename	Modify a host group configuration from file filename.

Fore more information, see How to Configure Host Groups From the Command Line.

Parallel Environment Configuration

Command	Description
`qconf -sp` pename	Show the configuration for the specified parallel environment.
`qconf -spl`	Show a list of all currently configured parallel environments.
`qconf -ap` pename	Add a new parallel environment.
`qconf -Ap` filename	Add a parallel environment from file filename.
`qconf -mp` pename	Modify the specified parallel environment using an editor.
`qconf -Mp` filename	Modify a parallel environment from file filename.
`qconf -dp` pename	Delete the specified parallel environment.

For more information, see How to Configure Parallel Environments From the Command Line.

Project Configuration

Command	Description
`qconf -sprj` projectname	Show the configuration for the specified project.
`qconf -sprjl`	Show a list of all currently configured projects.
`qconf -aprj`	Add a new project.
`qconf -Aprj` filename	Add a new project from file filename.
`qconf -mprj` projectname	Modify the specified project using an editor.
`qconf -Mprj` filename	Modify a project from file filename.
`qconf -dprj` projectname	Delete a project.

For more information, see How to Configure Projects From the Command Line.

Queue Configuration

The most flexible way to automate the configuration of queue instances is to use the `qconf` command with the `qselect` command. With the combination of these commands, you can build up your own custom administration scripts.

Command	Description
`qconf -sq` wc_queue_list	Show one or multiple cluster queues or queue instances.
`qconf -sql`	Show a list of all currently configured cluster queues.
`qconf -aq` queuename	Add a new queue.
`qconf -Aq` filename	Add a queue from file filename.
`qconf -mq` queuename	Modify the specified queue using an editor.

qconf -Mq filename	Modify a queue from file filename.
qconf -dq queuename	Delete a queue.

> -purge allows you to delete an entire list attribute from the underlying queue instances of a cluster queue, whereas -dattr only deletes a value or an item from a list attribute.

For more information, see Configuring Queues.

Scheduler Configuration

Command	Description
qconf -ssconf	Show the current scheduler configuration.
qconf -msconf	Modify the scheduler configuration using an editor.
qconf -tsm	Trigger scheduler monitoring.

For more information, see Managing the Scheduler.

Sharetree Configuration

Command	Description
qconf -sstree	Show the sharetree.
qconf -sstnode nodelist	Show one or more sharetree node.
qconf -rsstnode nodelist	Show one or more sharetree nodes and their children.
qconf -astree	Add a sharetree.
qconf -Astree filename	Add a sharetree from filename.
qconf -astnode nodelist	Add one or more sharetree nodes.
qconf -mstree	Modify a sharetree.
qconf -Mstree filename	Modify a sharetree from filename.
qconf -dstree	Delete a sharetree.
qconf -dstnode nodelist	Delete or more sharetree nodes.

User Configuration

Command	Description
qconf -suser username,...	Show the configuration for one or more users.
qconf -suserl	Show a list of all currently-configured users.
qconf -auser	Add a user to the list of registered users.
qconf -Auser filename	Add a user from file filename.
qconf -muser username	Modify the specified user configuration using an editor.
qconf -Muser filename	Modify a user from file filename.
qconf -duser username,...	Delete one or more users.
qconf -sm	Show a list of all currently-configured managers.

`qconf -am` username	Add a user to the manager list.
`qconf -dm` username,...	Delete one or more managers.
`qconf -so`	Show a list of all currently-configured operators.
`qconf -ao` username	Adds a user to the operator list.
`qconf -do` username,...	Deletes one or more operators.

For more information, see Configuring User Access.

User Access List Configuration

Command	Description
`qconf -su` aclname [,...]	Displays the configuration for the specified access list.
`qconf -sul`	Displays a list of all currently configured user access lists.
`qconf -au` username [,...] accesslistname [,...]	Adds a user or users to an access list or lists.
`qconf -Au` filename	Adds a user access list from file.
`qconf -mu` aclname	Opens an editor to modify the specified access list configuration.
`qconf -Mu` filename	Modifies a user access list from file.
`qconf -du` username [,...] aclname [,...]	Deletes a user or users to an access list or lists.
`qconf -dul` aclname [,...]	Deletes one or more user access lists.

For more information, see How to Configure User Access Lists From the Command Line.

Configuring Clusters

The basic cluster configuration is a set of information that is configured to reflect site dependencies and to influence Grid Engine system behavior. Site dependencies include valid paths for programs such as `mail` or `xterm`. A global configuration is provided for the master host as well as for every host in the Grid Engine system pool. In addition, you can configure the system to use a configuration local to each host to override particular entries in the global configuration.

The cluster administrator should adapt the global configuration and local host configurations to the site's needs immediately after the installation. The configurations should then be kept up to date.

Cluster Configuration Recommendations

The following recommendations apply to all clusters:

- Only in the smallest clusters does it makes sense for the execution daemon spooling directories to be located on a NFS-mounted file system.
- Configure the default priority for all users' jobs to be less than 0. If the default priority is left at 0, users are only able to reduce the priorities of their jobs because they are not allowed to submit jobs with a priority greater than 0. If the default priority is set at less than 0, then users have room to increase the priority of their jobs. `-100` is a suggested value.

Task	User Interface	Description
How to Configure Basic Clusters	CLI or QMON	Learn how to display and modify the global configuration and local host configurations and how to add and delete local host configurations.

How to Configure Clusters From the Command Line

Note
You must be an administrator to use the qconf command to change cluster configurations.

To configure clusters from the command line, use the following arguments for the qconf command:

- To show the global cluster configuration, type the following command:

  ```
  % qconf -sconf [global]
  ```

  The -sconf and -sconf global options are equivalent. They both show the global configuration.

- To show a local host configuration, type the following command:

  ```
  % qconf -sconf <host,...>
  ```

  The -sconf option shows the specified local host's configuration.

- To add a new local host configuration, type the following command:

  ```
  % qconf -aconf <host,...>
  ```

  The -aconf option adds new local host configurations. For each host, an editor is invoked that is used to enter the configuration for that host. The configuration is then registered with the sge_qmaster. The following host names are invalid, reserved, or otherwise not allowed to be used:

  - global
  - template
  - all
  - default
  - unknown
  - none
- To modify a local host configuration, type the following command:

  ```
  % qconf -mconf <host,...>
  ```

  The -mconf option modifies the local configuration of the specified execution host or master host. For more information, see Cluster Configuration Parameters.

- To modify the global configuration, type the following command:

  ```
  % qconf -mconf [global]
  ```

  The -mconf option modifies the global configuration. For more information, see Cluster Configuration Parameters.

- To delete a local host configuration, type the following command:

  ```
  % qconf -dconf <host,...>
  ```

  The -dconf option deletes the configuration of the specified execution host or master host.

For more information on how to configure the cluster from file or how to modify many objects at once, see Using qconf. For more examples of available `qconf` commands, see the `qconf(1)` man page.

Cluster Configuration Parameters

For detailed information about these parameters, see the `sge_conf(5)` man page.

Unless otherwise mentioned, the global configuration entry for the following parameters can be overwritten by the execution host local configuration.

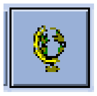
Parameter	Description
`execd_spool_dir`	The execution daemon spool directory path. Note: If you would like to change the global `execd_spool_dir` that was set during installation, you must restart all affected execution daemons. Default path: `$SGE_ROOT/$SGE_CELL/spool`
`mailer`	The absolute pathname to the electronic mail delivery agent on your system. It must accept `mailer -s <email subject> <recipient>`. The default depends on the operating system of the host on which the Sun Grid Engine master installation was run. Common values are `/bin/mail/` or `/usr/bin/Mail`.
`xterm`	The absolute pathname to the X Window System terminal emulator, `xterm(1)`. Default path: `/usr/bin/X11/xterm`
`load_sensor`	A list of executable shell script paths or programs that are started by the execution daemons to retrieve site configurable load information.
`prolog`	The executable path of a shell script that is started before the execution of Sun Grid Engine jobs with the same environment setting as the Sun Grid Engine jobs to be started afterwards. This procedure automates the execution of general site specific tasks like the preparation of temporary file systems with the need for the same context information as the job. For a complete list of special variables that can be used with `prolog`, see the `sge_conf(5)` man page. Default value: NONE
`epilog`	The executable path of a shell script that is started after the execution of Sun Grid Engine jobs with the same environment setting as the Sun Grid Engine jobs that have just completed. This procedure automates the execution of general site specific tasks like the cleaning up of temporary file systems with the need for the same context information as the job. Default value: NONE
`shell_start_mode`	Deprecated.
`login_shells`	A list of the executable names of the command interpretors that should started as login shells. `login_shells` a global configuration parameter only. Default value: `sh,csh,tcsh,ksh`
`min_uid`	Places a minimum value on user IDs that may use the cluster. `min_uid` is a global configuration parameter only. Default value: `0`
`min_gid`	Please a minimum value on group IDs that may use the cluster. `min_gid` is a global configuration parameter only. Default value: `0`
`user_lists`	A list of user access lists that grant users access to the cluster. `user_lists` is a global configuration parameter only. Default value: NONE
`xuser_lists`	A list of user access lists that deny users access to the cluster. `xuser_lists` is a global configuration parameter only. Default value: NONE
`projects`	A list of all projects that are granted access to Sun Grid Engine. `projects` is a global configuration parameter only. Default value: NONE
`xprojects`	A list of all projects that are denied access to Sun Grid Engine. `xprojects` is a global configuration parameter only. Default value: NONE
`enforce_project`	If set to `true`, users are required to request a project whenever submitting a job. `enforce_project` is a global configuration parameter only. Default value: `false`

enforce_user	If set to true, a user must exist to allow for job submission. If set to auto, a user object for the submitting user will automatically be created. enforce_user is a global configuration parameter only. Default value: false
load_report_time	The time interval between system load reports. Note: Reporting load too frequently may block the master daemon, especially if your cluster contains a large number of execution hosts. Default value: 00:00:40
max_unheard	The amount of time (in seconds) that the master daemon could not contact or was not contacted by a host's execution daemon. If the prescribed amount of time passes, all queues residing on that particular host are set to status unknown. max_unheard is a global configuration parameter only. Default value: 00:05:
reschedule_unknown	Determines whether the jobs on a host set to unknown should be rescheduled and then sent to other hosts. reschedule_unknown controls the time (in hours) that Sun Grid Engine will wait to reschedule jobs after a host becomes unknown. Default value: 00:00:00
finished_jobs	Deprecated.
gid_range	A list of range expressions that identify processes belonging to the same job. Since there is no default value, the administrator will need to assign one.
qlogin_command	Executed on the client side of a qlogin request. Usually, this is the builtin qlogin mechanism.
qlogin_daemon	Specifies the mechanism that is to be started on the server side of a qlogin request. Usually, this is the builtin mechanism. It is also possible to configure an external executable by specifying the full qualified pathname.
rlogin_command	Executed on the client side of a qrsh request without a command argument to be executed remotely. Usually, this is the builtin mechanism. If no value is given, a special Grid Engine component is used. The command is automatically started with the target host and port number as parameters. The Grid Engine rlogin client has been extended to accept and use the port number argument. You can only use clients, such as ssh, which also understand this syntax.
rlogin_daemon	Specifies the mechanism that is to be started on the server side of a qrsh request without a command argument to be executed remotely. Usually, this is the builtin mechanism.
rsh_command	Executed on the client side of a qrsh request with a command argument to be executed remotely. Usually, this is the builtin mechanism. If no value is given, a specialized Grid Engine component is used. The command is automatically started with the target host and port number as parameters similar to those required for telnet(1) plus the command with its arguments to be executed remotely. The Grid Engine rsh client has been extended to accept and use the port number argument. You can only used clients, such as ssh, which can understand this syntax.
rsh_daemon	Specifies the mechanism that is to be started on the server side of a qrsh(1) request with a command argument to be executed remotely. Usually, this is the builtin mechanism. If no value is given, a specialized Grid Engine component is used.
max_aj_instances	Defines the maximum amount of array tasks that can scheduled to run simultaneously per array job. An instance of an array task will be created within the master daemon when it gets a start order from the scheduler. The instance will be destroyed when the array task finishes. This feature is most useful for very large clusters and very large array jobs. max_aj_instances is a global configuration parameter only. Default value: 2000
max_aj_tasks	Defines the maximum number of array job tasks within an array job. The master daemon will reject all array job submissions that request more than max_aj_tasks array job tasks. Default value: 75000
max_u_jobs	The number of active jobs, which each user can have in the system simultaneously. A value greater than 0 defines the limit. max_u_jobs is a global configuration parameter only. Default Value: 0 (unlimited)
max_jobs	The number of active jobs simultaneously allowed in the Sun Grid Engine system. A value greater than 0 defines this limit. max_jobs is a global configuration parameter only. Default value: 0 (unlimited)
max_advance_reservations	The number of active advance reservations simultaneously allowed in Sun Grid Engine. A value greater than 0 defines this limit. max_advance_reservations is a global configuration parameter only. Default value: 0 (unlimited)

`auto_user_oticket`	The number of override tickets assigned to automatically created user objects. User objects are created automatically if the `enforce_user` attribute is set to `auto`. `auto_user_oticket` is a global configuration parameter only. Default value: `0`
`auto_user_fshare`	The number of functional shares assigned to automatically created user objects. User objects are created automatically if the `enforce_user` attribute is set to `auto`. `auto_user_fshare` is a global configuration parameter only. Default value: `0`
`auto_user_default_project`	The default project assigned to automatically created user objects. User objects are created automatically if the `enforce_user` attribute is set to `auto`. `auto_user_default_project` is a global configuration parameter only. Default value: `NONE`
`auto_user_delete_time`	The number of seconds of inactivity after which automatically created user objects will be deleted. User objects are created automatically if the `enforce_user` attribute is set to `auto`. If the user has no active or pending jobs for the specified amount of time, the object will automatically be deleted. A value of `0` can be used to indicate that the automatically created user object is permanent and should not be automatically deleted. `auto_user_delete_time` is a global configuration parameter only. Default value: `86400`
`delegated_file_staging`	This flag must be set to `true` when the `prolog` and `epilog` are ready for delegated file staging.
`reprioritize`	Deprecated.
`qmaster params`	A list of additional parameters that can be passed to the Sun Grid Engine master daemon. For a complete list of these parameters, see the `sge_conf(5)` man page.
`reporting_params`	A list of parameters that are used to define the behavior of reporting modules in the Sun Grid Engine master daemon. For a complete list of these parameters, see the `sge_conf(5)` man page.
`jsv_url`	This setting defines a server JSV instance which will be started and triggered by the sge_qmaster(8) process.
`jsv_allowed_mod`	If there is a server JSV script defined with jsv_url parameter, then all `qalter(1)` or `qmon(1)` modification requests for jobs are rejected by the master daemon. With the `jsv_allowed_mod` parameter, an administrator has the possibility to allow a set of switches which can then be used with clients to modify certain job attributes.

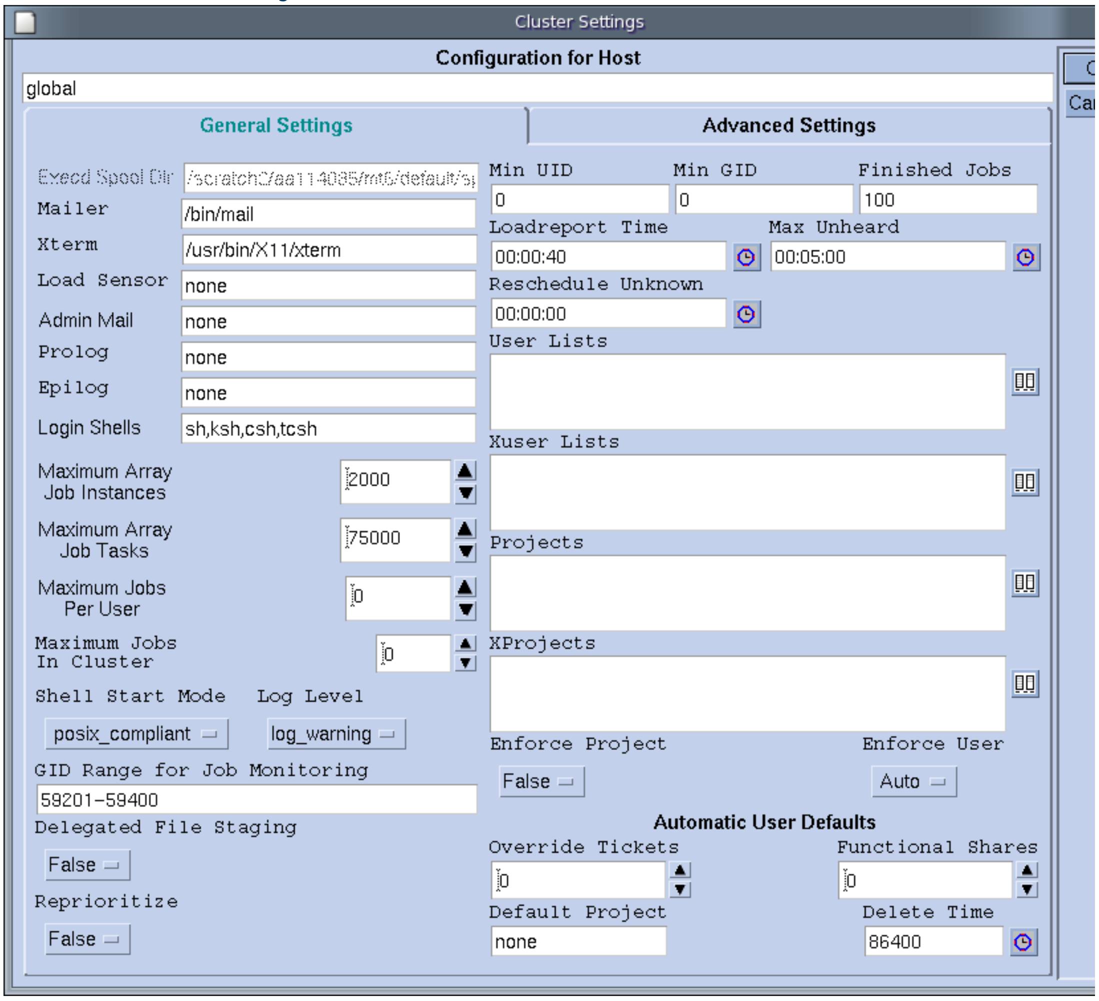
How to Configure Clusters With QMON

Note
Any change to a local host configuration supersedes the global configuration parameters.

1. On the QMON Main Control window, click the Cluster Configuration button, as shown below.
   
2. You can view all currently configured clusters and perform the following tasks from the Cluster Configuration dialog box:
   - To add a new local host configuration, do the following:
     a. Click the Add button.
        The Cluster Settings dialog box appears.
     b. Enter the name of the host and configuration information.
        The following host names are invalid, reserved, or otherwise not allowed to be used:
        - `global`
        - `template`
        - `all`
        - `default`
        - `unknown`
        - `none`
     c. When you finish making changes, click the OK button to save your changes and close the dialog box.

Click the Cancel button to close the dialog box without saving changes.

- To modify the global host configuration, do the following:
  a. Select the name `global` and then click the Modify button.
     The Cluster Settings dialog box appears, as shown in the following figure. You can modify all parameters. For more information, see Cluster Configuration Parameters.



Cluster Settings

Configuration for Host: global

General Settings

Execd Spool Dir	/scratch2/aa114085/mt6/default/s
Mailer	/bin/mail
Xterm	/usr/bin/X11/xterm
Load Sensor	none
Admin Mail	none
Prolog	none
Epilog	none
Login Shells	sh,ksh,csh,tcsh
Maximum Array Job Instances	2000
Maximum Array Job Tasks	75000
Maximum Jobs Per User	0
Maximum Jobs In Cluster	0
Shell Start Mode	posix_compliant
Log Level	log_warning
GID Range for Job Monitoring	59201-59400
Delegated File Staging	False
Reprioritize	False

Advanced Settings

Min UID	0
Min GID	0
Finished Jobs	100
Loadreport Time	00:00:40
Max Unheard	00:05:00
Reschedule Unknown	00:00:00
User Lists	
Xuser Lists	
Projects	
XProjects	
Enforce Project	False
Enforce User	Auto

Automatic User Defaults

Override Tickets	0
Functional Shares	0
Default Project	none
Delete Time	86400

  b. When you finish making changes, click the OK button to save your changes and close the dialog box.
     Click the Cancel button to close the dialog box without saving changes.
- To modify a local host configuration, do the following:
  a. Select a host name and then click the Modify button.
     The Cluster Settings dialog box appears. You can modify only those parameters that are feasible for local host changes. For more information, see Cluster Configuration Parameters.
  b. When you finish making changes, click the OK button to save your changes and close the dialog box.
     Click the Cancel button to close the dialog box without saving changes.

> Note
> The Advanced Settings tab shows a corresponding behavior, depending on whether you are modifying a configuration or are adding a new configuration. The Advanced Settings tab provides access to more rarely used cluster configuration parameters.

- To delete a local host, select the host whose configuration you want to delete, and then click Delete.

See the `sge_conf(5)` man page for a complete description of all cluster configuration parameters.

Configuring Hosts

For more information on hosts and daemons, see How the System Operates.

Note
The master host requires no further configuration other than that performed by the installation procedure. For information about how to initially set up the master host, see How to Install the Master Host. For information about how to configure dynamic changes to the master host, see How to Configure the Shadow Master Host below.

Task	User Interface	Description
How to Configure Execution Hosts	CLI or QMON	Learn how to display, add, modify, and delete execution hosts. An execution host is a system that has permission to run Grid Engine system jobs.
How to Configure Administration Hosts	CLI or QMON	Learn how to display, add, and delete administration hosts. An administration host is a system that has permission to carry out administrative activity for the Grid Engine system.
How to Configure Submit Hosts	CLI or QMON	Learn how to display, add, and delete submit hosts. A submit hosts is a system that can submit and control batch jobs only.
How to Configure Host Groups	CLI or QMON	Learn how to display, add, modify, and delete host groups. Host groups enable you to use a single name to refer to multiple hosts.
How to Kill Daemons	CLI or QMON	Learn how to kill the master daemon and execution daemons from the command line and how to kill execution daemons with QMON.
How to Restart Daemons	CLI	Learn how to restart the master and execution daemons from the command line.
How to Migrate the Master Daemon to Another Host	CLI	Learn how to migrate the master daemon to another host from the command line.
How to Migrate the Master Daemon to Another Host Using a Script	CLI	Learn how to migrate the master daemon to another host using a script.

How to Configure Execution Hosts From the Command Line

Before You Begin

Before you configure an execution host, you must first install the software on the execution host as described in How to Install Execution Hosts.

Note
Administrative or submit commands are allowed from execution hosts only if the execution hosts are also declared to be administration or submit hosts. See How to Configure Administration Hosts and How to Configure Submit Hosts.

Configuration Commands

To configure execution hosts from the command line, use the following arguments for the `qconf` command:

- To show a specific execution host configuration, type the following command:

```
qconf -se <hostname>
```

  The -se option (show execution host) shows the configuration of the specified execution host as defined in host_conf.

- To show a list of all configured execution hosts, type the following command:

```
qconf -sel
```

The -sel option (show execution host list) shows a list of hosts that are configured as execution hosts.

- To add an execution host, type the following command:

```
% qconf -ae [ <hostname> ]
```

The -ae option (add execution host) displays an editor that contains a configuration template for an execution host. The editor is either the default vi editor or the editor that corresponds to the EDITOR environment variable. If you specify exec-host, which is the name of an already configured execution host, the configuration of this execution host is used as a template. To configure the execution host, change and save the template. See the host_conf(5) man page for a detailed description of the template entries to be changed. Note: Before you configure an execution host, you must first install the software on the execution host as described in How to Install Execution Hosts.

- To modify an execution host, type the following command:

```
% qconf -me <hostname>
```

Note
If you would like to give all users access to an execution host, leave the default ACL setting at NONE. If the ACL is set to defaultdepartment, users cannot submit jobs to the project.

The -me option (modify execution host) displays an editor that contains the configuration file template for the specified execution host. The editor is either the default vi editor or the editor that corresponds to the EDITOR environment variable. To modify the execution host configuration, change and save the configuration file template. See the host_conf(5) man page for a detailed description of the template entries to be changed. For information on host configuration parameters, see Host Configuration Parameters.

- To modify an execution host from file, type the following command:

```
% qconf -Me <filename>
```

The -Me option (modify execution host) uses the content of filename as an execution host configuration template. The configuration in the specified file must refer to an existing execution host. The configuration of this execution host is replaced by the file content. This qconf option is useful for changing the configuration of offline execution hosts, for example, in cron jobs, as the -Me option requires no manual interaction. For information on host configuration parameters, see Host Configuration Parameters.

- To delete an execution host, type the following command:

```
% qconf -de <hostname>
```

The -de option (delete execution host) deletes the specified host from the list of execution hosts. All entries in the execution host configuration are lost.

For more information on how to configure hosts from file or modify many objects at a time, see Using qconf.

How to Configure Execution Hosts With QMON

Before You Begin

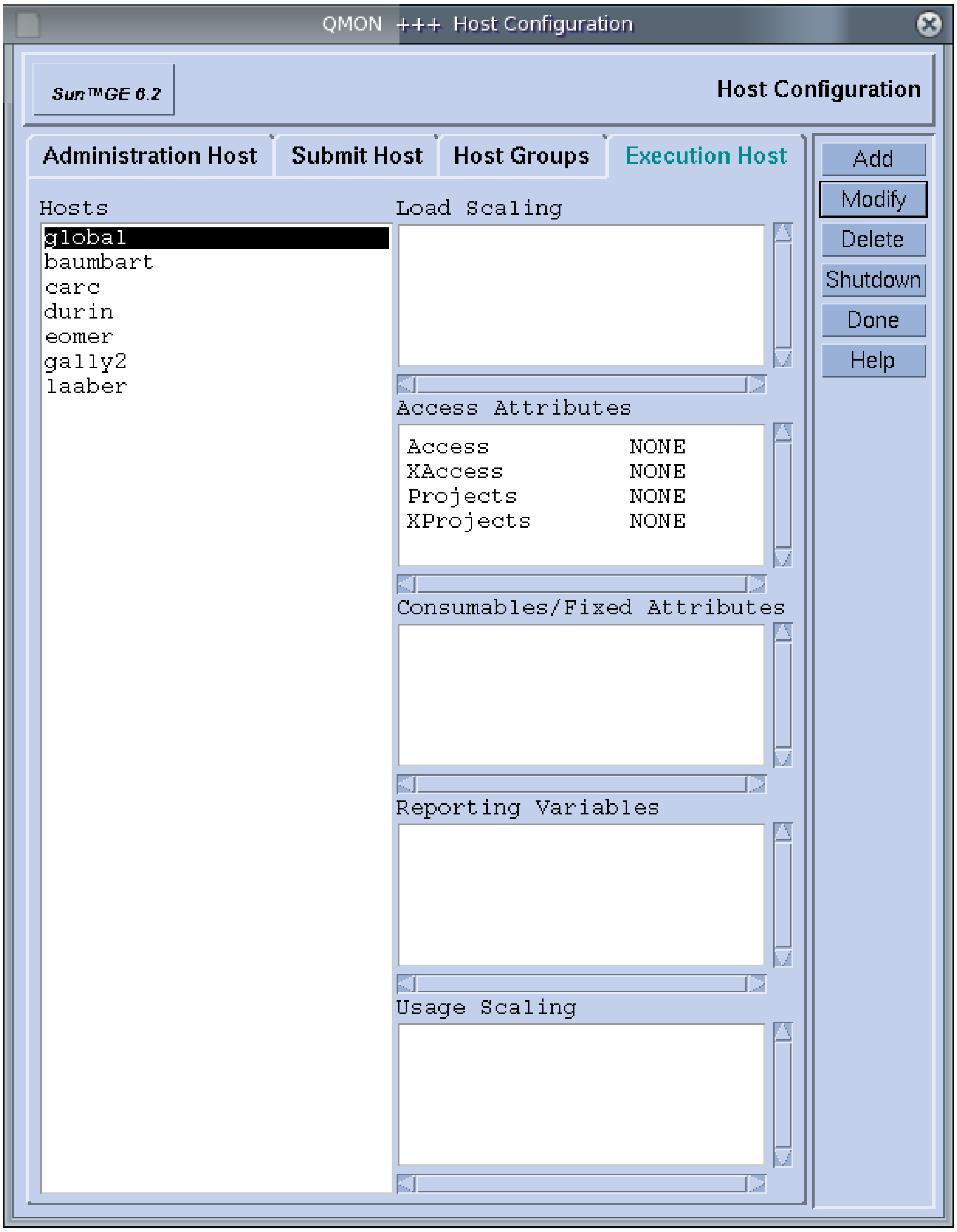
Before you configure an execution host, you must first install the software on the execution host as described in How to Install Execution Hosts.

Note
Administrative or submit commands are allowed from execution hosts only if the execution hosts are also declared to be administration or submit hosts. See How to Configure Administration Hosts and How to Configure Submit Hosts.

Steps

1. Click the Host Configuration button on the QMON Main Control window.

2. Click the Execution Host tab.



```
global
baumbart
carc
durin
eomer
gally2
laaber
```

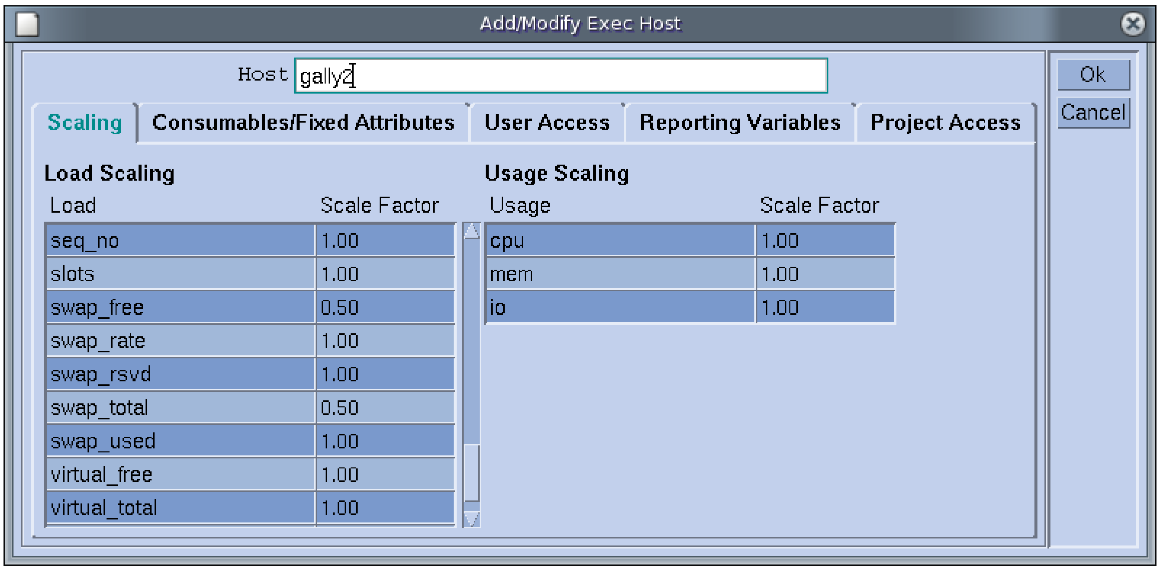
```
Access          NONE
XAccess         NONE
Projects        NONE
XProjects       NONE
```

Note the following in the Execution Host tab:

- The Hosts list displays the execution hosts that are already defined.
- The Load Scaling list displays the currently configured load-scaling factors for the selected execution host. See Configuring Load Parameters for information about load parameters.

- The Access Attributes list displays access permissions. See Managing User Access for information about access permissions.
- The Consumables/Fixed Attributes list displays resource availability for consumable and fixed resource attributes associated with the host. See Configuring Resource Attributes for information about resource attributes.
- The Reporting Variables list displays the variables that are written to the reporting file when a load report is received from an execution host. See Defining Reporting Variables for information about reporting variables.
- The Usage Scaling list displays the current scaling factors for the individual usage metrics CPU, memory, and I/O for different machines. Resource usage is reported by `sge_execd` periodically for each currently running job. The scaling factors indicate the relative cost of resource usage on the particular machine for the user or project running a job. These factors could be used, for instance, to compare the cost of a second of CPU time on a 400 MHz processor to that of a 600 MHz CPU. Metrics that are not displayed in the Usage Scaling list have a scaling factor of 1.

3. You can show, add, modify, or delete an execution host from the Execution Host Tab.
   - To show an execution host, select a host name from the Host list.
     The execution host's attributes are displayed.
   - To add or modify an execution host, click the Add button and then type its name in the Host field or click the Modify button. The Add/Modify Exec Host dialog box appears as shown in the following figure. For more information, see Host Configuration Parameters.



Host gally2

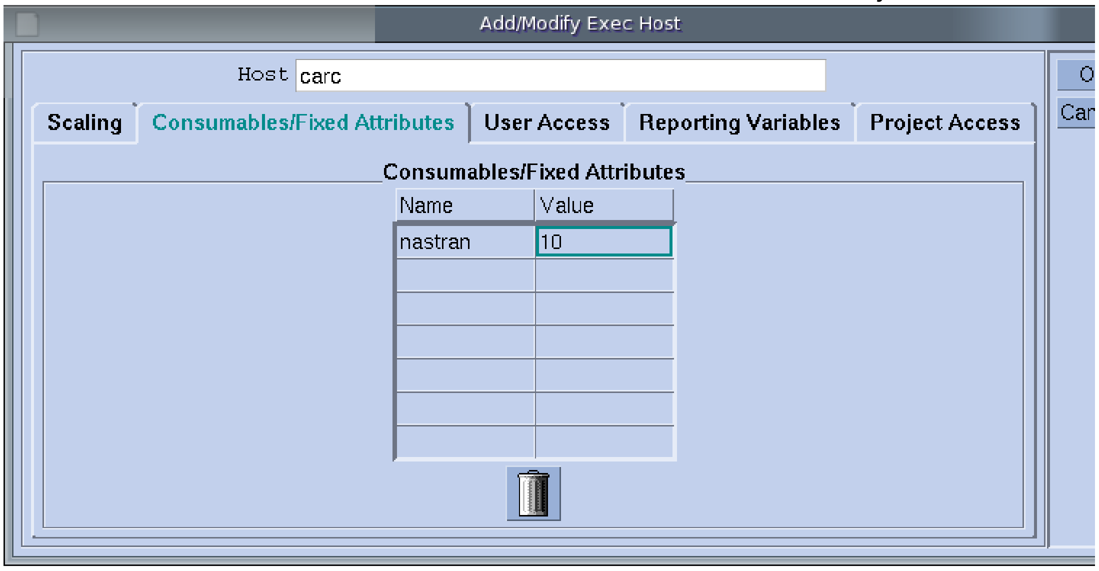
Load Scaling

Load	Scale Factor
seq_no	1.00
slots	1.00
swap_free	0.50
swap_rate	1.00
swap_rsvd	1.00
swap_total	0.50
swap_used	1.00
virtual_free	1.00
virtual_total	1.00

Usage Scaling

Usage	Scale Factor
cpu	1.00
mem	1.00
io	1.00

   - To define scaling factors, click the Scaling tab.
     - The Load column of the Load Scaling table lists all available load parameters. The Scale Factor column lists the corresponding definitions of the scaling. You can edit the Scale Factor column. Valid scaling factors are positive floating-point numbers in fixed-point notation or scientific notation.
     - The Usage column of the Usage Scaling table lists the current scaling factors for the usage metrics CPU, memory, and I/O. The Scale Factor column lists the corresponding definitions of the scaling. You can edit the Scale Factor column. Valid scaling factors are positive floating-point numbers in fixed-point notation or scientific notation.
   - To define the resource attributes to associate with the host, click the Consumables/Fixed Attributes tab.
     The Consumables/Fixed Attributes table lists all resource attributes for which a value is currently defined.



Host carc

Consumables/Fixed Attributes

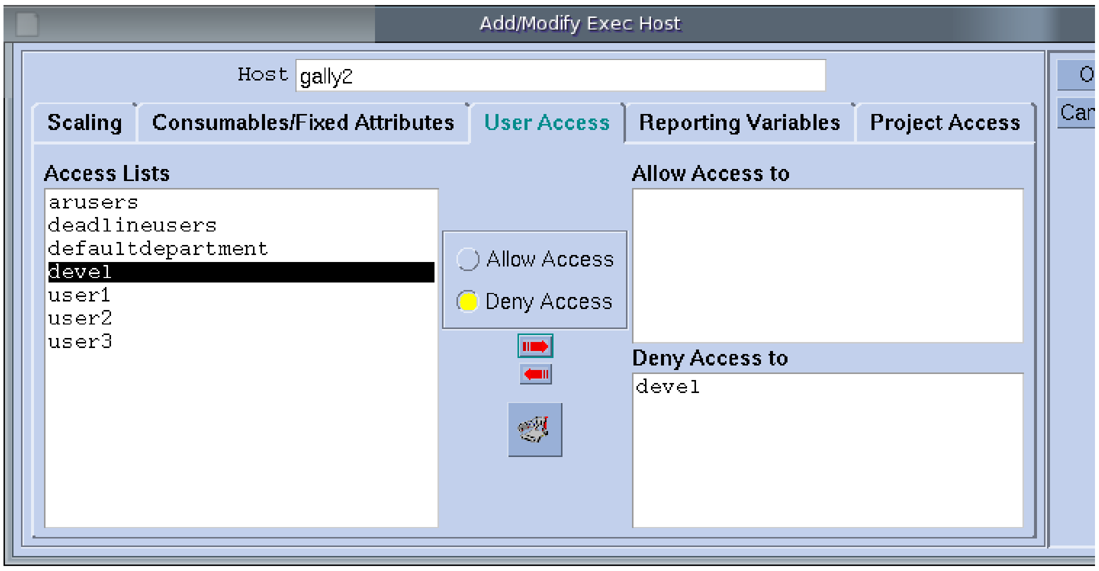
Name	Value
nastran	10

Tip
Use the Complex Configuration dialog box if you need more information about the current complex configuration, or if you want to modify it. For details about complex resource attributes, see Configuring Resource Attributes.

You can enhance the list by clicking either the Name or the Value column name. The Attribute Selection dialog box appears, which includes all resource attributes that are defined in the complex.

- To add an attribute to the Consumables/Fixed Attributes table, select the attribute, and then click OK.
- To modify an attribute value, double-click a Value field, and then type a value.
- To delete an attribute, select the attribute, and then press Control-D or click mouse button 3.
  Click Ok to confirm that you want to delete the attribute.

- To define user access permissions to the execution host based on previously configured user access lists, click the User Access tab.



Host: gally2

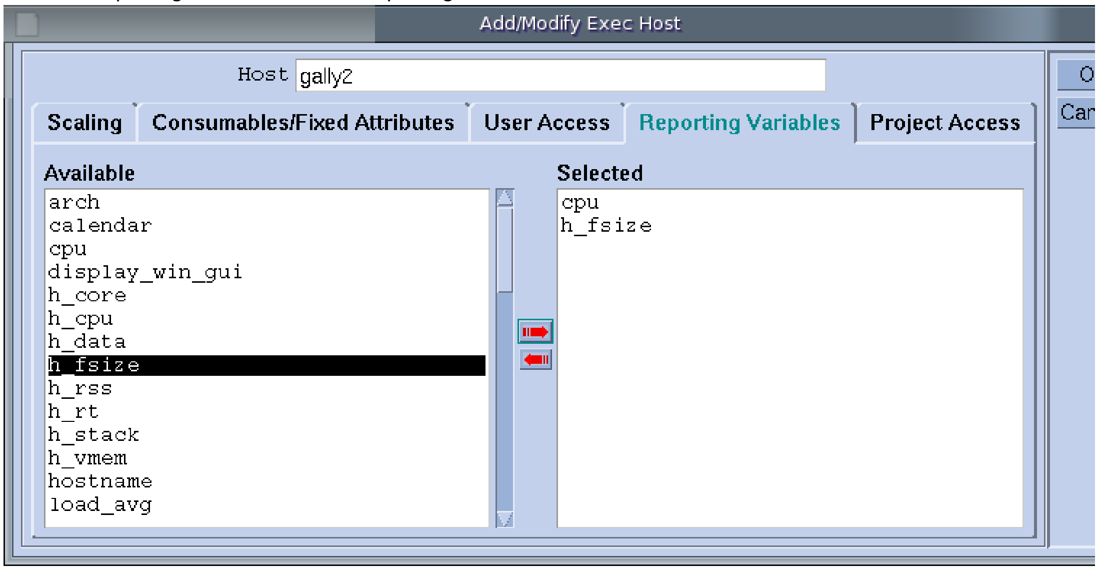
Access Lists:
arusers
deadlineusers
defaultdepartment
devel
user1
user2
user3

[Allow Access] [✓ Deny Access]

Allow Access to:

Deny Access to:
devel

- To define reporting variables, click the Reporting Variables tab.



Host: gally2

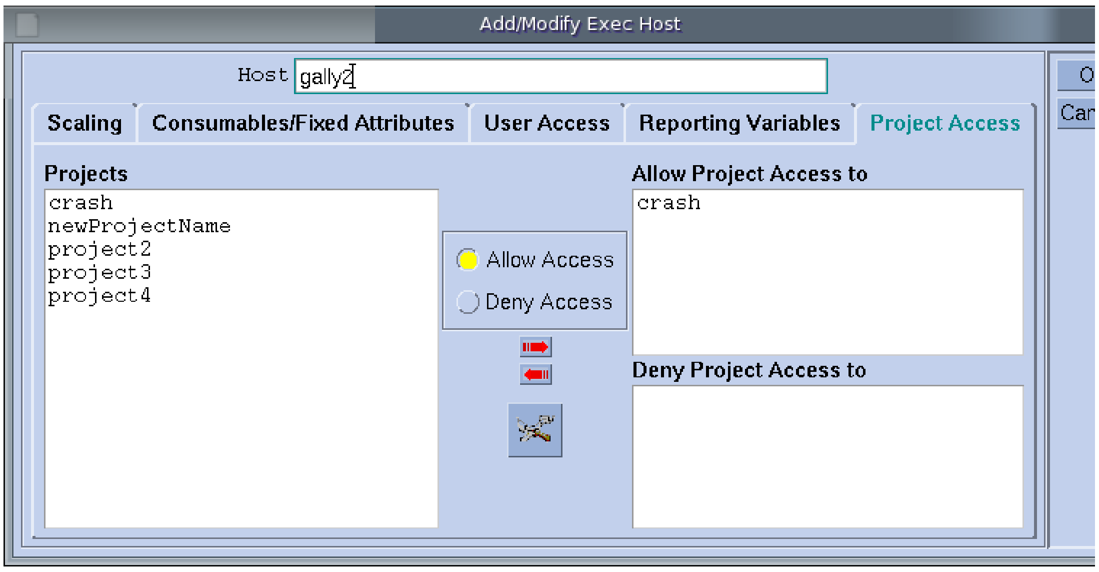
Available:
arch
calendar
cpu
display_win_gui
h_core
h_cpu
h_data
h_fsize
h_rss
h_rt
h_stack
h_vmem
hostname
load_avg

Selected:
cpu
h_fsize

The Available list displays all the variables that can be written to the reporting file when a load report is received from the execution host.

- To add a variable to the Selected list, select a reporting variable from the Available list, and then click the red right arrow.
- To remove a reporting variable from the Selected list, select the variable, and then click the left red arrow.

- To define project access permissions to the execution host based on previously configured projects, click the Project Access tab.



- To delete an execution host, select the execution host name from the Hosts list and then click the Delete button.

After you have configured the execution hosts, see Monitoring Hosts.

How to Configure Administration Hosts From the Command Line

To configure administration hosts from the command line, use the following arguments for the `qconf` command:

- To show a list of all currently configured administration hosts, type the following command:

  ```
  % qconf -sh
  ```

  The `-sh` option (show administration hosts) displays a list of all currently configured administration hosts.

- To add an administration host, type the following command:

  ```
  % qconf -ah <hostname>
  ```

  The `-ah` option (add administration host) adds the specified host to the list of administration hosts.

- To delete an administration host, type the following command:

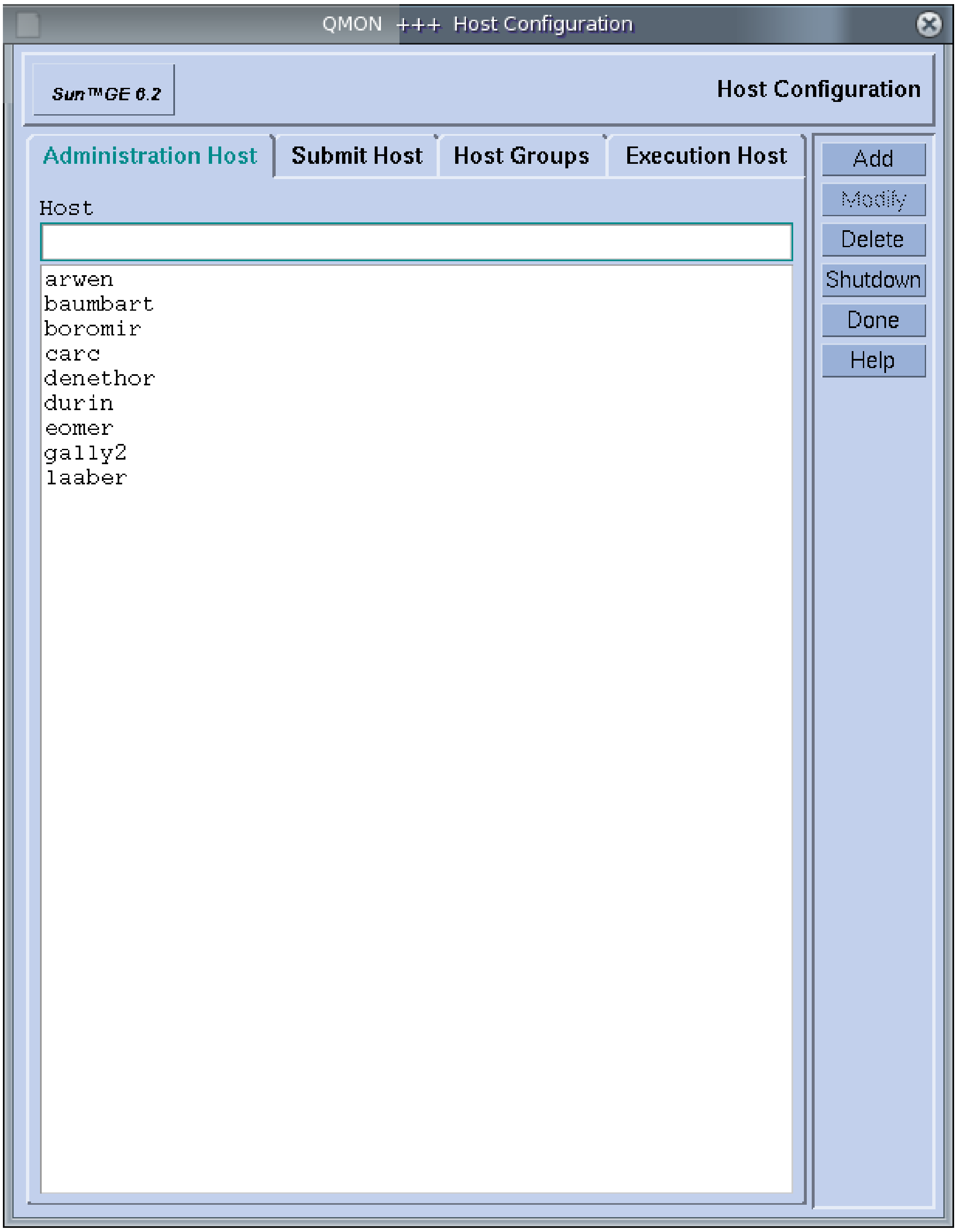
  ```
  % qconf -dh <hostname>
  ```

  The `-dh` option (delete administration host) deletes the specified host from the list of administration hosts.

For more information on how to configure hosts from file or modify many objects at a time, see Using qconf and the `qconf(1)` man page.

How to Configure Administration Hosts With QMON

1. On the QMON Main Control window, click the Host Configuration button.
   The Host Configuration dialog box appears and displays the Administration Host tab as shown in the following figure.



```
arwen
baumbart
boromir
carc
denethor
durin
eomer
gally2
laaber
```

Note
The Administration Host tab is displayed by default when you click the Host Configuration button for the first time.

2. From the Host Configuration dialog box, you can choose from the following functions:
   - To add a new administration host, type its name in the Host field, and then click Add, or press the Return key.
   - To delete an administration host, select the administrative host name from the list, and then click Delete.

How to Configure Submit Hosts From the Command Line

Note
No administrative commands are allowed from submit hosts unless the hosts are also declared to be administration hosts. See How to Configure Administration Hosts for more information.

To configure submit hosts from the command line, use the following arguments for the `qconf` command:

- To show a list of the names of all currently configured submit hosts, type the following command:

  ```
  % qconf -ss
  ```

  The `-ss` option (show submit hosts) displays a list of the names of all currently configured submit hosts.

- To add a host to the list of submit hosts, type the following command:

  ```
  % qconf -as <hostname>
  ```

  The `-as` option (add submit host) adds the specified host to the list of submit hosts.

- To delete a specified host from the list of submit hosts, type the following command:

  ```
  % qconf -ds <hostname>
  ```

  The `-ds` option (delete submit host) deletes the specified host from the list of submit hosts.

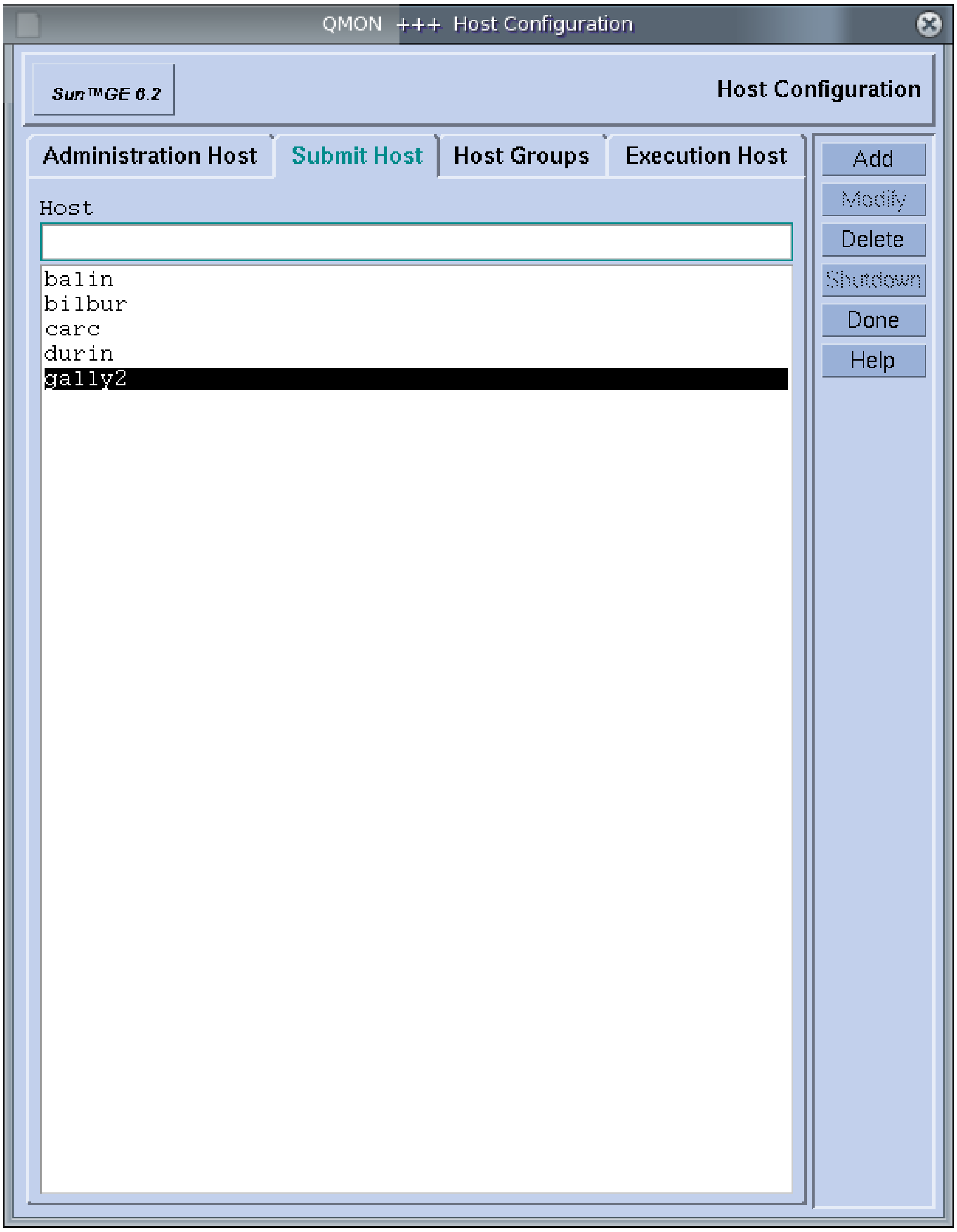
For more information on how to configure hosts from file or modify many objects at a time, see Using qconf or the `qconf(1)` man page.

How to Configure Submit Hosts With QMON

Note
No administrative commands are allowed from submit hosts unless the hosts are also declared to be administration hosts. See How to Configure Administration Hosts for more information.

1. On the QMON Main Control window, click the Host Configuration button.
   The Host Configuration button is shown below.

2. Click the Submit Host tab in the Host Configuration dialog box.
   The Submit Host tab is show below.



```
balin
bilbur
carc
durin
gally2
```

3. You can view all currently configured submit hosts and perform the following tasks from the Submit Host tab:
    - To add a submit host, type the submit host name in the Host field, and then click Add or press the Return key.
    - To delete a submit host, select the submit host name from the list, and then click Delete.

How to Configure Host Groups From the Command Line

Host groups enable you to use a single name to refer to multiple hosts. A host group can include other host groups as well as multiple individual hosts. Host groups that are members of another host group are subgroups of that host group.

For example, you might define a host group called `@bigMachines` that includes the following members:

- `@solaris64`
- `@solaris32`
- `fangorn`
- `balrog`

The initial @ sign indicates that the name is a host group. The host group `@bigMachines` includes all hosts that are members of the two subgroups `@solaris64` and `@solaris32`. `@bigMachines` also includes two individual hosts, `fangorn` and `balrog`.

Commands

To configure host groups from the command line, use the following arguments for the `qconf` command:

- To show a host group configuration, type the following command:

  ```
  % qconf -shgrp <hostgroupname>
  ```

  The `-shgrp` option (show host group) shows the configuration of the specified host group.

- To show the host group as a tree, type the following command:

  ```
  % qconf -shgrp_tree <hostgroupname>
  ```

  The `-shgrp_tree` option (show host group as a tree) shows the configuration of the specified host group and its sub-hostgroups as a tree.

- To show a host group with a resolved host list, type the following command:

  ```
  % qconf -shgrp_resolved <hostgroupname>
  ```

  The `-shgrp_resolved` option (show host group with resolved host list) shows the configuration of the specified host group with a resolved host list.

- To show the host group list, type the following command:

  ```
  % qconf -shgrpl
  ```

  The `-shgrpl` option (show host group list) displays a list of all host groups.

- To add a host group to the list of host groups, type the following command:

  ```
  % qconf -ahgrp <hostgroupname>
  ```

  The `ahgrp` option (add host group) adds a new host group to the list of host groups. See the `hostgroup(5)` man page for a detailed description of the configuration format.

- To add a host group from a file, type the following command:

  ```
  % qconf -Ahgrp <filename>
  ```

  The `-Ahgrp` option (add host group from a file) displays an editor that contains a host group configuration defined in filename. The editor is either the default `vi` editor or the editor that corresponds to the `$EDITOR` environment variable. To configure the host group, change and save the configuration file template.

- To modify a host group, type the following command:

```
% qconf -mhgrp <hostgroupname>
```

The `-mhgrp` option (modify host group) displays an editor that contains the configuration of the specified host group as template. You can modify the `groupname`, add a host to the `hostlist`, add a host group as a subgroup, and remove a host or host group. The editor is either the default `vi` editor or the editor that corresponds to the `EDITOR` environment variable. To modify the host group configuration, change and save the configuration file template.

- To modify a host group from a file, type the following command:

```
% qconf -Mhgrp <filename>
```

The `-Mhgrp` option (modify host group from a file) uses the content of filename as host group configuration template. The configuration in the specified file must refer to an existing host group. You can modify the `groupname`, add a host to the `hostlist`, add a host group as a subgroup, and remove a host or host group. The configuration of this host group is replaced by the file content.

- To delete a host group, type the following command:

```
qconf -dhgrp <hostgroupname>
```

The `-dhgrp` option (delete host group) deletes the specified host group from the list of host groups. All entries in the host group configuration are lost.

For more information on how to configure host groups from file or modify many objects at a time, see Using qconf or the `qconf(1)` man page.

How to Configure Host Groups With QMON

Host groups enable you to use a single name to refer to multiple hosts. A host group can include other host groups as well as multiple individual hosts. Host groups that are members of another host group are subgroups of that host group.

For example, you might define a host group called `@bigMachines` that includes the following members:

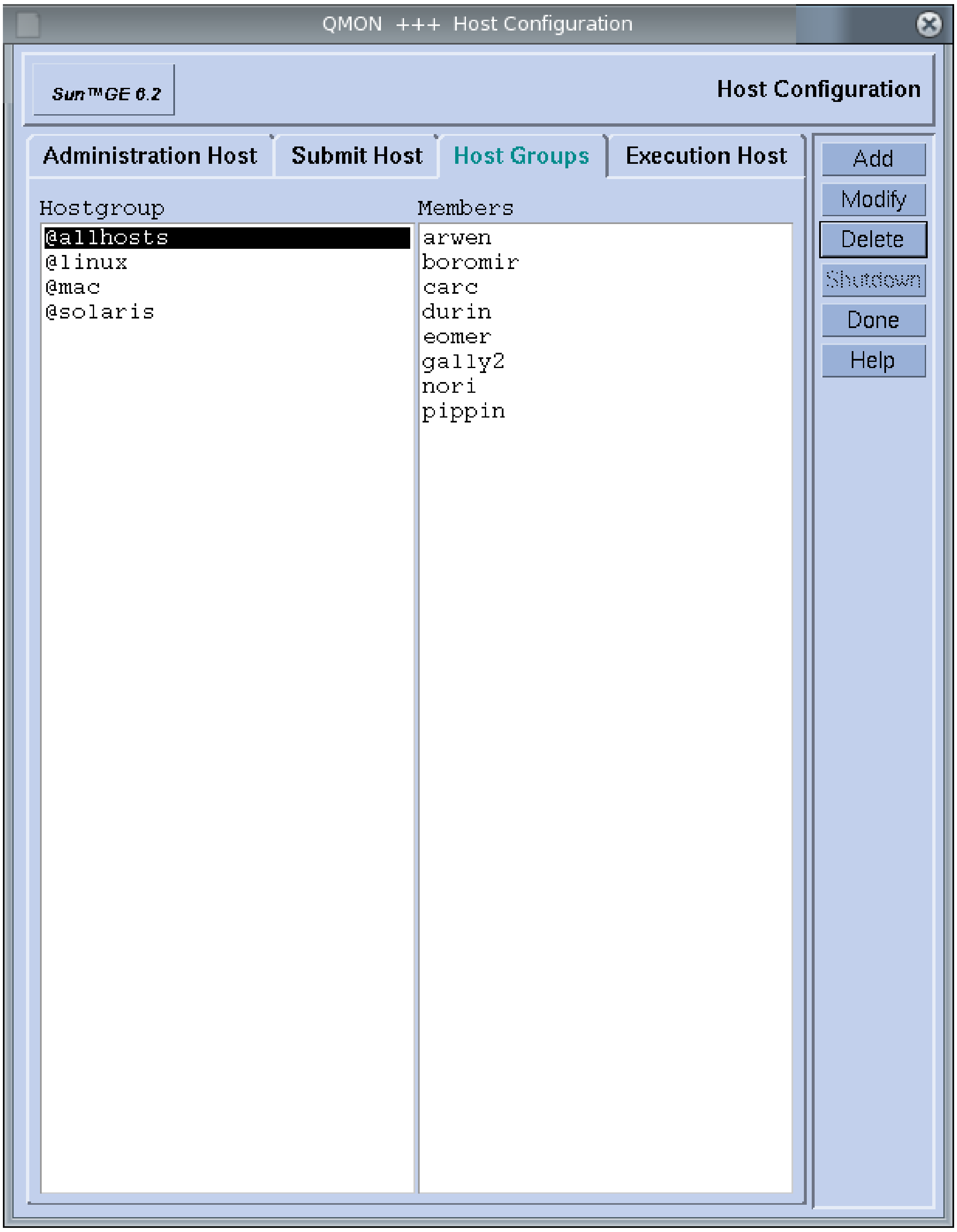
- `@solaris64`
- `@solaris32`
- `fangorn`
- `balrog`

The initial @ sign indicates that the name is a host group. The host group `@bigMachines` includes all hosts that are members of the two subgroups `@solaris64` and `@solaris32`. `@bigMachines` also includes two individual hosts, `fangorn` and `balrog`.

Steps

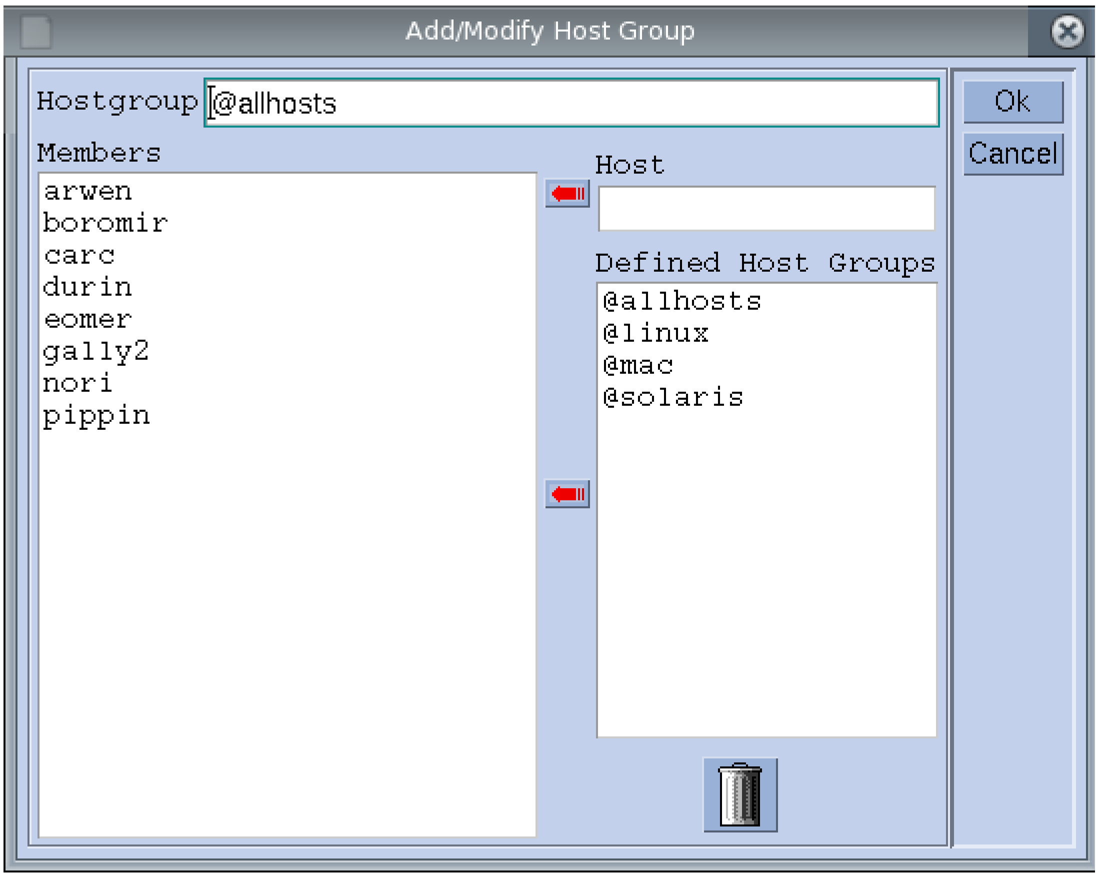
1. On the QMON Main Control window, click the Host Configuration button.
   The Host Configuration dialog box appears.

2. Click the Host Groups tab.
   The Host Groups tab is shown below.



```
Hostgroup        Members
@allhosts        arwen
@linux           boromir
@mac             carc
@solaris         durin
                 eomer
                 gally2
                 nori
                 pippin
```

3. You can perform the following tasks from the Host Groups tab:
   - To add a host group, do the following:
     a. Click Add.
        The Add/Modify Host Group dialog box appears, as shown in the following figure.



b. Type a host group name in the Hostgroup field.
   The host group name must begin with an @ sign.
c. Click Ok to save your changes.
   Click Cancel to leave the dialog box without saving your changes.

- To modify a host group, do the following:
  a. Select the host group in the Hostgroup list and click Modify.
  b. From the Add/Modify Host Group dialog box, you can do the following:
     - To add a host to the Members list for the selected host group, type the host name in the Host field and click the red arrow.
     - To add a host group as a subgroup, select a host group name from the Defined Host Groups list and click the red arrow.
     - To remove a host or a host group, select it from the Members list and click the trash icon.
  c. Click Ok to save your changes.
     Click Cancel to leave the dialog box without saving your changes.
- To delete a host group, select the host group and click Delete.

The page Configuring a Shadow Master Host From the Command Line does not exist.
The page How to Configure a Shadow Master Host From the Command Line does not exist.

Host Configuration Parameters

`host_conf` reflects the format of the template file for the execution host configuration.

For detailed information about these parameters, see the `host_conf(5)` man page.

Parameter	Description
`hostname`	Name of the execution host.
`load_scaling`	A list of scaling values to be applied to each or part of the load values being reported by the execution daemon on the host and being defined in the cluster global host complex. The load scaling factors are intended to level hardware or operating system specific differences between execution hosts.

`complex_values`	Defines quotas for resource attributes managed via this host.
`load_values`	A list of load values. This entry cannot be configured but is only displayed in reaction to the `qconf -se` command.
`processors`	Deprecated.
`usage_scaling`	Equivalent to `load_scaling`, the only valid attributes to scaled are `cpu` for CPU time consumption, `mem` for Memory consumption aggregated over the lifetime of jobs and `io` for data transferred via any I/O devices. Default: NONE (no scaling)
`user_lists`	A list of user access lists that grant users access to the host. Default value: NONE
`xuser_lists`	A list of user access lists that deny users access to the host. Default value: NONE
`projects`	A list of all projects that are granted access to the host. Default value: NONE
`xprojects`	A list of all projects that are denied access to the host. Default value: NONE
`report_variables`	A list of variables that are written to the reporting file when a load report arrives from an execution host.

How to Kill Daemons From the Command Line

Before You Begin

Note
You must have manager or operator privileges to use these commands. See Managing Users Access for more information about manager and operator privileges.

To keep scheduled and running jobs from being affected by a shutdown procedure, you must first disable all cluster queues, queue instances, and queue domains. Then, wait for all active jobs to finish.

To disable cluster queues, queue instances, and queue domains, type the following command:

```
% qmod -d {<clusterqueuename> | <queueinstancename> | <queuedomainname>}
```

Use the `qmod -d` command for each cluster queue, queue instance, or queue domain before you run the `qconf` sequence described below. The `qmod -d` command prevents new jobs from being scheduled to the disabled queue instances.

Next, you should wait until no jobs are running in the queue instances before you kill the daemons. For information about cluster queues, queue instances, and queue domains, see About Configuring Queues.

Kill Commands

To kill daemons from the command line, use the following arguments for the `qconf` command:

- To shut down the master daemon, type the following command:

  ```
  % qconf -km
  ```

  The `qconf -km` command forces the `sge_qmaster` process to terminate.

- To shut down the executive daemons and not cancel active jobs, type the following command:

  ```
  % qconf -ke {<hostname>[,...] | all}
  ```

The `qconf -ke` command shuts down the execution daemons. However, it does not cancel or lose active jobs. Jobs that finish while no `sge_execd` is running on a system are not reported to `sge_qmaster` until `sge_execd` is restarted. Use a comma-separated list of the execution hosts you want to shut down, or specify `all` to shut down all execution hosts in the cluster.

- To shut down the execution daemons and kill all active jobs, type the following command:

```
% qconf -kej {<hostname>[,...] | all}
```

The `qconf -kej` command kills all currently active jobs and brings down all execution daemons. Use a comma-separated list of the execution hosts that you want to shut down, or specify `all` to shut down all execution hosts in the cluster.

For more information on how to configure hosts from file or modify many objects at a time, see Using qconf or the `qconf(1)` man page.

How to Kill Daemons With QMON

Before You Begin

To keep scheduled and running jobs from being affected by a shutdown procedure, you must first disable all cluster queues, queue instances, and queue domains. Then, wait for all active jobs to finish.

To disable cluster queues, queue instances, and queue domains, do the following:

1. On the QMON Main Control window, click the Queue Control button.
   The Cluster Queue dialog box appears.

2. You can do the following from the Cluster Queue dialog box:
   - To disable cluster queues, click the Cluster Queues tab, select a cluster queue and then click the Disable Button.
   - To disable queue instances and queue domains, click the Queue Instances tab, select a queue instance or queue domain and then click the Disable button.

3. Click the Ok button to save your changes.
   Click Cancel to close the dialog box without saving any changes.

4. Wait for all active jobs to finish.

Steps

To kill the master daemon, you must use the command line interface. For more information, see How to Kill Daemons From the Command Line.

To kill an execution daemon With QMON, do the following:

1. On the QMON Main Control window, click the Host Configuration button.

2. Click the Execution Host tab.

3. In the Execution Host dialog box, select a host, and click Shutdown.

For information on how to restart execution daemons, see How to Restart Daemons From the Command Line.

How to Restart Daemons From the Command Line

To restart daemons from the command line, do the following:

1. Log in as root on the machine on which you want to restart Grid Engine system daemons.

2. type one of the following commands:

- To restart the **`sge_master`**, type the following command:

```
% $SGE_ROOT/$SGE_CELL/common/sgemaster
```

- To restart the **`sge_execd`**, type the following command:

```
% $SGE_ROOT/$SGE_CELL/common/sgeexecd
```

How to Migrate qmaster to Another Host From the Command Line

1. Stop the master daemon on the current master host.
   Type the following command:

```
qconf -km
```

2. Edit the **`$SGE_ROOT/$SGE_CELL/common/act_qmaster`** file according to the following guidelines:
   - Confirm the new master host's name. To get the new master host name, type the following command on the new master host:

```
$SGE_ROOT/utilbin/$SGE_ARCH/gethostname
```

   - In the `act_qmaster` file, replace the current host name with the new master host's name returned by the `gethostname` utility.

3. On the new master host, start **`sge_qmaster`**:

```
$SGE_ROOT/$SGE_CELL/common/sgemaster
```

How to Migrate qmaster to Another Host Using a Script

Note
Because the spooling database cannot be located on an NFS-mounted file system, the following procedure requires that you use the Berkeley DB RPC server for spooling. If you configure spooling to a local file system, you must transfer the spooling database to a local file system on the new `sge_qmaster` host.

1. Check that the new master host has read/write access.
   The new master host must have read/write access to the `qmaster` spool directory and common directory as does the current master. If the administrative user is the root user (check the global cluster configuration for the setting of `admin_user`), you should verify that the root user can create files in these directories under the root user name.

2. Run the migration script on the new master host.

On the new master host, type the following command as the root user:

```
# $SGE_ROOT/$SGE_CELL/common/sgemaster -migrate
```

This command stops `sge_qmaster` on the old master host and starts it on the new master host. The master host name listed in the file `$SGE_ROOT/$SGE_CELL/common/act_qmaster` is automatically changed to the new master host. If `qmaster` is not running, warning messages will appear and a delay of about one minute will occur until `qmaster` is started on the new host.

3. Modify the `shadow_masters` file if necessary.
   a. Check if the `$SGE_ROOT/$SGE_CELL/common/shadow_masters` file exists.
      If the file exists, you can add the new `qmaster` host to this file and remove the old master host, depending on your requirements.
   b. Then stop and restart the **sge_shadowd** daemons by issuing the following commands on the respective machines:

```
$SGE_ROOT/$SGE_CELL/common/sgemaster -shadowd stop
$SGE_ROOT/$SGE_CELL/common/sgemaster -shadowd start
```

Important Notes About Migration

The migration procedure migrates to the host on which the `sgemaster -migrate` command is issued. If the file `primary_qmaster` exists, any subsequent calls of `sgemaster` on the machine contained in the `primary_qmaster` file will cause a migration back to that machine. To avoid such a situation, change or delete the `$SGE_ROOT/$SGE_CELL/common/primary_qmaster` file.

> Note
> Existence of the `primary_qmaster` file does not imply that the `qmaster` is actually running.

Although jobs may continue to run during the migration procedure, the grid should be inactive. While the migration is taking place, any Grid Engine commands that are running, such as `qsub` or `qstat`, will return an error.

If the current `qmaster` is down, the scheduler will not shut down until it times out waiting for contact with the `qmaster`.

The `shadow_masters` file has no direct effect on the migration procedure. This file only exists if one or more shadow masters have been configured. For more information on how to set up shadow masters, see How to Install the Shadow Master Host.

Managing User Access

You need to perform the following tasks to set up a user for the Grid Engine system:

- Assign required logins – Users must have identical accounts on all hosts that they use within the Sun Grid Engine system. No login is required on the machine where `sge_qmaster` runs. For more information, see Configuring User Access.
- Set access permissions – The Grid Engine software enables you to restrict user access to the entire cluster, to queues, and to parallel environments. For more information, see Configuring User Access. In addition, you can grant users permission to suspend or enable certain queues. See How to Configure User Access Parameters for more information.
- Declare a Grid Engine System user – To add users to the share tree or to define functional or override policies for users, you must declare those users to the Grid Engine system. For more information, see Managing Policies and Configuring User Access.
- Set up project access – If projects are used for the definition of share-based, functional, or override policies, you should give the user access to one or more projects. Otherwise the user's jobs might end up in the lowest possible priority class, which would result in the jobs having access to very few resources. See Managing Policies for more information.
- Set file access restrictions – Users of the Grid Engine system must have read access to the directory `$SGE_ROOT/$SGE_CELL/common` . Before a job starts, the execution daemon creates a temporary working directory for the job and changes ownership of the directory to the job owner. The execution daemon runs as root. The temporary directory is removed as soon as the job finishes. The temporary working directory is created under the path defined by the queue configuration parameter `tmpdir`. See the `queue_conf(5)` man page for more information. Make sure that temporary directories can be created under the `tmpdir` location. The directories should be set to Grid Engine system user ownership. Users should be able to write to the temporary directories.
- Set up site dependencies – By definition, batch jobs do not have a terminal connection. Therefore UNIX commands like `stty` in the

command interpreter's startup resource file (for example, `.cshrc` for `csh`) can lead to errors. Check for the occurrence of `stty` in startup files. Avoid the commands that are described in Verifying the Installation. Because batch jobs are usually run off line, the Grid Engine system notifies users of error events in the following ways:

- Logs error messages to the Grid Engine system log file.
- Sends an email to the job owner – If the error log file can't be opened, email is the only way to notify the user of an error event. Therefore, the email system should be properly installed for Grid Engine users.
- Set up Grid Engine system definition files – You can set up the following definition files for Grid Engine users:
  - `qmon` – Resource file for the Grid Engine system GUI. See Choosing a User Interface.
  - `sge_aliases` – Aliases for the path to the current working directory. See Using Path Aliasing.
  - `sge_request` – Default request definition file. See Configuring Default Requests.

Topic	Description
Configuring User Access	Learn how to configure manager accounts, operator accounts, user access lists, and user objects.
Configuring Projects	Learn how to provide a means to organize joint computational tasks from multiple users. A project also defines resource usage policies for all jobs that belong to such a project.
Configuring Default Requests	Learn what default requests are and how they are formatted.
Using Path Aliasing	Learn about path aliasing, the format of path-aliasing files, and how path-aliasing files are interpreted.

Configuring User Access

There are four categories of users that each have access to their own set of Grid Engine system commands:

- Managers – Managers have full capabilities to manipulate the Grid Engine system. By default, the superusers of all administration hosts have manager privileges.
- Operators – Users who can perform the same commands as managers except that they cannot change the configuration. Operators are supposed to maintain operation.
- Users – People who can submit jobs to the grid and run them if they have a valid login ID on at least one submit host and one execution host. Users have no cluster management or queue management capabilities.
- Owners – Users who can suspend or resume and disable or enable the queues they own. Typically, users are owners of the queue instances that reside on their workstations. Queue owners can be managers, operators, or users. These privileges are necessary for successful use of `qidle`. Users are commonly declared to be owners of the queue instances that reside on their desktop workstations. See How to Configure Owners Parameters for more information.

For information on which command capabilities are available to the different user categories, see Command Line Interface Ancillary Programs.

Task	User Interface	Description
How to Configure Manager Accounts	CLI or QMON	Learn how to display, add, and delete managers.
How to Configure Operator Accounts	CLI or QMON	Learn how to display, add, and delete operators.
How to Configure User Access Lists	CLI or QMON	Learn how to display, add, and delete user access lists.
How to Configure User Objects	CLI or QMON	Learn how to display, add, modify, and delete user objects.

How to Configure Manager Accounts From the Command Line

To configure a manager account from the command line, use the following arguments for the `qconf` command:

- To display a list of all managers, type the following command:

```
qconf -sm
```

The `-sm` option (show managers) displays a list of all Grid Engine system managers.

- To add a manager, type the following command:

```
qconf -am <username>
```

The `-am` option (add manager) adds one or more users to the list of Grid Engine system managers. By default, the root accounts of all trusted hosts are Grid Engine system managers. See About Hosts and Daemons for more information.

- To delete a manager, type the following command:

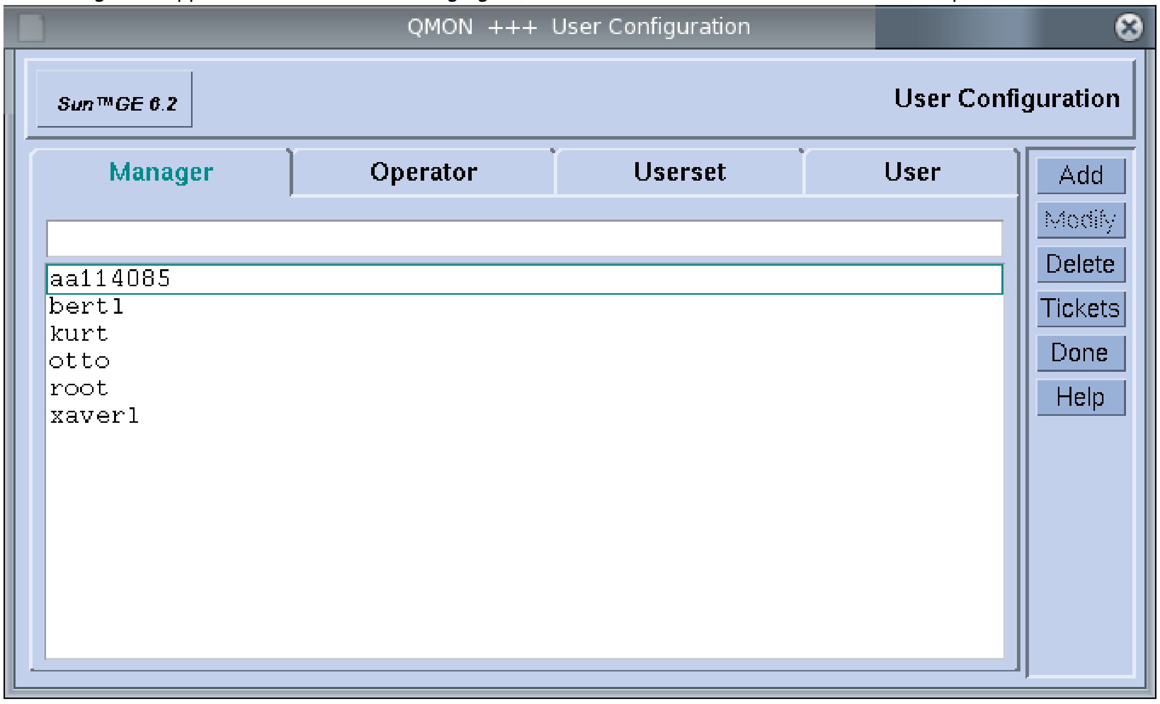
```
qconf -dm <username>
```

The `-dm` option (delete manager) deletes the specified users from the list of Grid Engine system managers.

For more information on how to configure the cluster from file or how to modify many objects at once, see Using qconf.

How to Configure Manager Accounts With QMON

1. On the QMON Main Control window, click the User Configuration button.
   The Manager tab appears, shown in the following figure, and lists all accounts that have administrative permission.



```
aa114085
bertl
kurt
otto
root
xaverl
```

2. You can perform the following functions from the the Manager tab:
   - To add a manager to the Grid Engine system, do the following:
     a. Type the user name in the field above the manager account list.
     b. Click Add or press the Return key.
   - To delete a manager account, select it, and then click Delete.

How to Configure Operator Accounts From the Command Line

To configure a operator account from the command line, use the following arguments for the `qconf` command:

- To display a list of operators, type the following command:

  ```
  qconf -so
  ```

  The `-so` option (show operators) displays a list of all Grid Engine system operators.

- To add an operator, type the following command:

  ```
  qconf -ao <username>
  ```

  The `-ao` option (add operator) adds one or more users to the list of Grid Engine system operators.

- To delete an operator, type the following command:

  ```
  qconf -do <username>
  ```

  The `-do` option (delete operator) deletes the specified users from the list of Grid Engine system operators.

For more information on how to configure the cluster from file or how to modify many objects at once, see Using qconf.

How to Configure Operator Accounts With QMON

1. On the QMON Main Control window, click the User Configuration button, and then click the Operator tab.
   The Operator tab lists all accounts that currently have restricted administrative permission. If the account also has manager access, then that overrides operator access. See Configuring Manager Accounts With QMON.

2. You can perform the following functions from the Operator tab:
   - To add a new operator account, type its name in the field above the operator account list.
     Click Add or press the Return key.
   - To delete an operator account, select it, and then click Delete.

How to Configure User Access Lists From the Command Line

The administrator can define access permissions to queues and other facilities, such as parallel environment interfaces. Access can be restricted to certain users or user groups.

Departments are a special form of access list that assign functional shares and override tickets. A user may be assigned to only one department access list.

For the purpose of restricting access permissions, the administrator creates and maintains access lists (ACLs). The ACLs contain user names and UNIX group names. The ACLs are then added to access-allowed or access-denied lists in the queue or in the parallel environment interface configurations. Users who belong to ACLs that are listed in access-allowed-lists have permission to access the queue or the parallel environment interface. Users who are members of ACLs in access-denied-lists cannot access the resource in question.

ACLS can also be used to define projects, to which assigned users can submit their jobs. The administrator can also restrict access to cluster resources on a per project basis. For more information on projects, see Configuring Projects.

For more information, see the queue_conf(5) or sge_pe(5) man pages.

Client Commands

To configure user access lists from the command line, use the following arguments for the qconf command:

- To display a user access list, type the following command:

  ```
  qconf -su <accesslistname> [,...]
  ```

  The -su option (show user access list) displays the specified access lists.

- To display all user access lists, type the following command:

  ```
  qconf -sul
  ```

  The -sul option (show user access lists) displays all access lists currently defined.

- To add a one or more users to an access list, type the following command:

  ```
  qconf -au <username> [,...] <accesslistname> [,...]
  ```

  The -au option (add user) adds one or more users to the specified access lists. The Grid Engine software provides the following default accesslistnames:

  - arusers – The arusers access list restricts the use of qrsub to request an advance reservation. Only users listed in this access list may use this option. For more information, see How to Enable a User to Create Advance Reservations.
  - deadlineusers – The deadlineusers access list restricts the use of the qsub -dl date_time switch, which specifies the deadline initiation time. The deadline initiation time is the time at which a job has to reach top priority in order to be completed within a given deadline. Only usernames listed in the deadlineusers access list are allowed to request this option. For more information, see Configuring the Urgency Policy.
- To add a user access list from a file, type the following command:

  ```
  qconf -Au <filename>
  ```

  The -Au option (add user access list from file) uses a configuration file, filename, to add an access list.

- To modify a user access list, type the following command:

  ```
  qconf -mu <accesslistname>
  ```

  The -mu option (modify user access list) modifies the specified access lists. For more information, see User Access List Configuration Parameters.

- To modify a user list from file, type the following command:

  ```
  qconf -Mu <filename>
  ```

  The -Mu option (modify user access list from file) uses a configuration file, filename, to modify the specified access lists. For more information, see User Access List Configuration Parameters.

- To delete one more more users from an access list, type the following command:

  ```
  qconf -du <username> [,...] <accesslistname> [,...]
  ```

The -du option (delete user) deletes one or more users from the specified access lists.

- To delete a user list, type the following command:

  ```
  qconf -dul <access-list-name> [,...]
  ```

  The -dul option (delete user list) completely removes userset lists.

For more information on how to configure the cluster from file or how to modify many objects at once, see Using qconf.

User Access List Configuration Parameters

For more information, see the `access_list(5)` man page.

Access lists are used in Grid Engine to define access permissions of users to queues or parallel environments.

Parameter	Description
`name`	Name of the access list.
`type`	The type of access list. Can be defined as an ACL, DEPT, or both.
`oticket`	Amount of override tickets currently assigned to the department.
`fshare`	Current functional share of the department.
`entries`	A list of UNIX users or primary UNIX groups that are assigned to the access list or department.

How to Configure User Access Lists With QMON

The administrator can define access permissions to queues and other facilities, such as parallel environment interfaces. Access can be restricted to certain users or user groups.

Departments are a special form of access list that assign functional shares and override tickets. A user may be assigned to only one department access list.

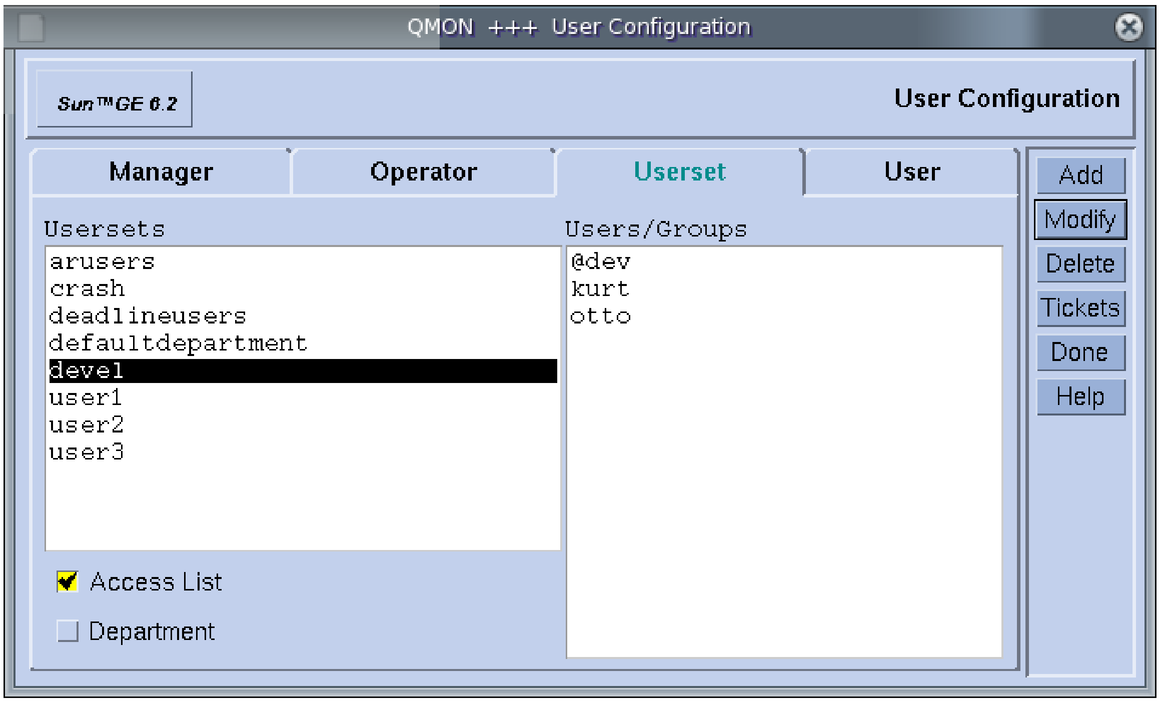
For the purpose of restricting access permissions, the administrator creates and maintains access lists (ACLs). The ACLs contain user names and UNIX group names. The ACLs are then added to access-allowed or access-denied lists in the queue or in the parallel environment interface configurations. Users who belong to ACLs that are listed in access-allowed-lists have permission to access the queue or the parallel environment interface. Users who are members of ACLs in access-denied-lists cannot access the resource in question.

ACLS can also be used to define projects, to which assigned users can submit their jobs. The administrator can also restrict access to cluster resources on a per project basis. For more information on projects, see Configuring Projects.

For more information, see the `queue_conf(5)` or `sge_pe(5)` man pages.

Steps

1. On the QMON Main Control window, click the User Configuration button, and then click the Userset tab.
   The Userset tab appears as shown in the following figure.


QMON +++ User Configuration

Sun™GE 6.2

User Configuration

Manager | Operator | Userset | User

Usersets

```
arusers
crash
deadlineusers
defaultdepartment
devel
user1
user2
user3
```

[✓] Access List

[ ] Department

Users/Groups

```
@dev
kurt
otto
```

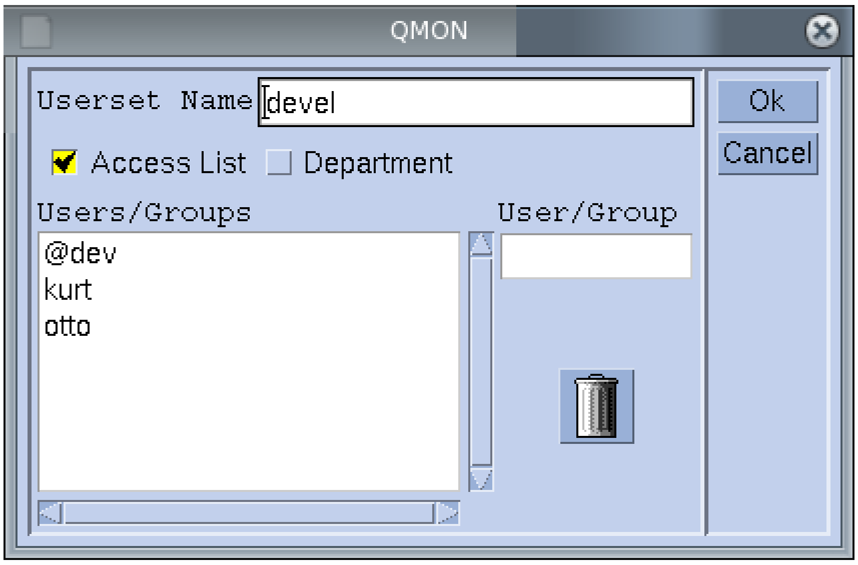
In the Grid Engine system, a userset can be either an access list, a Department, or both. The check boxes below the Usersets list indicate the type of the selected userset. This section describes access lists. Departments are explained in How to Configure User Objects.
The Usersets list displays all available access lists. The Grid Engine software provides the following default access lists:

- `arusers` – The `arusers` access list restricts the use of `qrsub` to request an advance reservation. Only users listed in this access list may use this option. For more information, see How to Enable a User to Create Advance Reservations.
- `deadlineusers` – The `deadlineusers` access list restricts the use of the `qsub -dl` date_time switch, which specifies the deadline initiation time. The deadline initiation time is the time at which a job has to reach top priority in order to be completed within a given deadline. Only usernames listed in the `deadlineusers` access list are allowed to request this option. For more information, see Configuring the Urgency Policy.
- `defaultdepartment` – The `defaultdepartment` access list automatically includes all users who are not assigned to an administrator-created department. For more information, see Configuring the Functional Policy.

To display currently defined users and groups, select the userset.

Note
The names of groups are prefixed with an @ sign.

1. You can perform the following tasks from the Userset Tab:
   - To add a new userset, click Add.
     An Access List Definition dialog box appears, as shown in the figure below. The Users/Groups list displays all currently defined users and groups.



- To add a new access list definition, type the name of the access list in the Userset Name field and click Ok.
- To add a new user or group to the access list, type a user or group name in the User/Group field and then click Ok. Be sure to prefix group names with an @ sign.
- To delete a user or group from the Users/Groups list, select it and then click the trash icon.

- To modify an existing userset, select it, and then click Modify.
  An Access List Definition dialog box appears with the name of the current userset in the Userset Name field.

- To delete a userset, select it, and then click Delete.

2. To save your changes and close the dialog box, click OK.
   Click Cancel to close the dialog box without saving changes.

How to Configure User Objects From the Command Line

To configure a user object from the command line, use the following arguments for the `qconf` command:

- To display a the configuration for a specific user, type the following command:

  ```
  qconf -suser <username>
  ```

  The `-suser` option (show user) displays the configuration of the specified user.

- To display a list of all currently defined users, type the following command:

  ```
  qconf -suserl
  ```

  The `-suserl` option (show user list) displays a list of all currently defined users.

- To add a user, type the following command:

  ```
  qconf -auser
  ```

  The `-auser` option (add user) opens a template user configuration in an editor. See the `user(5)` man page. The editor is either the default `vi` editor or the editor specified by the `EDITOR` environment variable. After you save your changes and exit the editor, the changes are registered with `sge_qmaster`.

- To add a user from file, type the following command:

```
qconf -Auser <filename>
```

The `-Auser` option (add user from file) parses the specified file and adds the user configuration. The file must have the format of the user configuration template.

- To modify a user, type the following command:

```
qconf -muser <username>
```

The `-muser` option (modify user) enables you to modify an existing user entry. The option loads the user configuration in an editor. From the editor, you can modify a user's configuration, including the user's default project. After you save your changes and exit the editor, the changes are registered with `sge_qmaster`. For more information, see User Configuration Parameters.

- To modify a user from file, type the following command:

```
qconf -Muser <filename>
```

The `-Muser` option (modify user from file) parses the specified file and modifies the user configuration. The file must have the format of the user configuration template. For more information, see User Configuration Parameters.

- To delete a user, type the following command:

```
qconf -duser <username> [,...]
```

The `-duser` option (delete user) deletes one or more user objects.

For more information on how to configure the cluster from file or how to modify many objects at once, see Using qconf.

User Configuration Parameters

For detailed information about these parameters, see the `user(5)` man page.

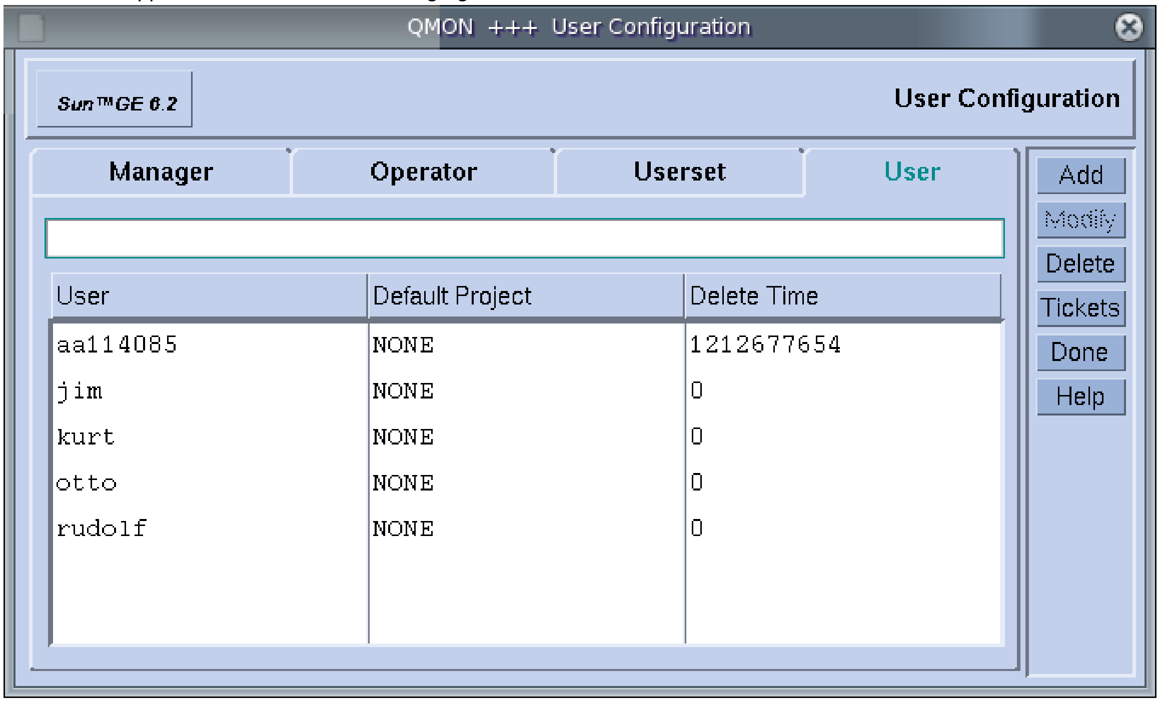
A user entry is used to store ticket and usage information on a per user basis. Maintain user entries for all users participating in a Grid Engine system is required if Grid Engine is operated under a user share tree policy. For more information, see Configuring the Share-Based Policy.

Parameter	Description
`name`	User name
`oticket`	Amount of override tickets currently assigned to the user.
`fshare`	Functional share of the user.
`default_project`	Default project of the user.
`delete_time`	Deprecated.

How to Configure User Objects With QMON

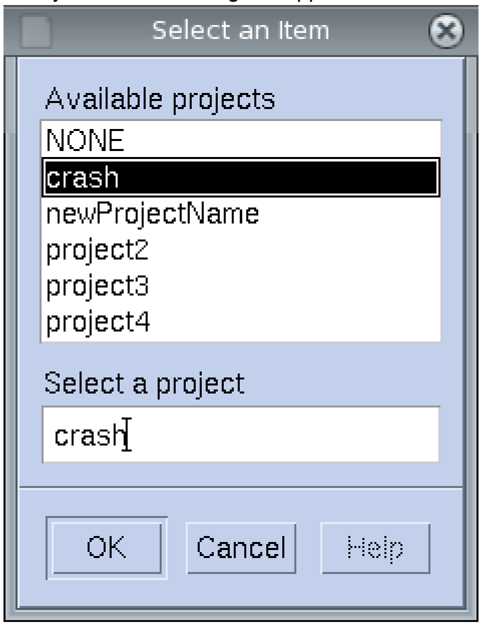
1. On the QMON Main Control window, click the User Configuration button.

2. Click the User tab.
   The User tab appears as shown in the following figure:



User	Default Project	Delete Time
aa114085	NONE	1212677654
jim	NONE	0
kurt	NONE	0
otto	NONE	0
rudolf	NONE	0

3. You can view all currently configured users and perform the following functions from the User Tab:
   - To add a new user, type a user name in the field above the User list, and then click Add or press the Return key.
   - To delete a user, select the user name in the User list, and then click Delete.
     The Delete Time column is read-only. The column indicates the time at which automatically created users are to be deleted from the Grid Engine system. Zero indicates that the user will never be deleted.
   - To assign a default project, do the following:
     a. Select a user, and then click the Default Project column heading.
        A Project Selection dialog box appears, as shown below.



        You can assign a default project to each user. The default project is attached to each job that users submit, unless those users request another project to which they have access.
        Departments are used for the configuration of the functional policy and the override policy. Departments differ from access lists in that a user can be a member of only one department, whereas one user can be included in multiple access lists. For more details, see Configuring the Functional Policy and Configuring the Override Policy.
        A Userset is identified as a department by the Department flag. A Userset can be defined as both a department and an access list at the same time. However, the restriction of only a single appearance by any user in any department applies.
     b. Select a project for the highlighted user entry.
     c. Click OK to assign the default project and close the dialog box.

Click Cancel to close the dialog box without assigning the default project.

Configuring Projects

Projects provide a means to organize joint computational tasks from multiple users. A project also defines resource usage policies for all jobs that belong to such a project.

Projects must be declared before they can be used in any of the three scheduling policy policies. Projects are used in three policy areas:

- Share-based, when shares are assigned to projects – see Configuring the Share-Based Policy
- Functional, when projects receive a percentage of the functional tickets – see Configuring the Functional Policy
- Override, when an administrator grants override tickets to a project – see Configuring the Override Policy

Grid Engine system managers define projects by giving them a name and some attributes. Grid Engine users can attach a job to a project when they submit the job. Attachment of a job to a project influences the job's dispatching, depending on the project's share of share-based, functional, or override tickets.

Task	User Interface	Description
How to Define Projects	CLI or QMON	Learn how to display, add, modify, and delete projects.

How to Configure Projects From the Command Line

> Note
> If you would like to give all users access to a project, leave the default ACL setting at NONE. If the ACL is set to `defaultdepartment`, users cannot submit jobs to the project.

To define projects from the command line, use the following arguments for the `qconf` command:

- To display the configuration for a specific project, type the following command:

  ```
  qconf -sprj <projectname>
  ```

  The `-sprj` option (show project) displays the configuration of a particular project.

- To display a list of all currently-configured projects, type the following command:

  ```
  qconf -sprjl
  ```

  The `-sprjl` option (show project list) displays a list of all currently defined projects.

- To add a project, type the following command:

  ```
  qconf -aprj
  ```

  The `-aprj` option (add project) opens a template project configuration in an editor. See the `project(5)` man page. The editor is either the default `vi` editor or the editor specified by the `EDITOR` environment variable. After you save your changes and exit the editor, the changes are registered with `sge_qmaster`.

- To add a project from file, type the following command:

```
qconf -Aprj <filename>
```

The -Aprj option (add project from file) parses the specified file and adds the new project configuration. The file must have the format of the project configuration template.

- To modify a project, type the following command:

```
qconf -mprj <projectname>
```

The -mprj option (modify project) enables you to modify an existing user entry. The option loads the project configuration in an editor. The editor is either the default vi editor or the editor specified by the EDITOR environment variable. After you save your changes and exit the editor, the changes are registered with sge_qmaster. For more information, see Project Configuration Parameters.

- To modify a project from file, type the following command:

```
qconf -Mprj <filename>
```

The -Mprj option (modify project from file) parses the specified file and modifies the existing project configuration. The file must have the format of the project configuration template. For more information, see Project Configuration Parameters.

- To delete a project from the command line, type the following command:

```
qconf -dprj <projectname>
```

The -dprj option (delete project) deletes one or more projects.

For more information on how to configure projects from file or how to modify many objects at once, see Using qconf.

Project Configuration Parameters

For detailed information about these parameters, see the project(5) man page.

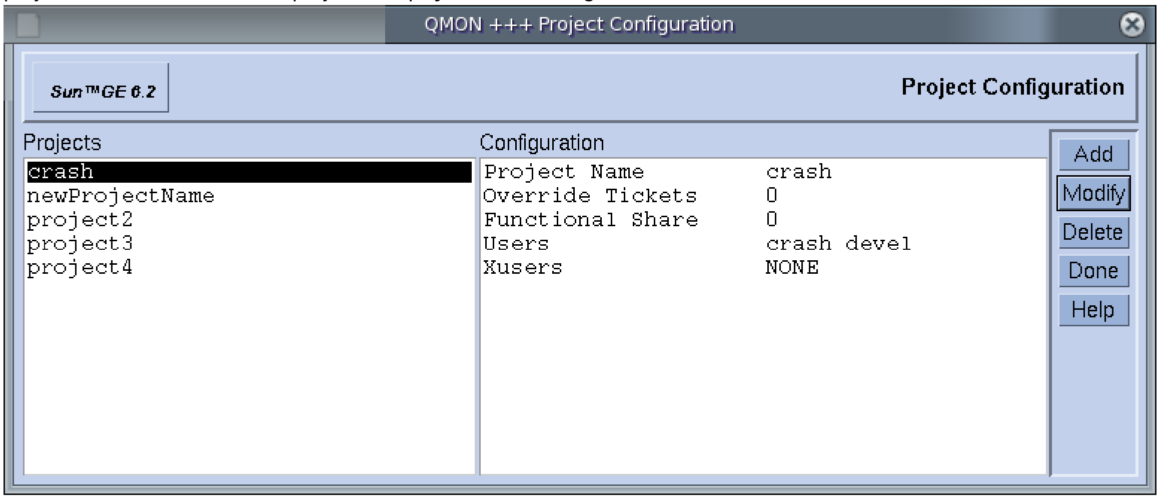
Jobs can be submitted to projects, and a project can be assigned with a certain level of importance via the functional or the override policy. Their level of importance is then inherited by the jobs executing under that project.

Parameter	Description
name	The name of the project.
oticket	The amount of override tickets currently assigned to the project.
fshare	The current functional share of the project.
acl	A list of user access lists referring to users who are allowed to submit jobs to the project.
xacl	A list of user access lists referring to users who are not allowed to submit jobs to the project.

How to Configure Projects With QMON

Grid Engine system managers can define and update definitions of projects by using the Project Configuration dialog box.

1. On the QMON Main Control window, click the Project Configuration button.
   The Project Configuration dialog box appears, as shown below. The currently defined projects are displayed in the Projects list. The project definition of a selected project is displayed under Configuration.



Projects

```
crash
newProjectName
project2
project3
project4
```

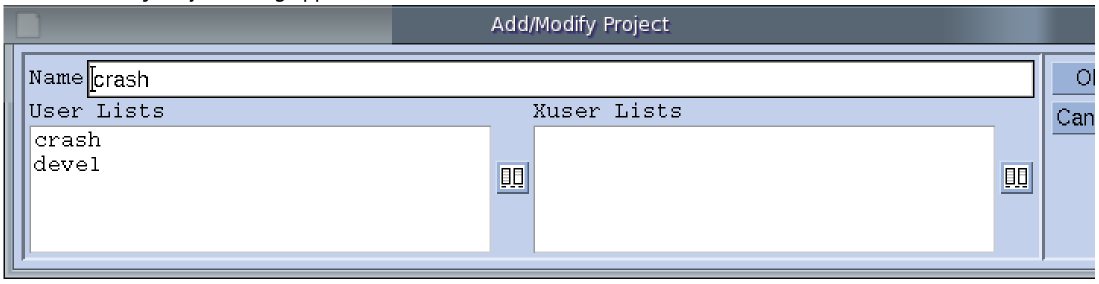
Configuration

```
Project Name         crash
Override Tickets     0
Functional Share     0
Users                crash devel
Xusers               NONE
```

2. You can perform the following tasks from the Project Configuration dialog box:
   - To add a new project, do the following:
     a. Click Add.
        The Add/Modify Project dialog box appears.
     b. Click Ok to save changes.
        Click Cancel to leave the dialog box without saving your changes.
   - To modify a project, do the following:

   > Note
   > If you would like to give all users access to a project, leave the default ACL setting at NONE. If the ACL is set to `defaultdepartment`, users cannot submit jobs to the project.

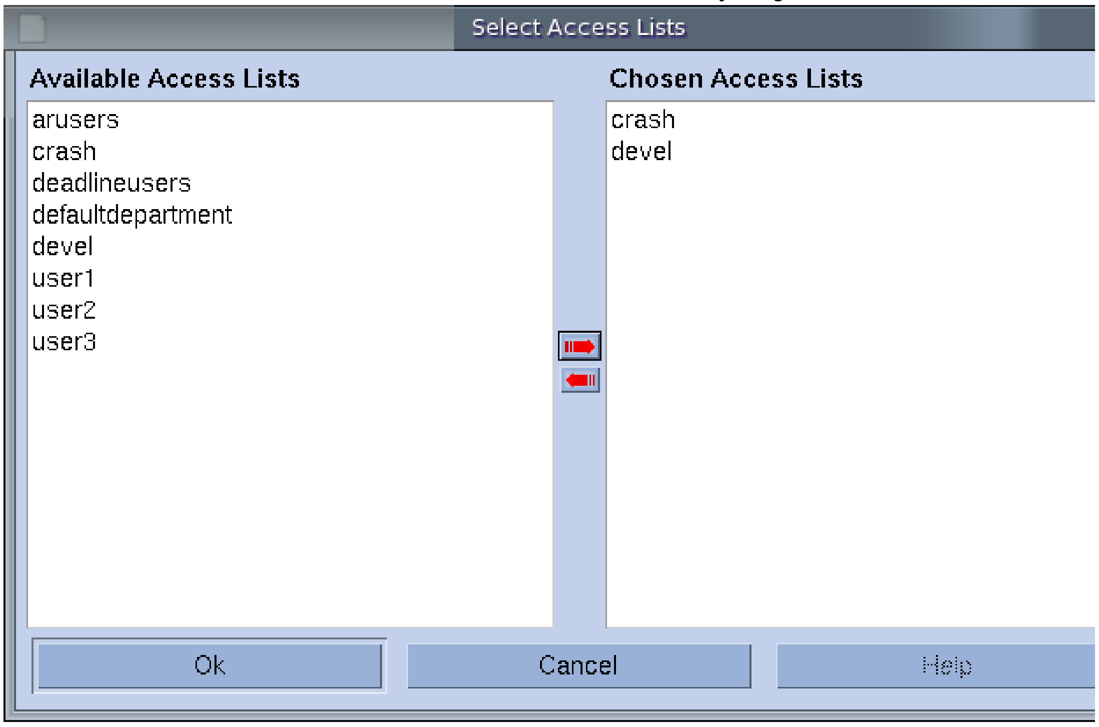
     a. Select a project, and then click Modify.
        The Add/Modify Project dialog appears, as shown below.



```
Name crash
User Lists                    Xuser Lists
crash
devel
```

        The Add/Modify Project dialog box shows the following access lists:
        - The User Lists column shows access lists that contain users who have permission to access the project.
        - The Xuser Lists column shows access lists that contain users who do not have permission to access the project.
        - If both lists are empty, all users can access the project. If a user belongs to an access list that is in the User Lists and to an access list that is in the Xuser Lists, the user is denied access to the project.
          The name of the selected project is displayed in the Name field. The project defines the access lists of users who are permitted access or who are denied access to the project. See Configuring Users for more information.
     b. To add or remove users from the User Lists or Xuser Lists, click the button at the right of the User Lists or the Xuser Lists.
        The Select Access Lists dialog box, shown in the figure below, appears. Currently defined access lists are displayed

under Available Access Lists. Currently selected access lists are displayed under chosen access lists. You can select access lists in either list. You can move access lists from one list to the other by using the red arrows.



Available Access Lists: arusers, crash, deadlineusers, defaultdepartment, devel, user1, user2, user3

Chosen Access Lists: crash, devel

c. Click Ok to save your changes.
   Click Cancel to close the dialog box without saving any changes.

- To delete a project, select it, and then click Delete.

Configuring Default Requests

Typically, a user defines a request profile for a particular job. If the user does not specify any requests for a job, the scheduler considers any queue to which the user has access without further restrictions. Batch jobs are normally assigned to queues with respect to this request profile. The scheduler considers only those queues that satisfy the set of requests for this job.

As an administrator, the Grid Engine software enables you to configure default requests that define resource requirements for jobs even when the user does not specify resource requirements explicitly.

You can configure default requests globally for all users of a cluster, as well as privately for any user. The default request configuration is stored in default request files. The global request file is located under `$SGE_ROOT/$SGE_CELL/common/sge_request`. The user-specific request file can be located either in the user's home directory or in the current working directory. The working directory is where the `qsub` command is run. The user-specific request file is called `.sge_request`.

If these files are present, they are evaluated for every job. The order of evaluation is as follows:

1. The global default request file
2. The user default request file in the user's home directory
3. The user default request file in the current working directory

Note
The requests specified in the job script or supplied with the `qsub` command take precedence over the requests in the default request files. See Submitting Jobs for details about how to request resources for jobs explicitly.

You can prevent the Grid Engine system from using the default request files by using the `qsub -clear` command, which discards any previous requirement specifications.

Format of Default Request Files

The format of both the local and the global default request files is as follows:

- Default request files can contain any number of lines. Blank lines and lines that begin with a # sign are skipped.
- Each line not to be skipped can contain any `qsub` option, as described in the qsub(1) man page. More than one option per line is allowed. The batch script file and the argument options to the batch script are not considered to be `qsub` options. Therefore these items are not allowed in a default request file.
- The `qsub -clear` command discards any previous requirement specifications in the currently evaluated request file or in request files processed earlier.

Suppose a user's local default request file is configured the same as `test.sh`, the script in the following example.

Example of a Default Request File

```
# Local Default Request File
# exec job on a sun4 queue offering 5h cpu
-l arch=solaris64,s_cpu=5:0:0
# exec job in current working dir
-cwd
```

To run the script, the user types the following command:

```
% qsub test.sh
```

The effect of running the `test.sh` script is the same as if the user specified all `qsub` options directly in the command line, as follows:

```
% qsub -l arch=solaris64,s_cpu=5:0:0 -cwd test.sh
```

Note
Like batch jobs submitted using `qsub`, interactive jobs submitted using `qsh` consider default request files also. Interactive or batch jobs submitted using QMON also take these request files into account.

Using Path Aliasing

In Solaris and in other networked UNIX environments, users often have the same home directory, or part of it, on different machines. For example, consider user home directories that are available across NFS and automounter. A user might have a home directory `/home/foo` on the NFS server. This home directory is accessible under this path on all properly installed NFS clients that are running automounter. However, `/home/foo` on a client is just a symbolic link to `/tmp_mnt/home/foo`. `/tmp_mnt/home/foo` is the actual location on the NFS server from where automounter physically mounts the directory.

A user on a client host might use the `qsub -cwd` command to submit a job from somewhere within the home directory tree. The `-cwd` flag requires the job to be run in the current working directory. However, if the execution host is the NFS server, the Grid Engine system might not be able to locate the current working directory on that host. The reason is that the current working directory on the submit host is `/tmp_mnt/home/foo`, which is the physical location on the submit host. This path is passed to the execution host. However, if the execution host is the NFS server, the path cannot be resolved, because its physical home directory path is `/home/foo`, not `/tmp_mnt/home/foo`.

Other occasions that can cause similar problems are the following:

- Fixed NFS mounts with different mount point paths on different machines. An example is the mounting of home directories under `/usr/people` on one host and under `/usr/users` on another host.
- Symbolic links from outside into a network-available file system.

To prevent such problems, the Grid Engine software enables both the administrator and the user to configure a path aliasing file. The locations of two such files are as follows:

- `$SGE_ROOT/$SGE_CELL/common/sge_aliases` – A global cluster path-aliasing file for the cluster
- `$HOME/.sge_aliases` – A user-specific path-aliasing file

Note
Only an administrator should modify the global file.

Format of Path-Aliasing Files

Both path-aliasing files share the same format:

- Blank lines and lines that begin with a # sign are skipped.
- Each line, other than a blank line or a line preceded by #, must contain four strings separated by any number of blanks or tabs. The strings are as follows:
  - The first string specifies a source path.
  - The second string specifies a submit host.
  - The third string specifies an execution host.
  - The fourth string specifies the source path replacement.
- Both the submit host and the execution host strings can be an * (asterisk), which matches any host.

How Path-Aliasing Files Are Interpreted

The files are interpreted in the following order:

1. After **qsub** retrieves the physical current working directory path, the global path-aliasing file is read, if present.
   The user path-aliasing file is read afterwards, as if the user path-aliasing file were appended to the global file.

2. Lines not to be skipped are read from the top of the file, one by one. The translations specified by those lines are stored, if necessary. A translation is stored only if both of the following conditions are true:
   - The submit host string matches the host on which the `qsub` command is run.
   - The source path forms the initial part either of the current working directory or of the source path replacements already stored.
3. After both files are read, the stored path-aliasing information is passed to the execution host along with the submitted job.

4. On the execution host, the path-aliasing information is evaluated. The source path replacement replaces the leading part of the current working directory if the execution host string matches the execution host. In this case, the current working directory string is changed. To be applied, subsequent path aliases must match the replaced working directory path.
   The following is an example of how the NFS automounter problem described earlier can be resolved with an aliases file entry.

Example – Path Aliasing File

```
# cluster global path aliases file
# src-path  subm-host   exec-host   dest-path
/tmp_mnt/   *           *           /
```

Configuring Queue Calendars

Queue calendars define the availability of queues according to the day of the year, the day of the week, or the time of day. You can configure queues to change their status at specified times. You can change the queue status to disabled, enabled, suspended, or resumed (unsuspended).

The Grid Engine system enables you to define a site-specific set of calendars, each of which specifies status changes and the times at which the changes occur. These calendars can be associated with queues. Each queue can attach a single calendar, thereby adopting the availability profile defined in the attached calendar.

The syntax of the calendar format is described in detail in the `calendar_conf(5)` man page. A few examples are given in the next sections, along with a description of the corresponding administration facilities.

For information about configuring queues, see Configuring Queues.

Task	User Interface	Description
How to Configure Queue Calendars	CLI or QMON	Learn how to display, add, modify, and delete queue calendars.

How to Configure Queue Calendars From the Command Line

To configure execution hosts from the command line, use the following arguments for the `qconf` command:

- To display the configuration for a specific calendar, type the following command:

  ```
  qconf -scal <calendarname>
  ```

- To display a list of all configured calendars, type the following command:

  ```
  qconf -scall
  ```

- To add a new calendar configuration, type the following command:

  ```
  qconf -acal <calendarname>
  ```

  The `-acal` option (add calendar) adds a new calendar configuration named calendarname to the cluster. An editor with a template configuration appears, enabling you to define the calendar.

- To add a calendar from file, type the following command:

  ```
  qconf -Acal <filename>
  ```

  The `-Acal` option (add calendar from file) adds a new calendar configuration to the cluster. The added calendar is read from the specified file.

- To modify a calendar, type the following command:

  ```
  qconf -mcal <calendarname>
  ```

  The `-mcal` option (modify a calendar configuration) modifies an existing calendar configuration named calendarname. An editor opens calendarname, enabling you to make changes to the definition.

- To modify a calendar from file, type the following command:

  ```
  qconf -Mcal <filename>
  ```

  The `-Mcal` option (modify calendar from file) modifies an existing calendar configuration. The calendar to modify is read from the specified file.

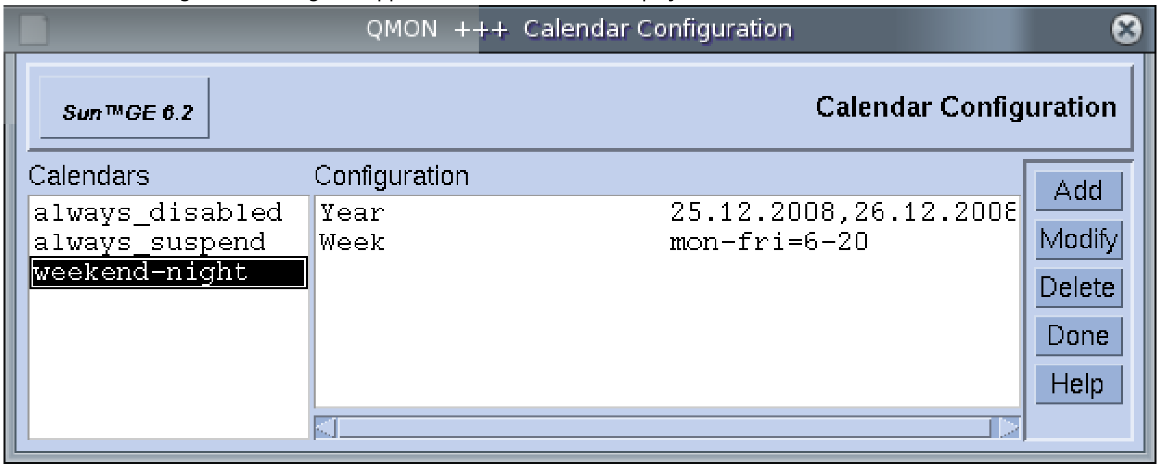
- To delete a calendar, type the following command:

  ```
  qconf -dcal <calendarname> [,...]
  ```

  The `-dcal` option (delete calendar) deletes the specified calendar.

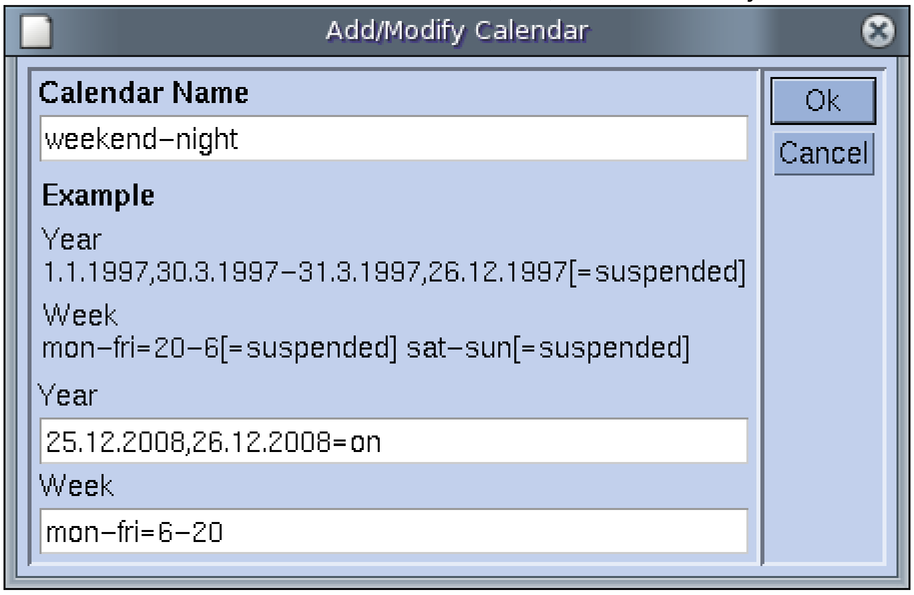
For more information on how to configure the queue calendars from file or how to modify many objects at once, see Using qconf.

How to Configure Queue Calendars With QMON

1. In the QMON Main Control window, click the Calendar Configuration button.
   The Calendar Configuration dialog box appears. The Calendars list displays the available calendars.
   

   always_disabled
   always_suspend
   weekend-night

   ```
   Year        25.12.2008,26.12.2008
   Week        mon-fri=6-20
   ```

2. In the Calendars list, click the calendar configuration that you want to modify or delete.
   - To delete the selected calendar, click Delete.
   - To modify the selected calendar, click Modify.
   - To add a calendar, click Add.
     In all cases, the Add/Modify Calendar dialog box appears. The following example shows the definition for a queue that is available outside office hours and on weekends. In addition, the Christmas holidays are defined to be handled like weekends.
     

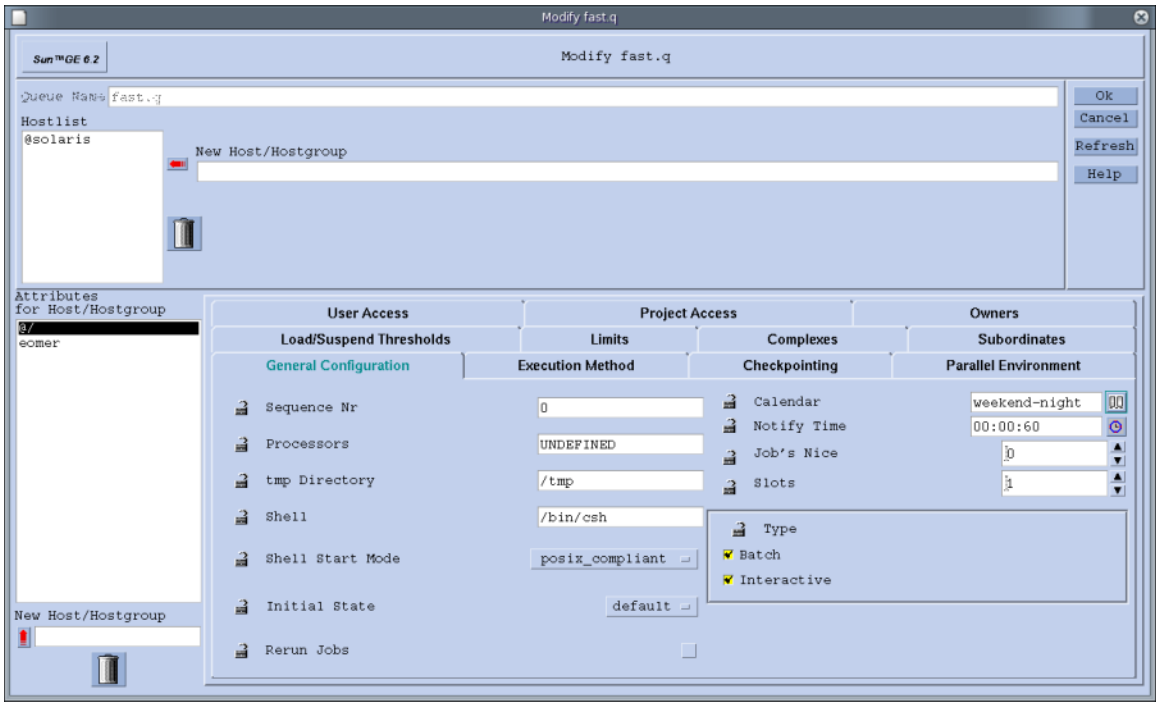
     Calendar Name: weekend-night

     Example
     Year
     1.1.1997,30.3.1997-31.3.1997,26.12.1997[=suspended]
     Week
     mon-fri=20-6[=suspended] sat-sun[=suspended]

     Year: 25.12.2008,26.12.2008=on
     Week: mon-fri=6-20

     If you click Modify or Delete, the Calendar Name field displays the name of the selected calendar. If you click Add, you need to add a name in the Calendar Name field.
     The Year and Week fields enable you to define the calendar events, using the syntax described in the `calendar_conf`(5) man page.
     See the `calendar_conf`(5) man page for a detailed description of the syntax and for more examples.

3. To control access to a specific queue based on calendar data, attach the calendar to the queue as follows:
   a. From the main QMON window, click Queue Control.
   b. Select the queue name to which you want to attach the calendar and click the Modify button. The Modify queuename window appears with the General Configuration tab selected.
   c. On the General Configuration tab, use the drop-down list next to the Calendar field to choose the calendar to attach.
   Attaching a calendar configuration to a queue sets the availability profile defined by the calendar for the queue. See About Configuring Queues for more details about configuring queues.


Modify fast.q

Sun™GE 6.2 — Modify fast.q

Queue Name: fast.q

Hostlist: @solaris

New Host/Hostgroup

Attributes for Host/Hostgroup: @/, eomer

New Host/Hostgroup

Sequence Nr	0	Calendar	weekend-night
Processors	UNDEFINED	Notify Time	00:00:60
tmp Directory	/tmp	Job's Nice	0
Shell	/bin/csh	Slots	1
Shell Start Mode	posix_compliant	Type	✓ Batch, ✓ Interactive
Initial State	default		
Rerun Jobs	☐		

Using Job Submission Verifiers

Job Submission Verifiers (JSVs) allow users and administrators to define rules that determine which jobs are allowed to enter into a cluster and which jobs should be rejected immediately. A JSV is a script or binary that can be used to verify, modify, or reject a job during the time of job submission or on the master host.

The following are examples of how an administrator might use JSVs:

- To verify that a user has write access to certain file systems.
- To make sure that jobs do not contain certain resource requests, such as memory resource requests (`h_vmem` or `h_data`).
- To add resource requests to a job that the submitting user may not know are necessary in the cluster.
- To attach a user's job to a specific project or queue to ensure that cluster usage is accounted for correctly.
- To inform a user about certain job details like queue allocation, account name, parallel environment, total number of tasks per node, and other job requests.

A verification can be performed by a client JSV instance at the time of job submission, by a server JSV instance on the master host, or by a combination client JSVs and server JSVs. In general, client JSVs should meet your cluster's needs. See below for more information on what client JSVs and server JSVs have to offer.

Job Submission Verifier Topics

Topic	Description
Understanding the Differences Between Client JSVs and Server JSVs	Before you get started, it is important that you learn the differences between client JSVs and server JSVs.
Writing JSV Scripts	Learn how different programming languages can impact the performance of your cluster, about JSV script-based functions, and how to write JSV scripts.
JSV Verification Process	Learn about how the Sun Grid Engine system executes JSVs.
Configuring JSVs	Learn how to configure JSVs.
JSV Communication Protocol	Learn about the communication protocol used by JSV instances to communicate with a client process and/or the master daemon.

Understanding the Differences Between Client JSVs and Server JSVs

To maximize this feature's usefulness, it is important to understand the differences between client JSVs and server JSVs.

	Client JSV	Server JSV
Configured by	Any user (see below under Configuration Location(s) for information on the exception)	An administrator or by a user who has been given administrative access in the global configuration
Executed by	Client application	Master daemon
Lifetime of JSV Process	After the verification is completed regardless of the result, the JSV instance is stopped.	The JSV remains active and unchanged as long as the master daemon remains running.
Configuration Location(s)	You can configure client JSVs in the following four locations: • By adding the `-jsv` submit parameter and a `jsv_url` to a submit client (`qsub`, `qrsh`, qsh, and `qlogin` only) at the time of job submission. For more information `jsv_url` syntax, see the `sge_types(1)` man page. For more information on `-jsv`, see the `qsub(1)` man page. • By adding the `-jsv` switch to the `$cwd/.sge_request` file. For more information on `sge_request` file syntax, see the `sge_request(5)` man page. • By adding the `-jsv` switch to the `$HOME/.sge_request` file. • By adding the `-jsv` switch to the `$SGE_ROOT/$SGE_CELL/common/sge_request` file.	You can only configure server JSVs in the following location: • By including a `jsv_url` in the global configuration. For more information on the cluster configuration parameters, see the `sge_conf(5)` man page. For more information `jsv_url` syntax, see the `sge_types(1)` man page.
Additional Information	Note the following about client JSVs: • The execution context for all client JSV configuration options is the same. • Client JSVs do not consume cluster resources because they are executed during job submission. In clusters where hundreds of jobs are submitted per second,server JSVs can reduce the submit rate by 30% or more. • Client JSVs log activity to `stdout`. • Client JSVs are an excellent way to test server JSV scripts. • A client JSV defined in the global `sge_request` file can only be defined by an administrator. • A client JSV defined in the global `sge_request` file is executed last of all client JSVs. This means this JSV script can touch a job after all user-defined JSV scripts have been executed.	Note the following about server JSVs: • Server JSVs can not be bypassed by end users. • Server JSV scripts can perform administrative functions. • Server JSVs can log activity to the global message file. • Server JSVs can perform tasks specific to one host. • Server JSVs are always executed last, after all client JSVs have been executed. • If you use a server JSV, then all `qalter` and `qmon` modification requests are rejected by the master daemon. As an administrator, you can configure the `jsv_allowed_mod` parameter in the global configuration to allow a set of switches to be used with submit clients.

Configuring JSVs

You can configure JSVs in up to five different locations. Each configured JSV will be executed by the Grid Engine system when a job is submitted, which means that a job can be verified up to five different times.

Generally, a client JSV, which consumes less cluster resources, will meet your computing needs. For more information on the differences between client JSVs and server JSVs, see Understanding the Differences Between Client JSVs and Server JSVs.

JSV Configuration Tasks

Task	Interface	Description
How to Configure a Client JSV	CLI	Learn how to configure a client JSV.
How to Configure a Server JSV	CLI	Learn how to configure a server JSV.

How to Configure a Client JSV From the Command Line

Generally, client JSVs will meet your cluster's needs. For more information on the differences between client JSVs and server JSVs, see Understanding the Differences Between Client JSVs and Server JSVs.

1. Write a JSV script that meets your cluster's job verification needs.

2. Configure a client JSV in one of the following locations:

   You may configure a client JSV in all of the following locations. The only distinction is that the last option (modifying the global `sge_request` file) is always executed last.

- By adding the -jsv submit parameter to a submit client (qsub, qrsh, qsh, and qlogin only) specifying a jsv_url at the time of job submission.

```
qsub -jsv <jsv_url>
```

- By adding the -jsv switch to the .cwd/.sge_request file.
  For more information on sge_request file syntax, see the `sge_request(5)` man page.
- By adding the -jsv switch to the $HOME/.sge_request file.
- By adding the -jsv switch to the $SGE_ROOT/$SGE_CELL/common/sge_request file.
  Since this option modifies the global `sge_request` file, it must be configured by an administrator. This option is always executed after the first three options are executed by the Sun Grid Engine System.

If a server JSV is configured, it will be executed after all configured client JSVs have been executed. For more information, see the JSV Verification Process page.

For more information on `-jsv`, see the `qsub(1)` man page. For more information jsv_url syntax, see the `sge_types(1)` man page.

How to Configure a Server JSV From the Command Line

Unless you need to write a script that requires administrative privileges, it generally makes sense to use client JSVs for your verification needs because they will consume less of your cluster's resources. For more information, see Understanding the Differences Between Client JSVs and Server JSVs.

1. Write a JSV script that meets your cluster's job verification needs.

2. Include a **`jsv_url`** in the global configuration.
   For more information on the cluster configuration parameters, see the `sge_conf(5)` man page. For more information `jsv_url` syntax, see the `sge_types(1)` man page.

A configured server JSV will always be executed after all configured client JSVs are executed.

Writing JSV Scripts

JSV scripts can be written in any scripting language, including Unix shells, Perl or Tcl.

The following tools are available to assist you in writing a JSV script:

- The JSV Script Functions are available in Bourne shell, Tcl or Perl scripts after sourcing the files `jsv_include.sh`, `jsv_include.tcl`, or `JSV.pm`.
- The files and corresponding JSV script templates are located in the directory `$SGE_ROOT/util/resource/jsv/`.

To maximize the performance of your script, the following suggestions should be considered:

- Use a scripting language that supports precompilation or other performance improvement methods during runtime.
- Use a scripting language that avoids forking processes. Startup of additional applications is expensive and will slow down JSV scripts heavily.
- Avoid accessing files and other input/output devices in your scripts.

Performance Considerations for Server JSV Scripts

Since server JSV instances are executed by the master daemon, it is important to choose a scripting language that minimizes the amount of resources that are consumed from the master daemon process. Simple scripting languages, like the Bourne shell, execute a lot of external commands to perform very basic operations. If used in a JSV script, the job acceptance rate could decrease by more than 90%. By contrast, a more efficient scripting language, like Perl and Tcl, should impact the job acceptance rate by less than 10% with a well-designed JSV script.

Performance Considerations for Client JSV Scripts

A client side JSV script is started whenever a user submits a job. If a series of job submissions are done from the same terminal or script, it is important to use an efficient scripting language, such as Perl or Tcl, to minimize the script's impact on client submit rate.

Topic	Description
JSV Script Functions	A list of JSV script functions and parameters.
Example - Writing a JSV Script Using the Bourne Shell	A hypothetical example of how a JSV script can be written using UNIX.
Example - Writing a JSV Script Using Tcl	A hypotehtical example of how a JSV script can be written using Tcl.

Submit Parameters

The JSV submit parameters communicated in the JSV Communication Protocol via the PARAM command are initially specified at the command line during the time of job submission or by a corresponding functionality in QMON. The parameters are almost identical to the switches used by `qsub`. The values for these parameters are described below:

Parameter	Description
a	This value of the a parameter defines or redefines the time and date at which a job is eligible for execution. If the `-a` command line switch or the corresponding value in QMON was specified during the submission of the job, then the parameter named a will be sent to configured JSV instances. The value for this parameter has the format [CCYY]MMDDhhmm.SS. This parameter can be changed by using JSV scripts.

ac	The value for the ac parameter represents the job context. The context can be specified with the qsub command line switches -ac, -dc and -sc or corresponding functionality in QMON. The outcome of the evaluation of all -ac, -dc, and -sc options or corresponding values in QMON are passed to defined JSV instances as parameters with the name ac. The value for this attribute is a comma separated list of variable/value pairs (ac_list).
ar	The value of the ar parameter contains the advance reservation ID. Jobs can be assigned to Advance Reservations using the -ar command line switch or corresponding functionality in QMON.
A	The value of the A parameter identifies the account to which the resource consumption for the job should be charged. If this option or a corresponding in QMON is specified, then this value will be passed to defined JSV instances as a parameter with the name A.
b	The value of the b parameter indicates explicitly whether command should be treated as binary or script. The value specified with this option or the corresponding value specified by QMON will only be passed to defined JSV instances if the value is yes or y.
binding_strategy binding_type binding_amount binding_step binding_socket binding_core binding_exp_n binding_exp_socket<id> binding_exp_core<id>	The -binding parameter specifies all core binding specific settings that should be applied to a job during execution. The values passed to this command line switch are passed as multiple parameters to JSV instances. binding_strategy represents the strategy to be used and has one of following values: linear, striding or explicit. binding_type represents the instance that should do the binding and is one of following strings: env, set or pe. binding_socket and binding_core are socket/core values whereas binding_step is the step size (used only for striding binding). Please note that the length of the socket/core value list of the explicit binding is reported as binding_exp_n. The <id> part of binding_exp_socket<id> and binding_exp_core<id> will be replaced by the position of the socket/core pair within the explicit binding list (0 <= id < binding_exp_n). The first socket/core pair of the explicit binding will be reported with the parameter names binding_exp_socket0 and binding_exp_core0. Values that do not apply for the specified binding will not be reported to JSV. E.g. binding_step will only be reported for the striding binding and all binding_exp_... values will only be passed to JSV if explicit binding was specified.
c_interval c_occasion	The value of the qsub -c parameter defines or redefines whether the job should be checkpointed, and, if so, what the circumstances are. The value specified with this option or the corresponding value specified in QMON will be passed to defined JSV instances. The interval will be available as a parameter with the name c_interval. The character sequence specified will be available as a parameter with the name c_occasion. Please note that changing c_interval will overwrite previous settings of c_occasion and vice versa.
ckpt	The value of the ckpt parameter selects the checkpointing environment. If this option or a corresponding value in QMON is specified, then this value will be passed to all defined JSV instances as a parameter with the name ckpt.
cwd	The value of the cwd parameter is the absolute path to the current working directory. JSV scripts can remove the path from jobs during the verification process by setting the value of this parameter to an empty string. As a result, the job behaves as if -cwd was not specified during job submission. If this option or a corresponding value in QMON is specified (and a script does not remove the path), then this value will be passed to all defined JSV instances as parameters with the name cwd.
dc	The dc parameter removes the given variables from the job's context. The outcome of the evaluation of all -ac, -dc, and -sc options or corresponding values in QMON are passed to defined JSV instances as parameters with the name ac.
display	The display parameter directs xterm to use display_specifier to contact the X server. If this option or a corresponding value in QMON is specified, then this value will be passed to defined JSV instances as a parameter with the name display. This value will also be available in the job environment, which can be passed to JSV scripts.
dl	The dl parameter specifies the deadline initiation time. The format for the date_time value is [CCYY] MMDDhhmm.SS. If this option or a corresponding value in QMON is specified, then this value will be passed to defined JSV instances as a parameter with the name dl.
e	The e parameter defines or redefines the path used for the standard error stream of the job. If this option or a corresponding value in QMON is specified, then this value will be passed to defined JSV instances as a parameter with the name e.

h	The h parameter places holds on a job. If this option is specified with qsub or during job submission in QMON, then the parameter h with the value u will be passed to the defined JSV instances indicating that the job will be in user hold after the submission finishes.
hold_jid	The hold_jid parameter defines or redefines the job dependency list of the submitted job. If this option or a corresponding value in QMON is specified, then this value will be passed to all defined JSV instances as a parameter with the name hold_jid.
hold_jid_ad	The hold_jid_ad parameter defines or redefines the job array dependency list of the submitted job. If this option or a corresponding value in QMON is specified, then this value will be passed to all defined JSV instances as a parameter with the name hold_jid_ad.
i	The i parameter defines or redefines the file used for the standard input stream of the job. If this option or a corresponding value in QMON is specified, then this value will be passed to all defined JSV instances as a parameter with the name i.
j	The j parameter specifies whether or not the standard error stream of the job is merged into the standard output stream. The value specified with this option or the corresponding value specified in QMON will only be passed to defined JSV instances if the value is yes or y. The name of the parameter will be j.
js	The js parameter defines or redefines the job share of the job relative to other jobs. If this option or a corresponding value in QMON is specified, then this value will be passed to all defined JSV instances as parameters with the name js.
l_hard l_soft	The qsub -l parameter launches the job for a Grid Engine queue meeting the given resource request list. If this option or a corresponding value in QMON is specified, then these hard and soft resource requirements will be passed to defined JSV instances as parameters with the names l_hard and l_soft. If regular expressions will be used for resource requests, then these expressions will be passed as they are. Also, shortcut names will not be expanded.
m	The m parameter defines or redefines under which circumstances mail is to be sent to the job owner or to the users defined with the -M option described below. If this option or a corresponding value in QMON is specified, then this value will be passed to defined JSV instances as a parameter with the name m.
M	The M parameter defines or redefines the list of users to which the server that executes the job has to send mail. If this option or a corresponding value in QMON is specified, then this value will be passed to all defined JSV instances as a parameter with the name M.
masterq	The masterq parameter defines or redefines a list of cluster queues, queue domains, and queue instances that may be used to become the master queue of this parallel job. If this option or a corresponding value QMON is specified, then this hard resource requirement will be passed to defined JSV instances as a parameter with the name masterq.
notify	The notify parameter sends warning signals to a running job prior to sending signals. This option provides the running job a configured time interval to do cleanup operations. This option is used, the parameter named notify with the value y will be passed to defined JSV instances.
now	The now parameter tries to start the job immediately or not at all. The value specified with this option or the corresponding value specified in QMON will only be passed to defined JSV instances if the value is yes. The name of the parameter will be now. The value will also be y when the long for yes was specified during submission.
N	The N parameter provides the name of the job. The value specified with this option or the corresponding value specified in QMON will be passed to defined JSV instances as the parameter with the name N.
o	The o parameter provides the path used for the standard output stream for the job. If this option or a corresponding value in QMON is specified, then this value will be passed to all defined JSV instances as a parameter with the name o.
p	The p parameter defines or redefines the priority of the job relative to other jobs. If this option or a corresponding value in QMON is specified and the priority is not 0, then this value will be passed to all defined JSV instances as a parameter with the name p.
pe_name pe_min pe_max	The qsub -pe parameter specifies a parallel environment and a slot or slot range n-m. If this option or a corresponding value in QMON is specified, then the parameters pe_name, pe_min, and pe_max will be passed to all configured JSV instances. The values pe_min and pe_max represent the values n and m, which have been provided with the -pe option. A missing specification of m will have the value 9999999 in JSV scripts, which represents infinity.

P	The P parameter specifies the project to which the job is assigned. If this option or a corresponding value in QMON is specified, then this value will be passed to defined JSV instances with the `ot` parameter.
`q_hard` `q_soft`	The qsub `-q` parameter defines or redefines a list of cluster queues, queue domains or queue instances that may be used to execute this job. If this option or a corresponding value in QMON is specified, then these hard and soft resource requirements will be passed to defined JSV instances as parameters with the names `q_hard` and `q_soft`. If regular expressions are used for resource requests, then these expressions will be passed as they are. Also shortcut names will not be expanded.
R	The R parameter indicates whether a reservation for this job should be done. The value specified with this option or the corresponding value specified in QMON will only be passed to defined JSV instances if the value is `yes`. The name of the parameter is R. The value is `y` also when the long form `yes` was specified during submission.
`r`	The `r` parameter identifies the ability of a job to be rerun or not. The value specified with this option or the corresponding value specified in QMON will only be passed to defined JSV instances if the value is `yes`. The name of the parameter will be `r`. The value will be w also when the long form `yes` was specified during submission.
`sc`	The `sc` parameter sets the given name/value pairs as the job's context. The outcome of the evaluation of all `-ac`, `-dc`, and `-sc` options or corresponding values in QMON is passed to all defined JSV instances as parameters with the name `ac`.
`shell`	The `shell` parameter determines whether or not to use a command shell. The value specified with this option or the corresponding value specified in QMON will only be passed to defined JSV instances if the value is `yes`. The name of the parameter will be `shell`. The value will be `y` also when the long form `yes` was specified during submission.
`soft`	The `soft` parameter signifies that all resource requirements following in the command line will be soft requirements. If this option or a corresponding value in QMON is specified, then the corresponding `-q` and `-l` resource requirements will be passed to all defined JSV instances as parameters with the names `q_soft` and `l_soft`. For more information, see descriptions for `-q` and `-l`.
S	The S parameter specifies the interpreting shell for the job. If this option or a corresponding value in QMON is specified, then this value will be passed to all defined JSV instances as a parameter with the name S.
`t`	The `t` parameter submits an array job. If this option or a corresponding value in QMON is specified, then this value will be passed to all defined JSV instances as parameters with the name `t_min`, `t_max`, and `t_step`.
`tc`	Allow users to limit concurrent array job task execution. The value specified with this option or the corresponding value specified in QMON will be passed to defined JSV instances as the parameter with the name tc.
`v`	The v parameter defines or redefines the environment variables to be exported to the execution context of the job. All environment variables specified with `-v`, `-V`, or the `DISPLAY` variable provided with `-display` will be exported to all defined JSV instances only optionally when this is requested explicitly during the job submission verification.
`V`	The V parameter specifies a validation level applied to the job to be submitted and/or the specified queued job. All environment variables specified with `-v`, `-V`, or the `DISPLAY` variable provided with `-display` will be exported to the defined JSV instances only optionally when this is requested explicitly during the job submission verification.
`wd`	The wd parameter executes the job from the directory specified in the working_directory. The parameter value will be available in defined JSV instances as a parameter with the name cwd.

For detailed information on JSV submit parameters, see the `qsub(1)` man page.

Pseudo Parameters

Parameter	Description

CLIENT	The corresponding value for the CLIENT parameters is either qmaster or the command name of a submit client like qsub, qsh , qrsh, or qlogin and so on. This parameter value can't be changed by JSV instances. ClIENT is always be sent as part of a job verification.
CMDARGS	CMDARGS displays a count of the number of arguments that are passed to a job script or command. CMDARGS is always sent during a job verification. There is also a CMDARGid for each argument, starting with CMDARG0. If no arguments are being passed to the job script or command, CMDARGS will have the value 0. To change an argument, you must change its CMDARGid value. To add an argument, add a new CMDARGid. For example, if your job is myjob.sh test 4, then CMDNAME is myjob.sh, CDMARG0 is test, and CDMARG1 is 4. To change myjob.sh test 4 to myjob.sh test 5, you must set CDMARG1 to 5. To change myjob.sh test 4 to myjob.sh test 5 true, you must set CDMARG2 to true. Note Currently, there is an unresolved issue that prevents users from removing arguments.
CMDNAME	Either the path to the script or the command name in the case of binary submission. It will always be sent as part of a job verification.
CONTEXT	It can be client if the JSV that receives this param_command command was started by a command line client like qsub, qsh, qrsh, or qlogin and so on. It is master if it was started by the master daemon process. It will always be sent as part of a job verification. Changing the value of this parameter is not possible within JSV instances.
GROUP	This is the primary group of the user who is submitting the job to be verified. Cannot be changed but it is always sent as part of the verification process. The groupname is passed as parameter with the name GROUP.
JOB_ID	Not available in the client context (see CONTEXT). Otherwise it contains the job number of the job which will be submitted to Grid Engine when the verification process is successful. JOB_ID is an optional parameter which can't be changed by JSV instances.
USER	This is the username of the user who is submitting the job to be verified. Cannot be changed but it is always sent as part of the verification process. The username is passed as parameters with the name USER.
VERSION	VERSION will always be sent as part of a job verification process and it will always be the first parameter that is sent. It will contain a version number of the format <major>.<minor>. In version 6.2u2 and higher, the value will be 1.0. The value of this parameters can't be changed.

JSV Verification Process

After you have configured the necessary JSV(s) and you submit a job, the verification process begins. All configured JSVs are executed in the order described below. Each configured client JSV instance communicates directly with the client process and each configured server JSV instance communicates directly with the master daemon process. For more information on how JSVs communicate with the client process or the master daemon, see the JSV Communication Protocol page. Using an example, the verification process is described in the below.

Order that the System Executes JSVs

Note
Only administrators can configure client JSVs in the global request file or server JSVs in the global configuration. These JSVs are always executed last.

Sun Grid Engine executes JSV instances in the following order:

Order of Execution	JSV Type	Configuration Location	JSV Trigger	Submitted by

1	client JSV	command line	`-jsv jsv_url` used with a submit client (`qsub`, `qrsh`, `qsh`, and `qmon`)	any user
2	client JSV	`$cwd/.sge_request` file	`-jsv jsv_url`	any user
3	client JSV	`$HOME/.sge_request` file	`-jsv jsv_url`	any user
4	client JSV	`$SGE_ROOT/$SGE_CELL/common/sge_request` file	`-jsv jsv_url`	administrator or user with administrative privileges
5	server JSV	global configuration	`jsv_url` parameter	administrator or user with administrative privileges

Duration of of the JSV Verification Process

The verification process stops when one of the following occurs:

- All configured JSVs deliver an `accept` result.
- All configured JSVS deliver either an `accept` or `accept with correction` result.
- One JSV delivers a `reject` or a `reject later` result.

Example – JSV Verification Process

For the purposes of this example, the following JSVs are configured:

Order of Execution	JSV Type	Configuration Location	JSV Trigger	Submitted By	Description
1	client JSV	`$HOME/.sge_request` file	`-jsv /home/Mike/jsvA.sh`	any user	The script `/home/Mike/jsvA.s` adds user Mike to each job specification (using `-l attr=$USER`) before it is accepted into the Grid Engine system.
2	client JSV	`$SGE_ROOT/$SGE_CELL/common/sge_request` file	`-jsv /sge_root/jsvB.sh`	administrator or user with administrative privileges	The script `/sge_root/jsvB.sh` rejects binary jobs (submitted with `-b y`). All other jobs are accepted.
3	server JSV	global configuration	`jsv_url /sge_root/jsvC.sh`	administrator or user with administrative privileges	The `sge_root/jsvC.sh` script accepts all jobs that don't contain a h_vmem resource request.

Scenario A

1. A job is submitted using `qsub script.sh`.
2. The first JSV instance, that contains the `/home/Mike/jsvA.sh` script, changes the job submission from `qsub script.sh` to `qsub -l attr=Mike script.sh`. The job is then passed to the second JSV instance.
3. The second JSV instance, that contains the `/sge_root/jsvB.sh` script, accepts the job because it is not a binary job. The job is then passed to the third JSV instance.
4. The third JSV instance, that contains the `/sge_root/jsvC.sh` script, also accepts the job because it has no h_vmem in its specification.

5. The job is created.

Scenario B

1. A job is submitted using `qsub -b y`.
2. The first JSV instance, that contains the `/home/Mike/jsvA.sh` script, changes the job specification from `qsub -b y` to `qsub -l attr=Mike -b y`. The job is then passed to the second JSV instance.
3. The second JSV instance, that contains the `sge_root/jsvB.sh` script, rejects the job because it is a binary job. The verification process stops.
4. The user who submitted the job receives an error message.

Scenario C

1. A job is submitted using `qsub -l attr=PETER,hvmem=3G script.sh`.
2. The first JSV instance, that contains the `/home/Mike/jsvA.sh` script, changes the job specification from `qsub -l attr=PETER,hvmem=3G script.sh` to `qsub -l attr=Mike,hvmem=3G script.sh`. The job is then passed to the second JSV instance.
3. The second JSV instance, that contains the `/sge_root/jsvB.sh` script, accepts the job because it is not a binary job. The job is then passed to the second JSV instance.
4. The third JSV instance, that contains the `/sge_root/jsvC.sh` script, rejects the job because it contains `hvmem`.
5. The user who submitted the job receives an error message.

Configuring Resource Attributes

Resource attribute definitions can be associated with a queue, a host, or the entire cluster. A set of default resource attributes is already attached to each queue and host. Default resource attributes are built in to the system and cannot be deleted, nor can their type be changed. These resource attribute definitions are stored in an entity called the Grid Engine system complex and make up the complex configuration.

Users can request all pertinent information about resource attributes that the complex configuration provides for jobs via the `qsub -l` or `qalter -l` commands. The complex configuration also provides information about how the Grid Engine system should interpret these resource attributes.

Configuring the Complex

Task	User Interface	Description
How to Configure the Complex	CLI or QMON	Learn how to display, add, modify, and delete currently-configured complex resource attributes. Although you can define complex resource attributes from the command line, it is easier to use the QMON Complex Configuration dialog box.

Adding Resource Attributes to the Complex

Topic	Description
Adding Resource Attributes to the Complex	Learn how to add resource attributes to the complex.

Assigning Resource Attributes to Queues, Hosts or to the Global Complex

Topic	Description
Assigning Queue Resource Attributes	Learn about how to assign resources attributes to queues.

Assigning Host Resource Attributes	Learn about how to assign resource attributes to hosts.
Assigning Global Resource Attributes	Learn about how to assign global resource attributes.

Defining Consumable Resources

Topic	Description
Defining Consumable Resources	Learn how to define consumable resources, which provide an efficient way to manage limited resources.

How to Configure the Complex From the Command Line

To configure the complex from the command line, use the following arguments for the `qconf` command:

- To display the current complex configuration, type the following command:

  ```
  qconf -sc
  ```

  The `-sc` option prints the current complex configuration to the standard output stream in the file format defined in the `complex(5)` man page. A sample output is shown in the following example.

  ```
  #name       shortcut  type  relop  requestable   consumable  default  urgency
  #----------------------------------------------------------------------------
  nastran     na        INT   <=     YES           NO          0        0
  pam-crash   pc        INT   <=     YES           YES         1        0
  permas      pm        INT   <=     FORCED        YES         1        0
  #---- # start a comment but comments are not saved across edits -----------
  ```

- To modify the complex configuration, type the following command:

  ```
  qconf -mc
  ```

  The `-mc` option opens an editor that contains either a template file for the complex configuration or an existing complex configuration file that you can modify. The changed complex configuration is the registered with the `sgemaster`. You must have root or manager privileges to implement this command.

- To modify the complex configuration from a file, type the following command:

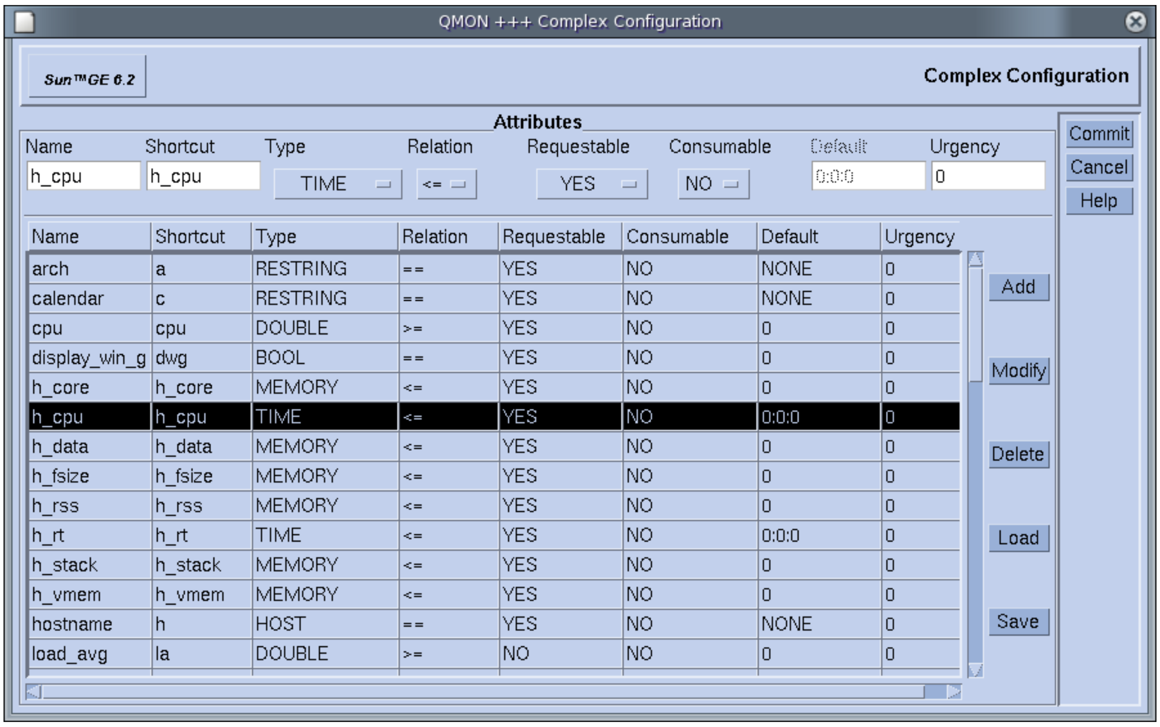
  ```
  qconf -Mc <filename>
  ```

  The `-Mc` option takes a complex configuration file as an argument. The argument file must comply to the format specified in the complex `complex (5)`. You must have root or manager privileges to implement this command.

See the `qconf(1)` man page for a detailed definition of the `qconf` command format and the valid syntax.

How to Configure the Complex With QMON

1. In the QMON Main Control window, click the Complex Configuration button.
   The Complex Configuration dialog box appears as shown in the following figure.



Attributes

Name	Shortcut	Type	Relation	Requestable	Consumable	Default	Urgency
h_cpu	h_cpu	TIME	<=	YES	NO	0:0:0	0

Name	Shortcut	Type	Relation	Requestable	Consumable	Default	Urgency
arch	a	RESTRING	==	YES	NO	NONE	0
calendar	c	RESTRING	==	YES	NO	NONE	0
cpu	cpu	DOUBLE	>=	YES	NO	0	0
display_win_g	dwg	BOOL	==	YES	NO	0	0
h_core	h_core	MEMORY	<=	YES	NO	0	0
h_cpu	h_cpu	TIME	<=	YES	NO	0:0:0	0
h_data	h_data	MEMORY	<=	YES	NO	0	0
h_fsize	h_fsize	MEMORY	<=	YES	NO	0	0
h_rss	h_rss	MEMORY	<=	YES	NO	0	0
h_rt	h_rt	TIME	<=	YES	NO	0:0:0	0
h_stack	h_stack	MEMORY	<=	YES	NO	0	0
h_vmem	h_vmem	MEMORY	<=	YES	NO	0	0
hostname	h	HOST	==	YES	NO	NONE	0
load_avg	la	DOUBLE	>=	NO	NO	0	0

The Complex Configuration dialog box enables you to add, modify, or delete complex resource attributes. See the `complex(5)` man page for details about the meaning of the rows and columns in the table.

2. To add a new attribute, follow these steps:
   a. Make sure that no line in the Attributes table is selected.
      To deselect a highlighted attribute, hold down the Control key and click mouse button 1.
   b. In the fields above the Attributes table, type or select the values that you want.
   c. Click the Add button.

   > Tip
   > You can add a new attribute by copying an existing attribute and then modifying it. Make sure that the attribute name and its shortcut are unique.

3. To modify an attribute, follow these steps:
   a. Select the attribute in the table.
      The values of the selected attribute are displayed above the Attributes table.
   b. Change the attribute values.
   c. Click the Modify button.

4. To save configuration changes to a file, click Save.

5. To load values from a file into the complex configuration, click Load, and select the name of a file from the list.

6. To delete an attribute, select the attribute in the Attributes table, and click Delete.

7. To register your new or modified complex configuration with **`sge_qmaster`**, click Commit.

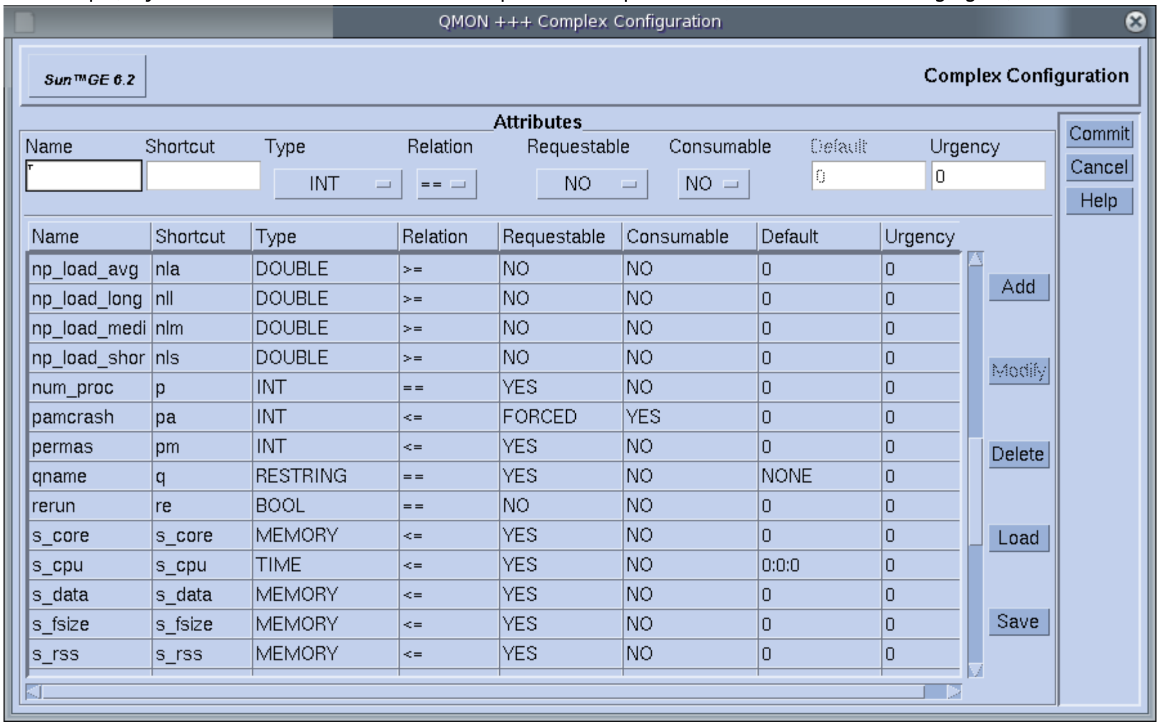
Adding Resource Attributes to the Complex

By adding resource attributes to the complex, the administrator can extend the set of attributes managed by the Grid Engine system. The administrator can also restrict the influence of user-defined attributes to particular queues, hosts, or both.

User-defined attributes are a named collection of attributes with the corresponding definitions as to how the Grid Engine software is to handle these attributes. You can attach one or more user-defined attributes to a queue, to a host, or globally to all hosts in the cluster. Use the complex_values parameter for the queue configuration and the host configuration. For more information, see Configuring Queues and Configuring Hosts. The attributes defined become available to the queue and to the host, respectively, in addition to the default resource attributes.

The complex_values parameter in the queue configuration and the host configuration must set concrete values for user-defined attributes that are associated with queues and hosts.

For example, say the user-defined resource attributes permas and pamcrash, shown in the following figure, are defined.

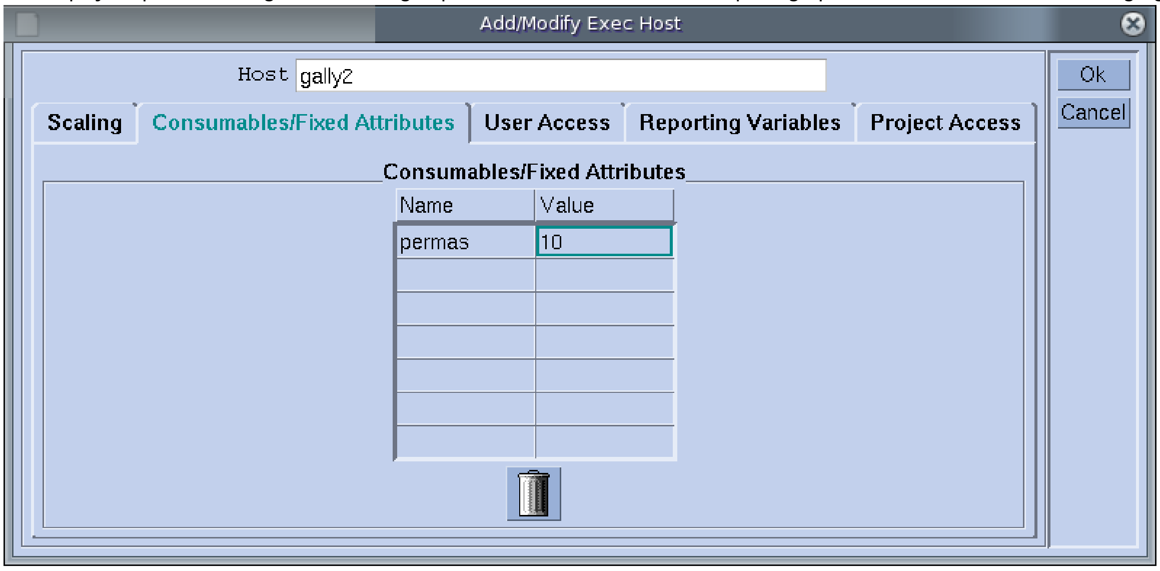


Name	Shortcut	Type	Relation	Requestable	Consumable	Default	Urgency
		INT	==	NO	NO	0	0

Name	Shortcut	Type	Relation	Requestable	Consumable	Default	Urgency
np_load_avg	nla	DOUBLE	>=	NO	NO	0	0
np_load_long	nll	DOUBLE	>=	NO	NO	0	0
np_load_medi	nlm	DOUBLE	>=	NO	NO	0	0
np_load_shor	nls	DOUBLE	>=	NO	NO	0	0
num_proc	p	INT	==	YES	NO	0	0
pamcrash	pa	INT	<=	FORCED	YES	0	0
permas	pm	INT	<=	YES	NO	0	0
qname	q	RESTRING	==	YES	NO	NONE	0
rerun	re	BOOL	==	NO	NO	0	0
s_core	s_core	MEMORY	<=	YES	NO	0	0
s_cpu	s_cpu	TIME	<=	YES	NO	0:0:0	0
s_data	s_data	MEMORY	<=	YES	NO	0	0
s_fsize	s_fsize	MEMORY	<=	YES	NO	0	0
s_rss	s_rss	MEMORY	<=	YES	NO	0	0

For at least one or more queues, add the resource attributes to the list of associated user-defined attributes as shown in the Complex tab of the Modify queue-name dialog box. For details on how to configure queues, see Configuring Queues and its related sections.

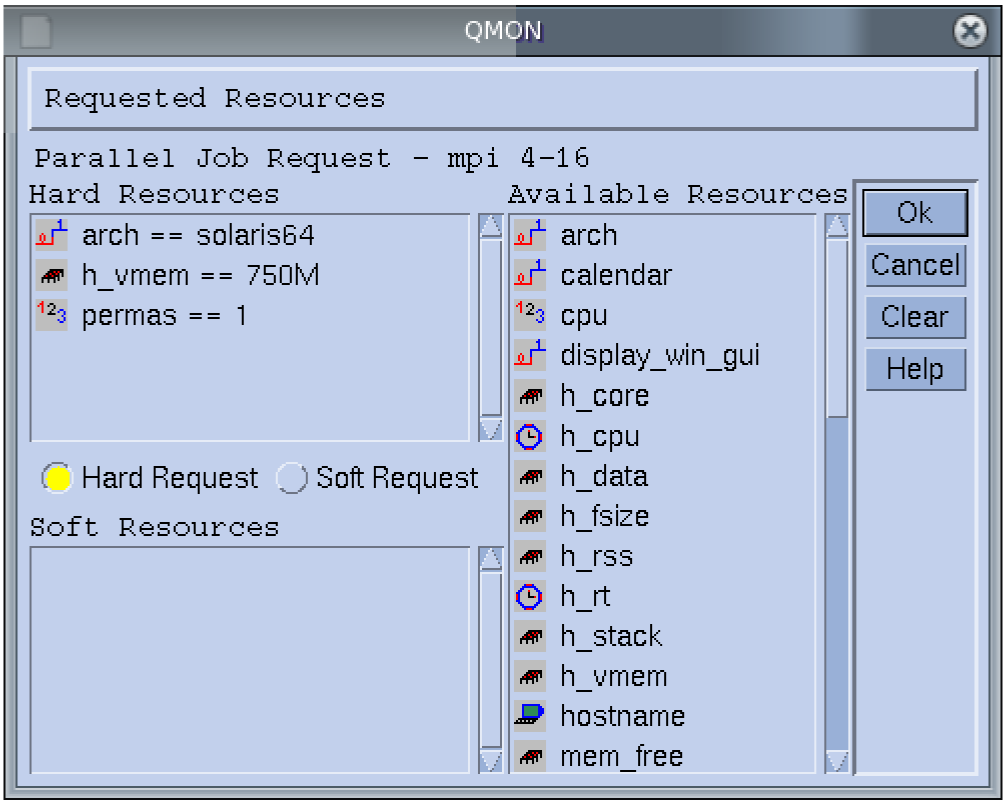
The displayed queue is configured to manage up to 10 licenses of the software package permas as shown in the following figure.



Host gally2

Name	Value
permas	10

The attribute permas becomes requestable for jobs, as expressed in the Available Resources list in the Requested Resources dialog box shown below.


QMON

Requested Resources

Parallel Job Request - mpi 4-16

Hard Resources
- arch == solaris64
- h_vmem == 750M
- permas == 1

Hard Request [✓] Soft Request

Soft Resources

Available Resources
- arch
- calendar
- cpu
- display_win_gui
- h_core
- h_cpu
- h_data
- h_fsize
- h_rss
- h_rt
- h_stack
- h_vmem
- hostname
- mem_free

Consequently, the only eligible queues for these jobs are the queues that are associated with the user-defined resource attributes and that have `permas` licenses configured and available.

For details about how to submit jobs, see Submitting Jobs.

Alternatively, the user could submit jobs from the command line and could request attributes as follows:

```
% qsub -l pm=1 permas.sh
```

Tip
You can use the pm shortcut instead of the full attribute name `permas`.

Assigning Host Resource Attributes

The default host-related attributes are load values. As an administrator, you can add new resource attributes to the default attributes. Every execution daemon periodically reports load to the master daemon. The reported load values are either the standard load values such as the CPU load average, or the load values defined by the administrator, as described in Configuring Load Parameters.

The definitions of the standard load values are part of the default host resource attributes, whereas administrator-defined load values require extending the host resource attributes.

Host-related attributes are commonly extended to include nonstandard load parameters. Host-related attributes are also extended to manage host-related resources such as the number of software licenses that are assigned to a host, or the available disk space on a host's local file system.

If host-related attributes are associated with a host or with a queue instance on that host, a concrete value for a particular host resource attribute is determined by one of the following items:

- The queue configuration, if the attribute is also assigned to the queue configuration

- A reported load value
- The explicit definition of a value in the `complex_values` entry of the corresponding host configuration. For details, see Configuring Hosts.

In some cases, none of these values are available. For example, say the value is supposed to be a load parameter, but `sge_execd` does not report a load value for the parameter. In such cases, the attribute is not defined, and the `qstat -F` command shows that the attribute is not applicable.

For example, the total free virtual memory attribute `h_vmem` is defined in the queue configuration as limit and is also reported as a standard load parameter. The total available amount of virtual memory on a host can be defined in the `complex_values` list of that host. The total available amount of virtual memory attached to a queue instance on that host can be defined in the `complex_values` list of that queue instance. Together with defining `h_vmem` as a consumable resource, you can efficiently exploit memory of a machine without risking memory over-subscription, which often results in reduced system performance that is caused by swapping. For more information about consumable resources, see Defining Consumable Resources.

Note
Only the Shortcut, Relation, Requestable, Consumable, and Default columns can be changed for the default resource attributes. No default attributes can be deleted.

Assigning Global Resource Attributes

Global resource attributes are cluster-wide resource attributes, such as available network bandwidth of a file server or the free disk space on a network-wide available file system.

Global resource attributes can also be associated with load reports if the corresponding load report contains the `GLOBAL` identifier, as described in Configuring Load Parameters. Global load values can be reported from any host in the cluster. No global load values are reported by default, therefore there are no default global resource attributes.

Concrete values for global resource attributes are determined by the following items:

- Global load reports.
- Explicit definition in the `complex_values` parameter of the `global` host configuration. See Configuring Hosts.
- In association with a particular host or queue and an explicit definition in the corresponding `complex_values` lists.

Sometimes none of these cases apply. For example, a load value might not yet be reported. In such cases, the attribute does not exist.

Defining Consumable Resources

Introducing Consumable Resources

Consumable resources, or consumables, provide an efficient way to manage limited resources. The complex also builds the framework for the system's consumable resources facility. The resource attributes that are defined in the complex can be attached to the global cluster, to a host, or to a queue instance. The attached attribute identifies a resource with the associated capability. Attribute definitions in the Grid Engine complex define how resource attributes should be interpreted. During the scheduling process, the availability of resources and the job requirements are taken into account. The Grid Engine system also performs the bookkeeping and the capacity planning that is required to prevent over-subscription of consumable resources.

Typical consumable resources include:

- Available free memory
- Unoccupied software licenses

- Free disk space
- Available bandwidth on a network connection

The definition of a resource attribute includes the following:

- Name of the attribute
- Shortcut to reference the attribute name
- Value type of the attribute, for example, `STRING`, `RESTRING`, `TIME`, or any other complex(5) type
- Relational operator used by the scheduler
- Requestable flag, which determines whether users can request the attribute for a job
- Consumable flag, which identifies the attribute as a consumable resource
- Default request value that is taken into account for consumable attributes if jobs do not explicitly specify a request for the attribute
- Urgency value, which determines job priorities on a per resource basis

Defining Consumable Resources

To enable consumable resource management, you must define the total capacity of a resource. You can define resource capacity globally for the cluster, for specified hosts, and for specified queues. These categories can supersede each other in the given order. Thus a host can restrict availability of a global resource, and a queue can restrict host resources and global resources.

The consumption of the resource is then monitored by Grid Engine software internal bookkeeping. Jobs are dispatched only if the internal bookkeeping indicates that sufficient consumable resources are available.

You define resource capacities by using the `complex_values` attribute in the queue and host configurations. The `complex_values` definition of the `global` host specifies global cluster consumable settings. For more information, see the `host_conf(5)` and `queue_conf(5)` man pages, as well as Configuring Queues and Configuring Hosts.

To each consumable attribute in a `complex_values` list, a value is assigned that denotes the maximum available amount for that resource. The internal bookkeeping subtracts from this total the assumed resource consumption by all running jobs as expressed through the jobs' resource requests.

Multiplied Resource Requests Versus Non-Multiplied Resource Requests

By default Sun Grid Engine performs multiplied resource requests, which means that a consumable resource request is multiplied by the number of slots allocated to a parallel job. The configuration for multiplied resource requests is designated by a YES flag in the `consumable` column of the `job` row in the complex definition.

The following multiplied resource request is explained below:

```
qsub -l mem=100M -pe make=8
```

Sun Grid Engine multiples the consumable resource request (`100 M`) by the number of slots allocated for the parallel job (8). The consumable usage is split across the queues and hosts on which the job runs. If four tasks run on host A and four tasks run on host B, the job consumes 400 Mbytes on each host.

While multiplied resource requests typically work well, in the case of software licenses, it is more practical to make a per job request, or a non-multiplied resource request, which debits the exact amount requested. Starting in Sun Grid Engine 6.2u2, you can configure the complex to accept non-multiplied resource requests by changing the `jobs` consumable flag from YES to JOB, as shown below:

```
#name   shortcut   type   relop   requestable   consumable   default   urgency
#------------------------------------------------------------------------------
jobs       j        INT    <=          YES           JOB        0         0
```

For more on the complex configuration, see the `queue_conf(5)` man page.

Examples – Defining Consumable Resources

Only numeric attributes can be configured as consumables. Numeric attributes are attributes whose type is `INT`, `DOUBLE`, `MEMORY`, or `TIME`.

In the QMON Main Control window, click the Complex Configuration button. The Complex Configuration dialog box appears.

To enable the consumable management for an attribute, set the Consumable flag for the attribute in the complex configuration.

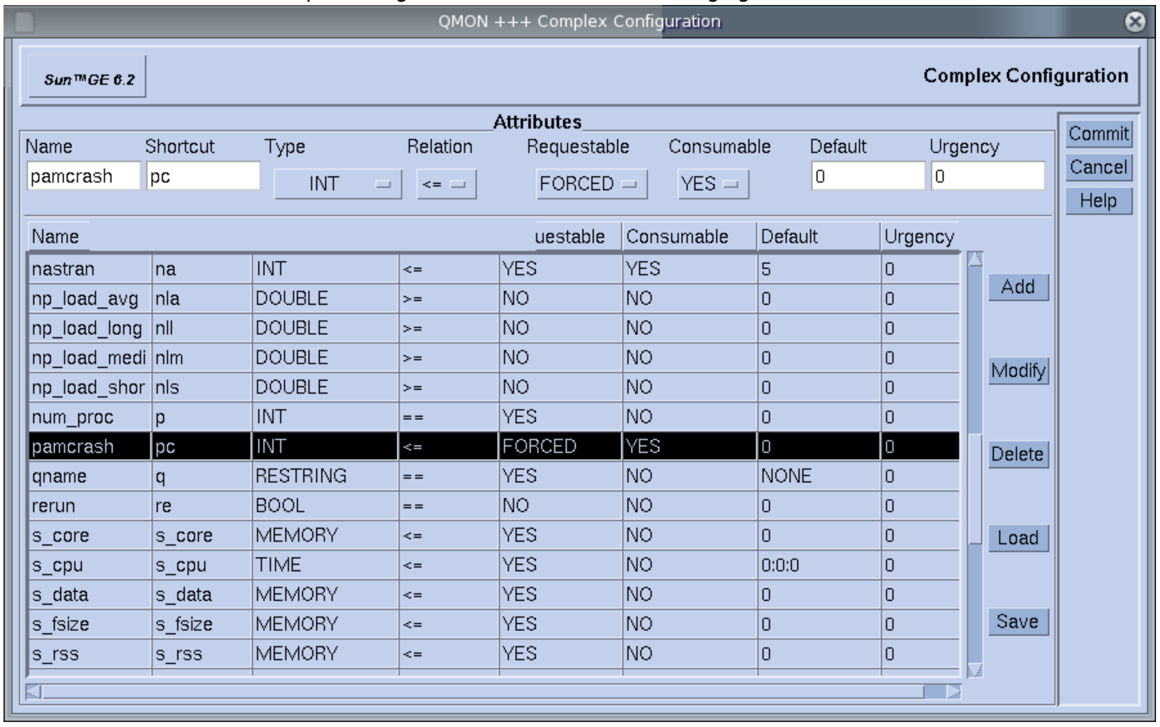
To set up other consumable resources, follow these examples:

Topic
Example 1 - Floating Software License Management
Example 2 - Space Sharing for Virtual Memory
Example 3 - Managing Available Disk Space

Example 1 - Floating Software License Management

Suppose you are using the software package `pam-crash` in your cluster, and you have access to 10 floating licenses. You can use `pam-crash` on every system as long as no more than 10 invocations of the software are active. The goal is to configure the Grid Engine system to prevent scheduling `pam-crash` jobs while all 10 licenses are occupied by other running `pam-crash` jobs.

With consumable resources, you can achieve this goal easily. First you must add the number of available `pam-crash` licenses as a global consumable resource to the complex configuration, as shown in the following figure.



Attributes

Name	Shortcut	Type	Relation	Requestable	Consumable	Default	Urgency
pamcrash	pc	INT	<=	FORCED	YES	0	0

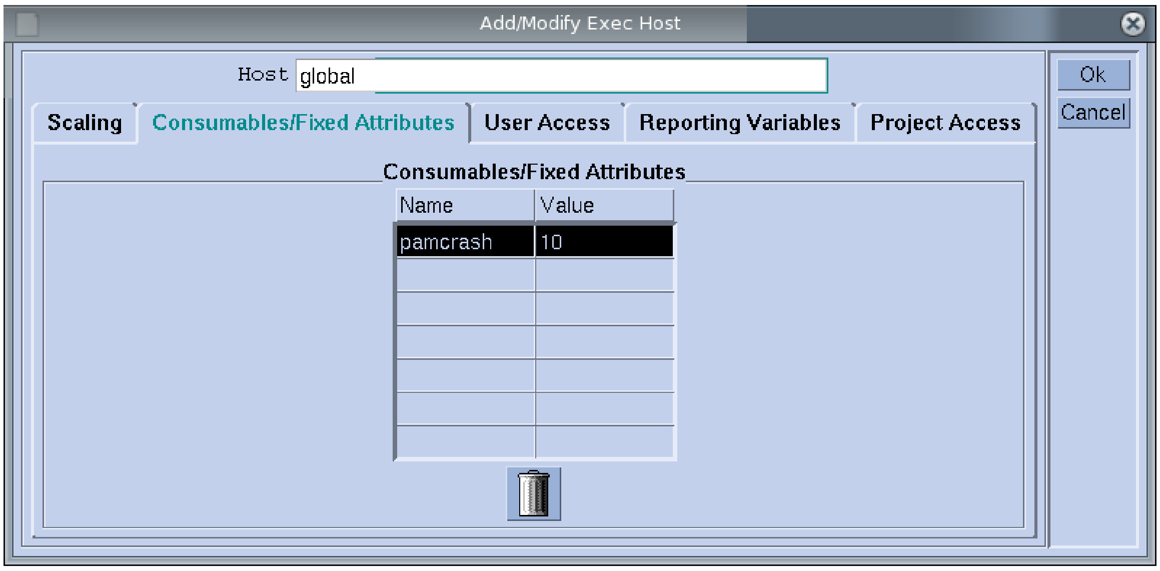
Name				uestable	Consumable	Default	Urgency
nastran	na	INT	<=	YES	YES	5	0
np_load_avg	nla	DOUBLE	>=	NO	NO	0	0
np_load_long	nll	DOUBLE	>=	NO	NO	0	0
np_load_medi	nlm	DOUBLE	>=	NO	NO	0	0
np_load_shor	nls	DOUBLE	>=	NO	NO	0	0
num_proc	p	INT	==	YES	NO	0	0
pamcrash	pc	INT	<=	FORCED	YES	0	0
qname	q	RESTRING	==	YES	NO	NONE	0
rerun	re	BOOL	==	NO	NO	0	0
s_core	s_core	MEMORY	<=	YES	NO	0	0
s_cpu	s_cpu	TIME	<=	YES	NO	0:0:0	0
s_data	s_data	MEMORY	<=	YES	NO	0	0
s_fsize	s_fsize	MEMORY	<=	YES	NO	0	0
s_rss	s_rss	MEMORY	<=	YES	NO	0	0

In the figure above:

- The name of the consumable attribute is set to `pam-crash`. You can use `pc` as a shortcut in the `qalter -l`, `qselect -l`, `qsh -l`, `qstat -l`, or `qsub -l` commands instead.
- The attribute type is defined to be an integer counter.
- The `Requestable` flag is set to `FORCED`. The user must request how many `pam-crash` licenses a job will occupy when the job is submitted.
- The `Consumable` flag specifies that the attribute is a consumable resource.
- The setting `Default` is irrelevant since `Requestable` is set to `FORCED`, which means that a request value must be received for this attribute with any job.

Consumables receive their value from the global, host, or queue configurations through the `complex_values` lists. See the `host_conf(5)` and `queue_conf(5)` man pages, as well as Configuring Queues and Configuring Hosts.

To activate resource planning for this attribute and for the cluster, the number of available `pam-crash` licenses must be defined in the global host configuration, as shown in the following figure.



Host: global

Name	Value
pamcrash	10

In this figure, the value for the attribute `pam-crash` is set to 10, corresponding to 10 floating licenses.

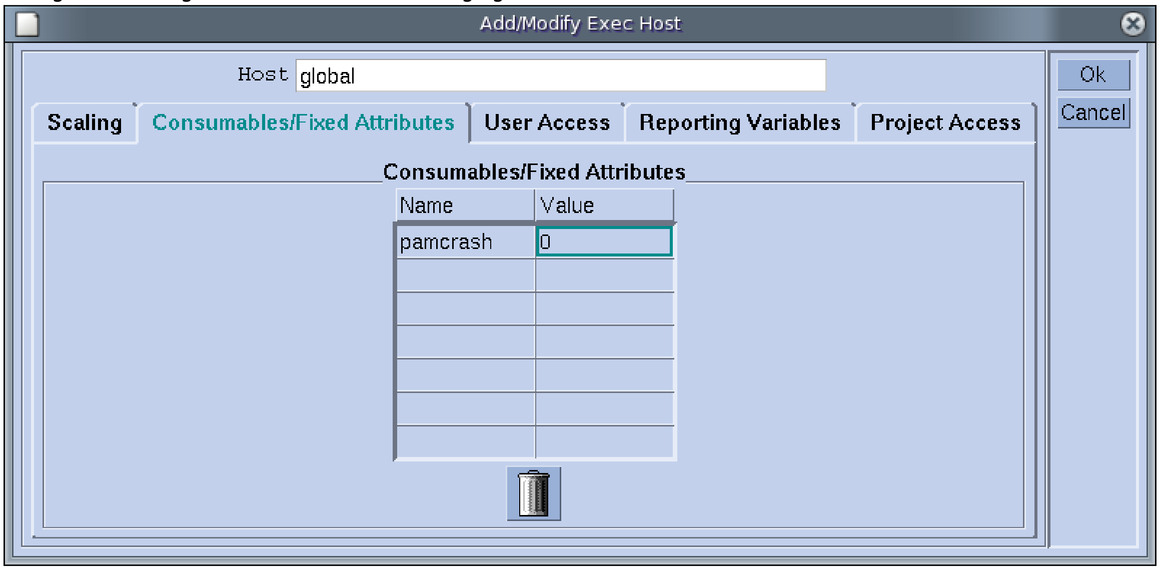
Note
The table `Consumables/Fixed Attributes` corresponds to the `complex_values` entry that is described in the host configuration file format `host_conf`(5).

Assume that a user submits the following job:

```
% qsub -l pc=1 pam-crash.sh
```

The job starts only if fewer than 10 `pam-crash` licenses are currently occupied. The job can run anywhere in the cluster, however, and the job occupies 1 `pam-crash` license throughout its run time.

One of your hosts in the cluster might not be able to be included in the floating license. For example, you might not have `pam-crash` binaries for that host. In such a case, you can exclude the host from the `pam-crash` license management. You can exclude the host by setting to zero the capacity that is related to that host for the consumable attribute `pam-crash`. To exclude the host, use the Execution Host tab of the Host Configuration dialog box as shown in the following figure.

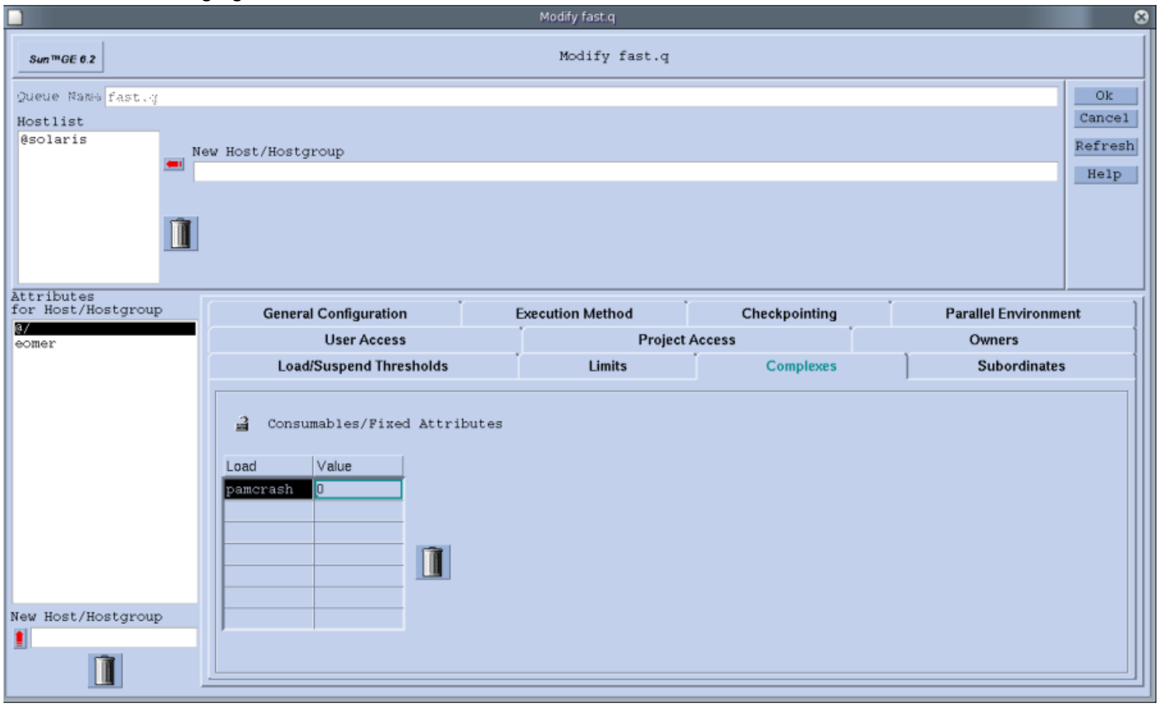


Host: global

Name	Value
pamcrash	0

Note
The `pam-crash` attribute is implicitly available to the execution host because the global attributes of the complex are inherited by all execution hosts. By setting the capacity to zero, you could also restrict the number of licenses that a host can manage to a nonzero value such as two. In this case, a maximum of two `pam-crash` jobs could coexist on that host.

Similarly, you might want to prevent a certain queue from running `pam-crash` jobs. For example, the queue might be an express queue with memory and CPU-time limits not suitable for `pam-crash`. In this case, set the corresponding capacity to zero in the queue configuration, as shown in the following figure.



Load	Value
pamcrash	0

> Note
> The `pam-crash` attribute is implicitly available to the queue because the global attributes of the complex are inherited by all queues.

Example 2 - Space Sharing for Virtual Memory

Administrators must often tune a system to avoid performance degradation caused by memory over-subscription, and consequently swapping of a machine. The Grid Engine software can support you in this task through the Consumable Resources facility.

The standard load parameter `virtual_free` reports the available free virtual memory, that is, the combination of available swap space and the available physical memory. To avoid swapping, the use of swap space must be minimized. In an ideal case, all the memory required by all processes running on a host should fit into physical memory.

The Grid Engine software can guarantee the availability of required memory for all jobs started through the Grid Engine system, given the following assumptions and configurations:

- `virtual_free` is configured as a consumable resource, and its capacity on each host is set to the available physical memory, or lower.
- Jobs request their anticipated memory usage, and the value that jobs request is not exceeded during run time.

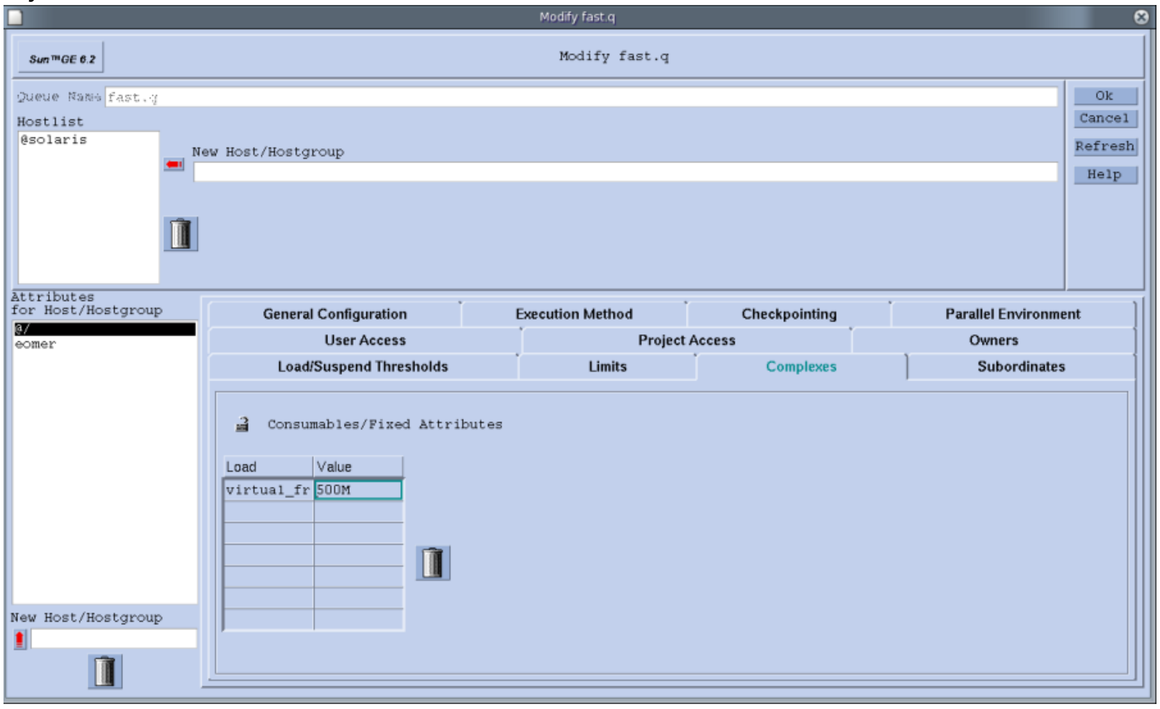
An example of a possible `virtual_free` resource definition is shown in the Complex Configuration Dialog Box: `virtual_free`. A corresponding execution host configuration for a host with one Gbyte of main memory is shown in Add-Modify Exec Host: `virtual_free`.

In the `virtual_free` resource definition example, the Requestable flag is set to YES instead of to FORCED, as in the example of a global configuration. This means that users need not indicate the memory requirements of their jobs. The value in the Default field is used if an explicit memory request is missing. The value of 1 Gbyte as default request in this case means that a job without a request is assumed to occupy all available physical memory.

Note
`virtual_free` is one of the standard load parameters of the Grid Engine system. The additional availability of recent memory statistics is taken into account automatically by the system in the virtual memory capacity planning. If the load report for free virtual memory falls below the value obtained by Grid Engine software internal bookkeeping, the load value is used to avoid memory over-subscription. Differences in the reported load values and the internal bookkeeping can occur easily if jobs are started without using the Grid Engine system.

If you run queues with different memory requirements on one machine, you might want to partition the memory that these queues use. This functionality is called space sharing. Then you assign to each queue a portion of the total memory on that host.

In the example, the queue configuration attaches half of the total memory that is available to host `carc` to the queue `fast.q` for the host `carc`. Hence the accumulated memory consumption of all jobs that are running in queue `fast.q` on host `carc` cannot exceed 500 Mbytes. Jobs in other queues are not taken into account. Nonetheless, the total memory consumption of all running jobs on host `carc` cannot exceed 1 Gbyte.



Queue Name fast.q

Hostlist
@solaris

Attributes for Host/Hostgroup
@/
eomer

Consumables/Fixed Attributes

Load	Value
virtual_fr	500M

Note
The attribute `virtual_free` is available to all queues through inheritance from the complex.

Users might submit jobs to a system configured similarly to the example in either of the following forms:

```
% qsub -l vf=100M honest.sh
% qsub dont_care.sh
```

The job submitted by the first command can be started as soon as at least 100 Mbytes of memory are available. This amount is taken into account in the capacity planning for the `virtual_free` consumable resource. The second job runs only if no other job is on the system, as the second job implicitly requests all the available memory. In addition, the second job cannot run in the queue `fast.q` because the job exceeds the queue's memory capacity.

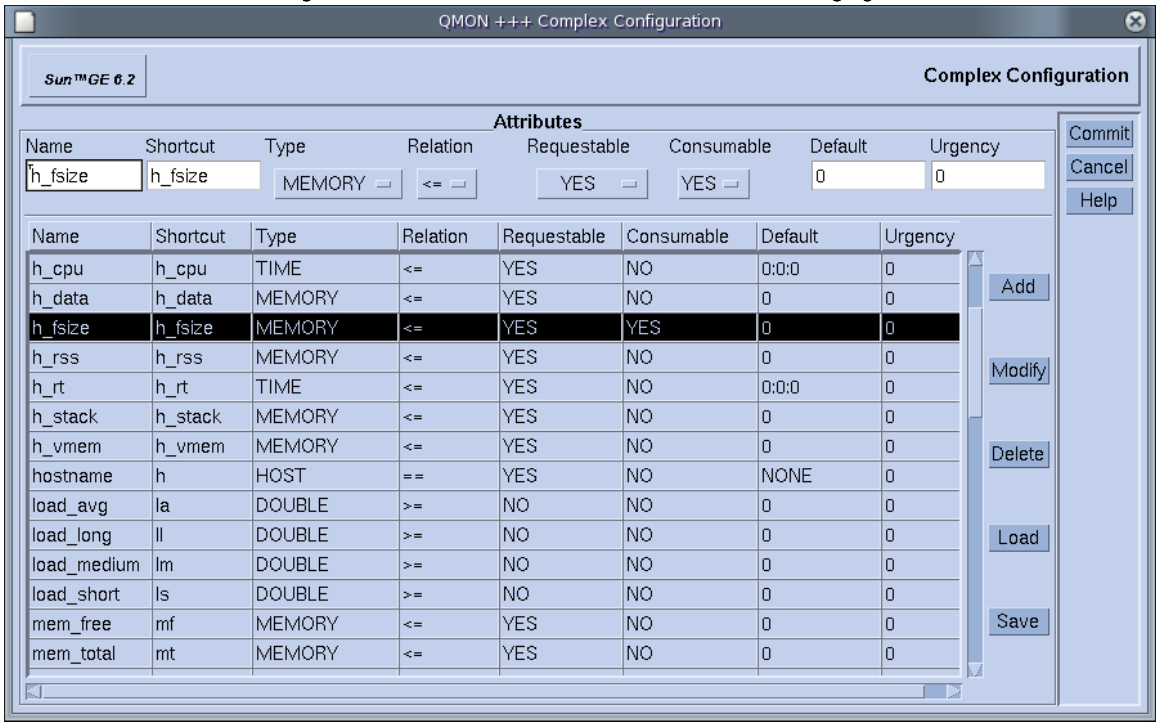
Example 3 - Managing Available Disk Space

Some applications need to manipulate huge data sets stored in files. Such applications therefore depend on the availability of sufficient disk space throughout their run time. This requirement is similar to the space-sharing of available memory, as discussed in the preceding example.

The main difference is that the Grid Engine system does not provide free disk space as one of its standard load parameters. Free disk space is not a standard load parameter because disks are usually partitioned into file systems in a site-specific way. Site-specific partitioning does not allow identifying the file system of interest automatically.

Nevertheless, available disk space can be managed efficiently by the system through the consumables resources facility. You should use the host resource attribute `h_fsize` for this purpose.

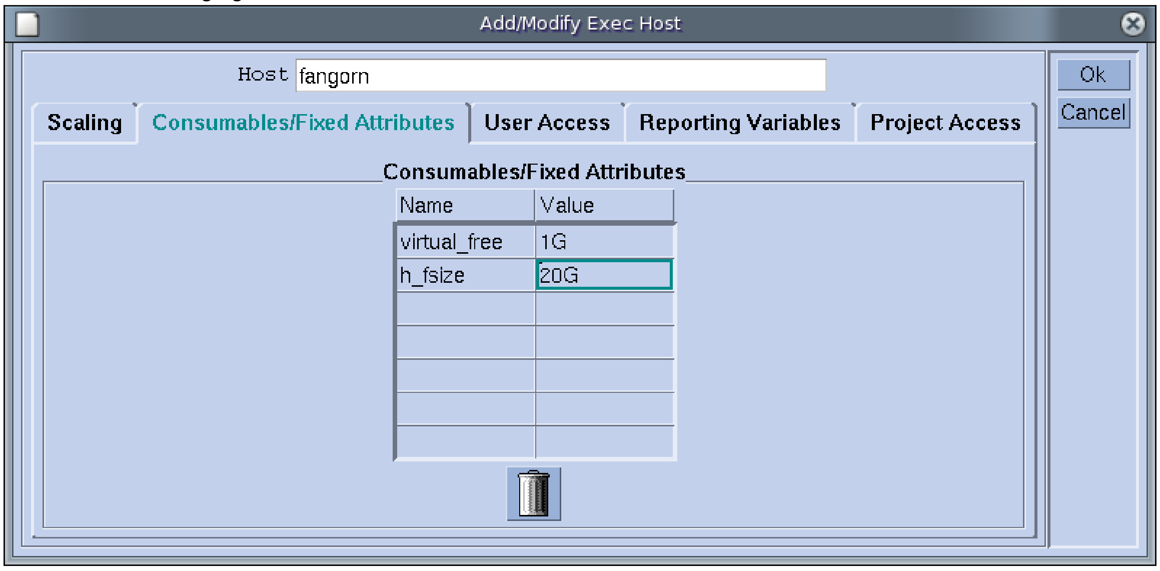
First, the attribute must be configured as a consumable resource, as shown in the following figure.



Name	Shortcut	Type	Relation	Requestable	Consumable	Default	Urgency
h_fsize	h_fsize	MEMORY	<=	YES	YES	0	0

Name	Shortcut	Type	Relation	Requestable	Consumable	Default	Urgency
h_cpu	h_cpu	TIME	<=	YES	NO	0:0:0	0
h_data	h_data	MEMORY	<=	YES	NO	0	0
h_fsize	h_fsize	MEMORY	<=	YES	YES	0	0
h_rss	h_rss	MEMORY	<=	YES	NO	0	0
h_rt	h_rt	TIME	<=	YES	NO	0:0:0	0
h_stack	h_stack	MEMORY	<=	YES	NO	0	0
h_vmem	h_vmem	MEMORY	<=	YES	NO	0	0
hostname	h	HOST	==	YES	NO	NONE	0
load_avg	la	DOUBLE	>=	NO	NO	0	0
load_long	ll	DOUBLE	>=	NO	NO	0	0
load_medium	lm	DOUBLE	>=	NO	NO	0	0
load_short	ls	DOUBLE	>=	NO	NO	0	0
mem_free	mf	MEMORY	<=	YES	NO	0	0
mem_total	mt	MEMORY	<=	YES	NO	0	0

In the case of local host file systems, a reasonable capacity definition for the disk space consumable can be put in the host configuration, as shown in the following figure.



Host fangorn

Name	Value
virtual_free	1G
h_fsize	20G

Submission of jobs to a Grid Engine system that is configured as described here works similarly to the previous examples:

```
% qsub -l hf=5G big-sort.sh
```

The `h_fsize` attribute is recommended because `h_fsize` also is used as the hard file size limit in the queue configuration. The file size limit restricts the ability of jobs to create files that are larger than what is specified during job submission. The `qsub` command in this example specifies a file size limit of 5 Gbytes. If the job does not request the attribute, the corresponding value from the queue configuration or host configuration is used. If the `Requestable` flag for `h_fsize` is set to `FORCED` in the example, a request must be included in the `qsub` command. If the `Requestable` flag is not set, a request is optional in the `qsub` command.

By using the queue limit as the consumable resource, you control requests that the user specifies instead of the real resource consumption by the job scripts. Any violation of the limit is sanctioned, which eventually aborts the job. The queue limit ensures that the resource requests on which the Grid Engine system internal capacity planning is based are reliable. See the `queue_conf(5)` and the `setrlimit(2)` man pages for details.

> Note
> Some operating systems provide only per-process file size limits. In this case, a job might create multiple files with a size up to the limit. On systems that support per-job file size limitation, the Grid Engine system uses this functionality with the `h_fsize` attribute. See the `queue_conf(5)` man page for further details.

You might want applications that are not submitted to the Grid Engine system to occupy disk space concurrently. If so, the internal bookkeeping might not be sufficient to prevent application failure due to lack of disk space. To avoid this problem, you can periodically receive statistics about disk space usage. These statistics indicate the total disk space consumption, including any space that is consumed outside of the Grid Engine system.

The load sensor interface enables you to enhance the set of standard load parameters with site-specific information, such as the available disk space on a file system. See Adding Site-Specific Load Parameters for more information.

By adding an appropriate load sensor and reporting free disk space for `h_fsize`, you can combine consumable resource management and resource availability statistics. The Grid Engine system compares job requirements for disk space with the available capacity and with the most recent reported load value. Available capacity is derived from the internal resource planning. Jobs get dispatched to a host only if both criteria are met.

Backing Up and Restoring System Configuration

You can back up your Grid Engine system configuration files automatically. The automatic backup process uses a configuration file called `backup_template.conf`. The backup configuration file is located by default in `$SGE_ROOT/util/install_modules/backup_template.conf`.

The backup configuration file must define the following elements:

- The Grid Engine system root directory (`$SGE_ROOT`).
- The Grid Engine system cell directory (`$SGE_CELL`).
- The Grid Engine system backup directory.
- Type of backup. Your backup can be just the Grid Engine system configuration files, or the backup can be a compressed tar file that contains the configuration files.
- The file name of the backup file.

The backup template file looks like the following example:

```
###################################################
# Autobackup Configuration File Template
###################################################

# Please, enter your $SGE_ROOT here (mandatory)
$SGE_ROOT=""

# Please, enter your $SGE_CELL here (mandatory)
$SGE_CELL=""

# Please, enter your Backup Directory here
# After backup you will find your backup files here (mandatory)
# The autobackup will add a time /date combination to this dirname
# to prevent an overwriting!
BACKUP_DIR=""

# Please, enter true to get a tar/gz package
# and false to copy the files only (mandatory)
TAR="true"

# Please, enter the backup file name here. (mandatory)
BACKUP_FILE="backup.tar"
```

To start the automatic backup process, type the following command on the `sge_qmaster` host:

```
inst_sge -bup -auto <backup-conf>
```

backup-conf is the full path to the backup configuration file.

Note
You do not need to shut down any of the Grid Engine system daemons before you back up your configuration files.

Your backup is created in the directory specified by BACKUP_FILE. A backup log file called `install.`pid is also created in this directory. pid is the process ID number.

Topic	Description
How to Perform a Manual Backup	Learn how to perform a manual backup.
How to Restore from a Backup	Learn how to restore from a backup.

How to Perform a Manual Backup

1. Type the following command to start a manual backup:

```
inst_sge -bup
```

2. Enter the **`$SGE_ROOT`** directory or use the default.

```
SGE Configuration Backup
------------------------

This feature does a backup of all configuration you made
within your cluster.
Please enter your $SGE_ROOT directory.
Default: [/home/user/ts/u10]
```

3. Enter the **$SGE_CELL** name or use the default.

```
Please enter your $SGE_CELL name. Default: [default]
```

4. Enter the backup destination directory or use the default.

```
Where do you want to save the backup files?
Default: [/home/user/ts/u10/backup]
```

5. Choose whether to create a compressed tar backup file.

   Caution
   If you create a compressed tar file, use the same tar binary to pack and unpack the files. Using different tar versions (gnu tar/ solaris tar) might result in corrupt tar packages.

```
Shall the backup function create a compressed tar package with your files? (y/n) [y] >>
```

6. Enter the file name of the backup file or use the default.

```
... starting with backup

Please enter a filename for your backupfile. Default: [backup.tar] >>
```

   Once the filename is specified, the backup process completes. Output similar to the following is displayed.

```
2007-01-11_22_43_22.dump
bootstrap
qtask
settings.sh
act_qmaster
sgemaster
settings.csh
sgeexecd
jobseqnum

... backup completed
All information is saved in
[/home/user/ts/u10/backup/backup.tar.gz[Z]]
```

How to Restore From a Backup

Caution
Shut down the `qmaster` daemon before you start the restore process. During the restore process, the spooling database is changed. If the `qmaster` and restore processes try to access the same data concurrently, data loss might result.

1. Type the following command to start the restore process:

```
inst_sge -rst
```

2. Read the messages on the screen and press Return.

```
SGE Configuration Restore
-------------------------

This feature restores the configuration from a backup you made
previously.

Hit, <ENTER> to continue!
```

3. Enter the **`$SGE_ROOT`** directory or use the default.

```
Please enter your $SGE_ROOT directory.
Default: [/home/user/ts/u10]
```

4. Enter the **`$SGE_CELL`** name or use the default.

```
Please enter your $SGE_CELL name. Default: [default]
```

5. Confirm the format of the backup file.
   The backup file can be in a format other than a compressed tar file.

```
Is your backupfile in tar.gz[Z] format? (y/n) [y]
```

6. Enter the full path to the backup file.

```
Please enter the full path and name of your backup file.
Default: [/home/user/ts/u10/backup/backup.tar.gz]
```

7. Verify the information for the spooling database.
   The restore feature unpacks the backup file and reads system information. To prevent data loss, confirm that the correct spooling

database is detected.

```
Copying backupfile to /tmp/bup_tmp_22_51_40

/home/user/ts/u10/backup/backup.tar.gz
2007-01-11_22_43_22.dump
bootstrap
qtask
settings.sh
act_qmaster
sgemaster
settings.csh
sgeexecd
jobseqnum

Spooling Method: berkeleydb detected!


The path to your spooling db is [/tmp/dom/spooldb]
If this is correct hit <ENTER> to continue, else enter the path. >>
```

8. Restart **qmaster**.

Configuring Load Parameters

- Default Load Parameters
- Adding Site-Specific Load Parameters
- Load Parameter Configuration Tasks

Default Load Parameters

By default, `sge_execd` periodically reports several load parameters and their corresponding values to `sge_qmaster`. These values are stored in the `sge_qmaster` internal host object, which is described in About Hosts and Daemons. However, the values are used internally only if a complex resource attribute with a corresponding name is defined. Such complex resource attributes contain the definition as to how load values are to be interpreted. See Assigning Resource Attributes to Queues, Hosts, and the Global Cluster for more information.

After the primary installation, a standard set of load parameters is reported. All attributes required for the standard load parameters are defined as host-related attributes. Subsequent releases of Grid Engine software might provide extended sets of default load parameters, therefore the set of load parameters that is reported by default is documented in the file `$SGE_ROOT/doc/load_parameters.asc`.

How load attributes are defined determines their accessibility. By defining load parameters as global resource attributes, you make them available for the entire cluster and for all hosts. By defining load parameters as host-related attributes, you provide the attributes for all hosts but not for the global cluster.

> Note
> Do not define load attributes as queue attributes. Queue attributes would not be available to any host nor to the cluster.

Adding Site-Specific Load Parameters

The set of default load parameters might not be adequate to completely describe the load situation in a cluster. This possibility is especially likely with respect to site-specific policies, applications, and configurations. The Grid Engine software provides the means to extend the set of load parameters. For this purpose, `sge_execd` offers an interface to feed load parameters and the current load values into `sge_execd`. Afterwards, these parameters are treated like the default load parameters. As for the default load parameters, corresponding attributes must be defined in the complex for the site-specific load parameters to become effective. See Default Load Parameters for more information.

Load Parameter Configuration Tasks

Topic	Description
Writing Your Own Load Sensors	Learn how to write your own load sensors.

Writing Your Own Load Sensors

To feed sge_execd with additional load information, you must supply a load sensor. If needed, multiple load sensors can be configured. The load sensor can be a script or a binary executable. In either case, the load sensor's handling of the standard input and standard output streams and its control flow must comply with the following rules:

- The load sensor must be written as an infinite loop that waits at a certain point for input from STDIN.
- If the string quit is read from STDIN, the load sensor is supposed to exit.
- As soon as an end-of-line is read from STDIN, a retrieval cycle for loading data is supposed to start.

The load sensor then performs whatever operation is necessary to compute the desired load figures. At the end of the cycle, the load sensor writes the result to STDOUT.

Note
If load retrieval takes a long time, the load measurement process can be started immediately after sending a load report. When quit is received, the load values are then available to be sent.

Load Sensor Rules Format

The format for the load sensor rules is as follows:

- A load value report starts with a line that contains nothing but the word begin.
- Individual load values are separated by newlines.
- Each load value consists of three parts separated by colons (:) and contains no blanks.
- The first part of a load value is either the name of the host for which load is reported or the special name global.
- The second part of the load sensor is the symbolic name of the load value, as defined in the complex. See the complex(5) man page for details. If a load value is reported for which no entry in the complex exists, the reported load value is not used.
- The third part of the load sensor is the measured load value. A load value report ends with a line that contains the word end.

There are examples of load sensors installed in the $SGE_ROOT/util/resources/loadsensors directory.

Example of a Load Sensor Script

The following example shows a load sensor. The load sensor is a Bourne shell script.

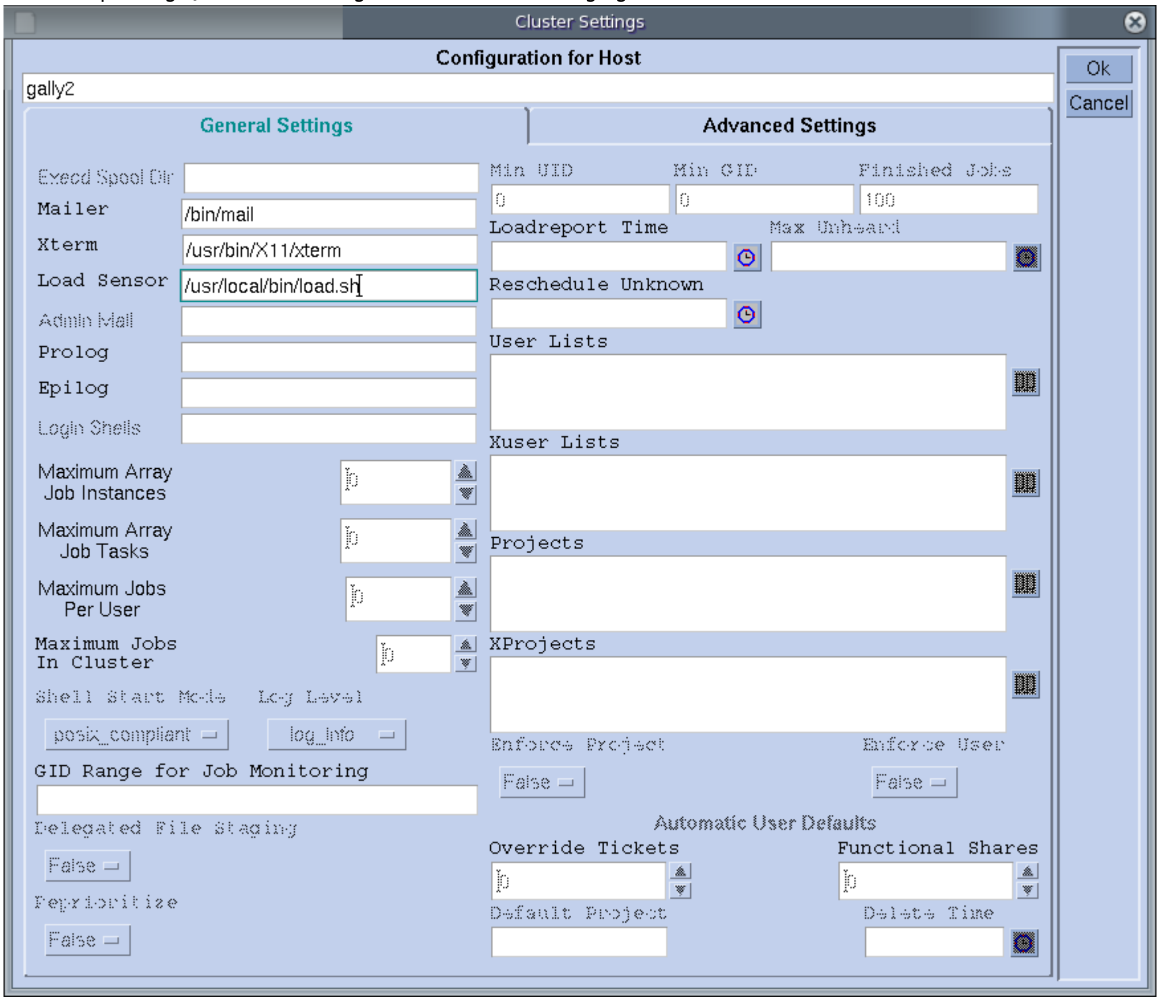
Example – Load Sensor Bourne Shell Script

```
#!/bin/sh

myhost=`uname -n`

while [ 1 ]; do
     # wait for input
     read input
     result=$?
     if [ $result != 0 ]; then
          exit 1
     fi
     if [ "$input" = quit ]; then
          exit 0
     fi
     #send users logged in
     logins=`who | cut -f1 -d" " | sort | uniq | wc -l | sed "s/^ *//"`
echo begin
     echo "$myhost:logins:$logins"
     echo end
done

# we never get here

exit 0
```

Save this script to the file `load.sh`. Assign executable permission to the file with the chmod command. To test the script interactively from the command line, type `load.sh` and repeatedly press the `Return` key.

As soon as the procedure works, you can install it for any execution host. To install the procedure, configure the load sensor path as the `load_sensor` parameter for the cluster configuration, global configuration, or the host-specific configuration. See Configuring Clusters or the `sge_conf(5)` man page for more information.

The corresponding QMON window might look like the following figure:



Cluster Settings

Configuration for Host

gally2

General Settings | Advanced Settings

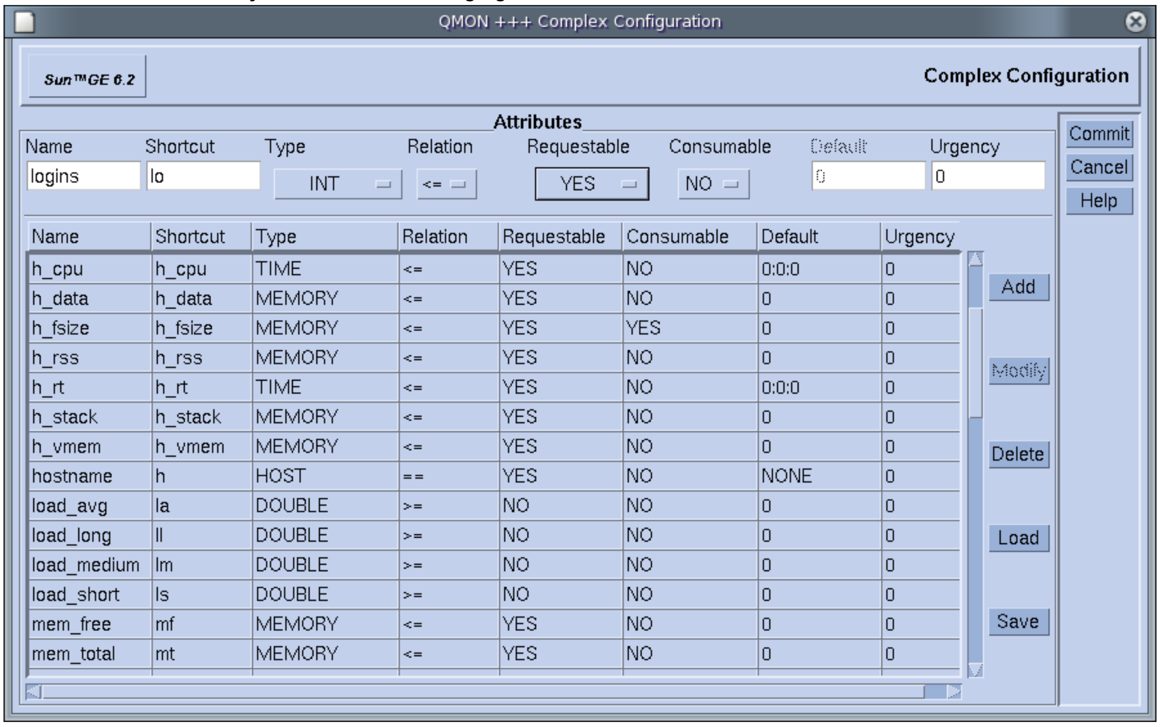
Execd Spool Dir:
Mailer: /bin/mail
Xterm: /usr/bin/X11/xterm
Load Sensor: /usr/local/bin/load.sh
Admin Mail:
Prolog:
Epilog:
Login Shells:
Maximum Array Job Instances: 0
Maximum Array Job Tasks: 0
Maximum Jobs Per User: 0
Maximum Jobs In Cluster: 0
Shell Start Mode: posix_compliant
Log Level: log_info
GID Range for Job Monitoring:
Delegated File Staging: False
Reprioritize: False

Min UID: 0
Min GID: 0
Finished Jobs: 100
Loadreport Time:
Max Unheard:
Reschedule Unknown:
User Lists:
Xuser Lists:
Projects:
XProjects:
Enforce Project: False
Enforce User: False

Automatic User Defaults
Override Tickets: 0
Functional Shares: 0
Default Project:
Delete Time:

Ok
Cancel

The reported load parameter `logins` is usable as soon as a corresponding attribute is added to the complex. The required definition might look like the last table entry shown in the following figure.



Name	Shortcut	Type	Relation	Requestable	Consumable	Default	Urgency
logins	lo	INT	<=	YES	NO	0	0

Name	Shortcut	Type	Relation	Requestable	Consumable	Default	Urgency
h_cpu	h_cpu	TIME	<=	YES	NO	0:0:0	0
h_data	h_data	MEMORY	<=	YES	NO	0	0
h_fsize	h_fsize	MEMORY	<=	YES	YES	0	0
h_rss	h_rss	MEMORY	<=	YES	NO	0	0
h_rt	h_rt	TIME	<=	YES	NO	0:0:0	0
h_stack	h_stack	MEMORY	<=	YES	NO	0	0
h_vmem	h_vmem	MEMORY	<=	YES	NO	0	0
hostname	h	HOST	==	YES	NO	NONE	0
load_avg	la	DOUBLE	>=	NO	NO	0	0
load_long	ll	DOUBLE	>=	NO	NO	0	0
load_medium	lm	DOUBLE	>=	NO	NO	0	0
load_short	ls	DOUBLE	>=	NO	NO	0	0
mem_free	mf	MEMORY	<=	YES	NO	0	0
mem_total	mt	MEMORY	<=	YES	NO	0	0

Configuring Queues

- Introduction to Queues
- Cluster Queues
- Queue Configuration Tasks

Introduction to Queues

A queue is a container for a class of jobs that are allowed to run on one or more hosts concurrently. A queue determines certain job attributes, for example, whether the job can be migrated. Throughout its lifetime, a running job is associated with its queue. Association with a queue affects some of the things that can happen to a job. For example, if a queue is suspended, all jobs associated with that queue are also suspended.

Jobs do not need to be submitted directly to a queue. If you submit a job to a specified queue, the job is bound to this queue. As a result, the Grid Engine system daemons are unable to select a better-suited device or a device that has a lighter load.

You only need to specify the requirement profile of the job. A profile might include requirements such as memory, operating system, available software, and so forth. The Grid Engine software automatically dispatches the job to a suitable queue and a suitable host with a light execution load.

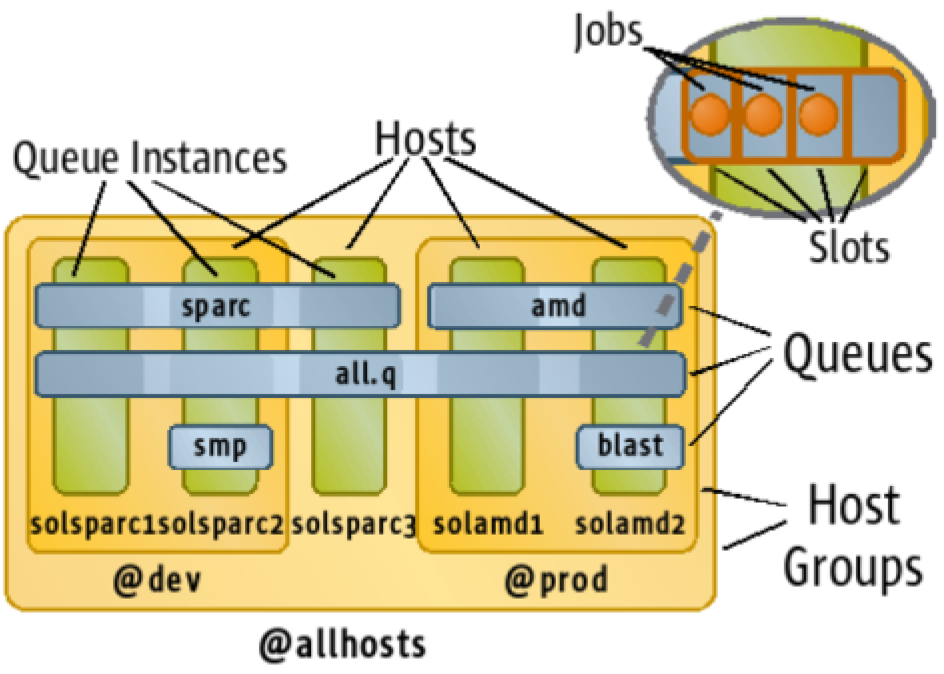
A queue can reside on a single host, or a queue can extend across multiple hosts. For this reason, Grid Engine system queues are also referred to as cluster queues. Cluster queues enable users and administrators to work with a cluster of execution hosts by means of a single queue configuration. Each host that is attached to a cluster queue receives its own queue instance from the cluster queue.

The following concepts are helpful:

- Host Groups – Host groups enable you to manage multiple hosts by means of a single host group configuration. For more on host groups, see How to Configure Host Groups.
- Hosts – A host is a system on which Grid Engine system functions occur. For more information about hosts, see Configuring Hosts.
- Cluster Queues – Cluster queues are queues that extend across multiple hosts.
- Queue Domains – Queue domains enable you to manage groups of queue instances that are part of the same cluster queue and whose assigned hosts are part of the same host group.
- Queue Instances – A queue is composed of a series of queue instances, one per execution host. Each queue instance inherits all the properties of the queue to which it belongs. Jobs always run in queue instances, not cluster queues.

- Job Slots – The number of jobs that are allowed to run concurrently in the queue.

The following figure illustrates how queues (sparc, amd, all.q, smp, and blast) fit into the Sun Grid Engine system:



In this example, the host group, @allhosts, contains the host, solsparc3, and the host groups, @dev and @prod. The host group, @dev, contains hosts solsparc1 and solsparc2, and the host group, @prod, contains the hosts solamd1 and solamd2. The queue, all.q, spans all hosts in the cluster. The queue, sparc, spans only the hosts with UltraSPARC™ processors. Similarly, the queue, amd, spans the hosts with AMD processors. In each of these three queues, the area where the queue intersects with the host is shaded slightly. The shading represents a queue instance. The queue, smp, is available only on the host, solsparc2, presumably because that host is the only large SMP host in the cluster.

Cluster Queues

Each host that is associated with a cluster queue receives a queue instance of that cluster queue, which resides on that host. By configuring individual queue instances, you can manage a heterogeneous cluster of execution hosts by means of a single cluster queue configuration.

When you modify a cluster queue, all of its queue instances are modified simultaneously. Within a single cluster queue, you can specify differences in the configuration of queue instances. Consequently, a typical setup might have only a few cluster queues, and the queue instances controlled by those cluster queues remain largely in the background.

When you configure a cluster queue, you can associate any combination of the following host objects with the cluster queue:

- One execution host
- A list of separate execution hosts
- One or more host groups

When you associate individual hosts with a cluster queue, the name of the resulting queue instance on each host combines the cluster queue name with the host name. The cluster queue name and the host name are separated by an @ sign. For example, if you associate the host myexechost with the cluster queue myqueue, the name of the queue instance on myexechost is myqueue@myexechost.

When you associate a host group with a cluster queue, you create what is known as a queue domain. Queue domains enable you to manage groups of queue instances that are part of the same cluster queue and whose assigned hosts are part of the same host group. A queue domain name combines a cluster queue name with a host group name, separated by an @ sign. For example, if you associate the host group @myhostgroup with the cluster queue myqueue, the name of the queue domain is myqueue@@myhostgroup. Queue domain names always include two @ signs, because all host group names begin with an @ sign.

Configuring queues registers the queue attributes with sge_qmaster. As soon as queues are configured, they are instantly visibly to the whole cluster and to all users on all hosts belonging to the Grid Engine system.

Queue Configuration Tasks

Task	User Interface	Description
How to Display Queues	CLI or QMON	Learn how to display the configuration for a queue.
How to Add Queues	CLI or QMON	Learn how to add a queue to your Grid Engine Cluster.
How to Configure General Parameters for Queues	CLI or QMON	Learn how to configure general parameters for a queue.
How to Configure Execution Method Parameters for Queues	CLI or QMON	Learn how to configure the parameters for the prolog script, epilog script, `starter_method`, `suspend_method`, `resume_method`, and `terminate_method`.
How to Configure Checkpointing Parameters	CLI or QMON	Learn how to configure the checkpointing parameters for a queue.
How to Configure Parallel Environment Parameters	CLI or QMON	Learn how to configure the parallel environment parameter for a queue.
How to Configure Load and Suspend Thresholds	CLI or QMON	Learn how to configure load and suspend threshold parameters for a queue.
How to Configure Complex Resource Attributes	CLI or QMON	Learn how to configure complex resource attributes for a queue.
How to Configure Resource Limits	CLI or QMON	Learn how to configure resource limits for a queue.
How to Configure Subordinate Queues	CLI or QMON	Learn how to subordinate queues on the same host to the queue that you are configuring.
How to Configure User Access Parameters	CLI or QMON	Learn how to configure user access to a queue.
How to Configure Project Access Parameters	CLI or QMON	Learn how to configure project access to a queue.
How to Configure Owners Parameters	CLI or QMON	Learn how to configure the owners parameters for a queue.

How to Display Queues From the Command Line

To display queues and a cluster queue list from the command line, use the following arguments for the `qconf` command:

- To display a queue, type the following command:

  ```
  qconf -sq [ <queue> [,...] ]
  ```

  The `-sq` option (show queue) without arguments displays the default template cluster queue, queue domain, or queue instance configuration. The `-sq` option with arguments displays the current configuration of the specified queues.

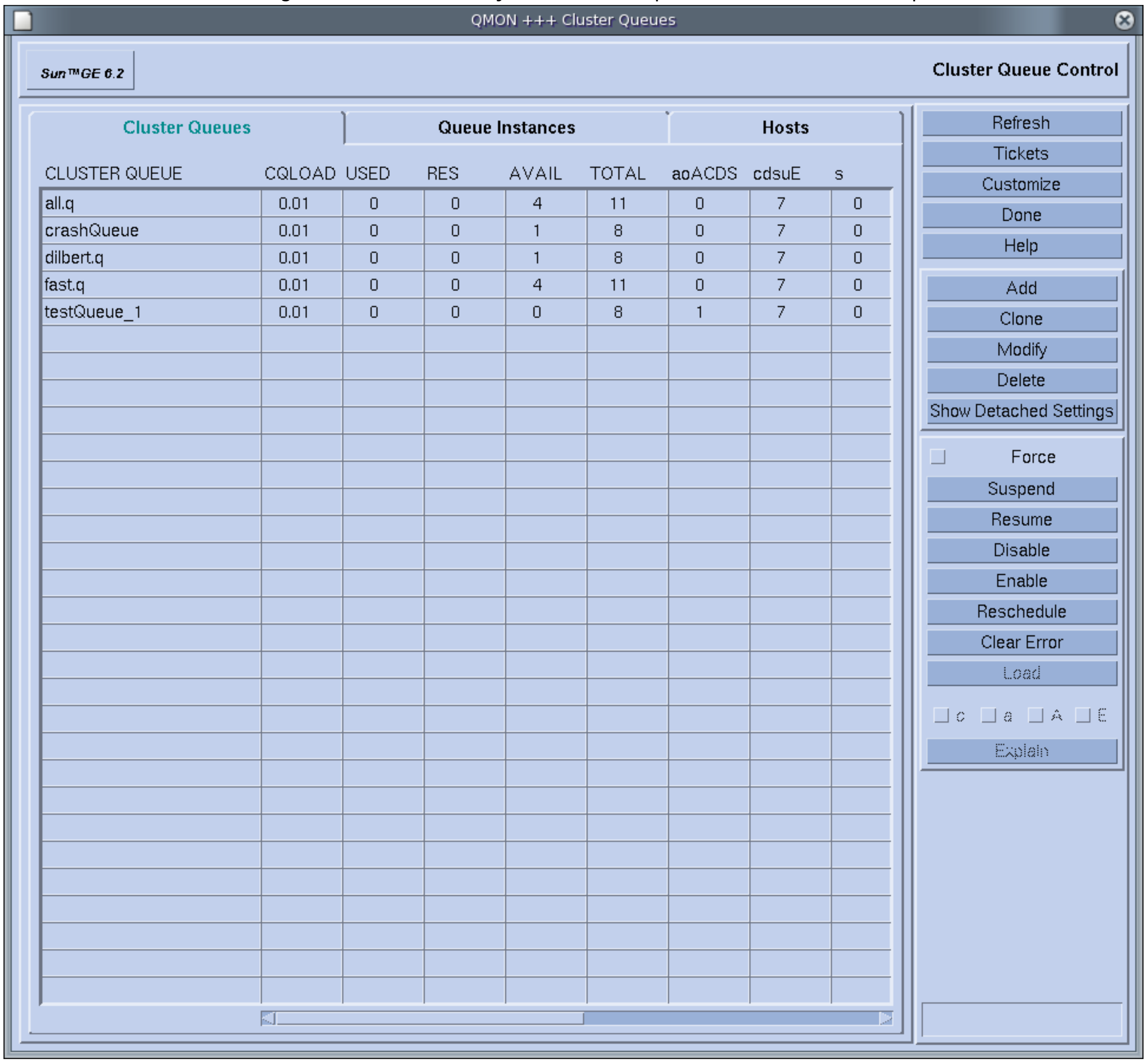
- To display the cluster queue list, type the following command:

  ```
  qconf -sql
  ```

  The `-sql` option (show cluster queue list) displays a list of all currently configured cluster queues.

How to Display Queues With QMON

To display queues with QMON, click the Queue Control button.
From the Cluster Queues dialog box, as shown below, you can view all queue instances and cluster queues.



CLUSTER QUEUE	CQLOAD	USED	RES	AVAIL	TOTAL	aoACDS	cdsuE	s
all.q	0.01	0	0	4	11	0	7	0
crashQueue	0.01	0	0	1	8	0	7	0
dilbert.q	0.01	0	0	1	8	0	7	0
fast.q	0.01	0	0	4	11	0	7	0
testQueue_1	0.01	0	0	0	8	1	7	0

How to Add Queues From the Command Line

- To add a cluster queue, type the following command:

```
qconf - aq [ <cluster_queue> ]
```

  The `-aq` option (add cluster queue) displays an editor containing a template for cluster queue configuration. The editor is either the default `vi` editor or an editor defined by the `EDITOR` environment variable. If cluster-queue is specified, the configuration of this cluster queue is used as template. Configure the cluster queue by changing the template and then saving it. See the `queue_conf(5)` man page for a detailed description of the template entries to change.

- To add a cluster queue from a file, type the following command:

```
qconf -Aq <filename>
```

The -Aq option (add cluster queue from file) uses the file filename to define a cluster queue. The definition file might have been produced by the `qconf -sq` queue command.

How to Add Queues With QMON

1. On the QMON Main Control window, click the Queue Control button.
   The Cluster Queues dialog box appears.

2. From the Cluster Queues dialog box, you can choose between adding a new queue or importing the parameters from an existing queue.
   - To add a new queue, do the following:
     a. Click add.
        The Add dialog box appears.

        > If you are adding a new cluster queue, you must specify a queue name and the names of the hosts on which the queue instances are to reside.

     b. Click Ok to save your changes.
        Click Cancel to leave the dialog box without saving your changes.
   - To import all parameters of an existing cluster queue, do the following:
     a. Select a queue and then click Clone.
     b. Click Ok to save your changes.
        Click Cancel to leave the dialog box without saving your changes.

How to Configure General Parameters for Queues From the Command Line

To modify a cluster queue, type the following command:

```
qconf -mq <cluster_queue>
```

The `-mq` option (modify cluster queue) modifies the specified cluster queue. The `-mq` option displays an editor containing the configuration of the cluster queue to be changed. The editor is either the default `vi` editor or an editor defined by the `EDITOR` environment variable. Modify the cluster queue by changing the configuration and then saving your changes.

You can configure the following general parameters from within the editor:

- `qname` – The name of the queue. The default value is `Template`.
- `hostlist` – A list of host names or host group names on which the queue runs. The default value is `NONE`.
- `seq_no` – The sequence number of the queue. The default value is `0`.
- `priority` – The nice value, or scheduling priority, at which jobs in this queue will be run. The default value is `0`, which means no priority is set. Values can range from `-20`, the highest scheduling priority value, to `20`, the lowest scheduling priority value.
- `processors` – A specifier set to be used by the jobs running in that queue. For some operating system architectures, this specifier can be a range, such as 1-4,8,10, or just an integer identifier of the processor set. See the `arc_depend_*.asc` files in the doc directory of your Grid Engine software distribution for more information.

  > Caution
  > Do not change this value unless you are certain that you need to change it.

- `tmpdir` – The temporary directory path. The default value is `/tmp`.
- `shell` – Default command interpreter to use for running job scripts. The default value is `bin/csh`.

- `shell_start_mode` – The mode in which to start the job script. The default value is `posix_compliant`.
- `qtype` – The type of the queue and of the jobs that are allowed to run in this queue. Type can be Batch, Interactive, or both.
- `rerun` – Defines the default behavior for jobs on the queue host that are aborted when the system crashes or the system is manually shut down.
- `calendar` – Specifies the calendar associated with this queue, if there is one. The default value for this parameter is `NONE`.
- `priority` – Specifies the nice value at which jobs in this queue will be run.
- `initial state` – Specifies the initial state (default, enabled, disabled) for the queue.
- `slots` – The maximum number of concurrently executing jobs allowed in the queue.

How to Configure General Parameters for Queues With QMON

1. On the QMON Main Control window, click the Queue Control button.
   The Cluster Queues dialog box appears.

2. Select a queue and then click Modify.
   The Queue Configuration dialog box appears.

3. To configure general parameters, click the General Configuration tab.

4. From the General Configuration tab, you can modify the following parameters:
   - Sequence Nr – The sequence number of the queue.
   - Processors – A specifier for the processor set to be used by the jobs running in that queue. For some operating system architectures, this specifier can be a range, such as 1-4,8,10, or just an integer identifier of the processor set. See the `arc_depend_*.asc` files in the `doc` directory of your Grid Engine software distribution for more information.

     > Caution
     > Do not change this value unless you are certain that you need to change it.

   - `tmp` Directory – Temporary directory path.
   - Shell – Default command interpreter to use for running the job scripts.
   - Shell Start Mode – The mode in which to start the job script.
   - Initial State – The state in which a newly added queue comes up. Also, the state in which a queue instance is restored if the `sge_execd` running on the queue instance host gets restarted.
   - Rerun Jobs – The queue's default rerun policy to be enforced on jobs that were aborted, for example, due to system crashes. The user can override this policy using the `qsub -r` command or the Submit Job dialog box. See the Extended Job Example.
   - Calendar – A calendar attached to the queue. This calendar defines on-duty and off-duty times for the queue.
   - Notify Time – The time to wait between delivery of `SIGUSR1/SIGUSR2` notification signals and suspend or kill signals.
   - Job's Nice – The nice value with which to start the jobs in this queue. `0` means use the system default.
   - Slots – The number of jobs that are allowed to run concurrently in the queue. Slots are also referred to as job slots.
   - Type – The type of the queue and of the jobs that are allowed to run in this queue. Type can be Batch, Interactive, or both.

5. Click Ok to save your changes.
   Click Cancel to leave the dialog box without saving your changes.

See the `queue_conf(5)` man page for detailed information about these parameters.

How to Configure Execution Method Parameters for Queues From the Command Line

To modify a cluster queue, type the following command:

```
qconf -mq <cluster_queue>
```

The `-mq` option (modify cluster queue) modifies the specified cluster queue. The `-mq` option displays an editor containing the configuration of the cluster queue to be changed. The editor is either the default `vi` editor or an editor defined by the `EDITOR` environment variable. Modify the cluster queue by changing the configuration and then saving your changes.

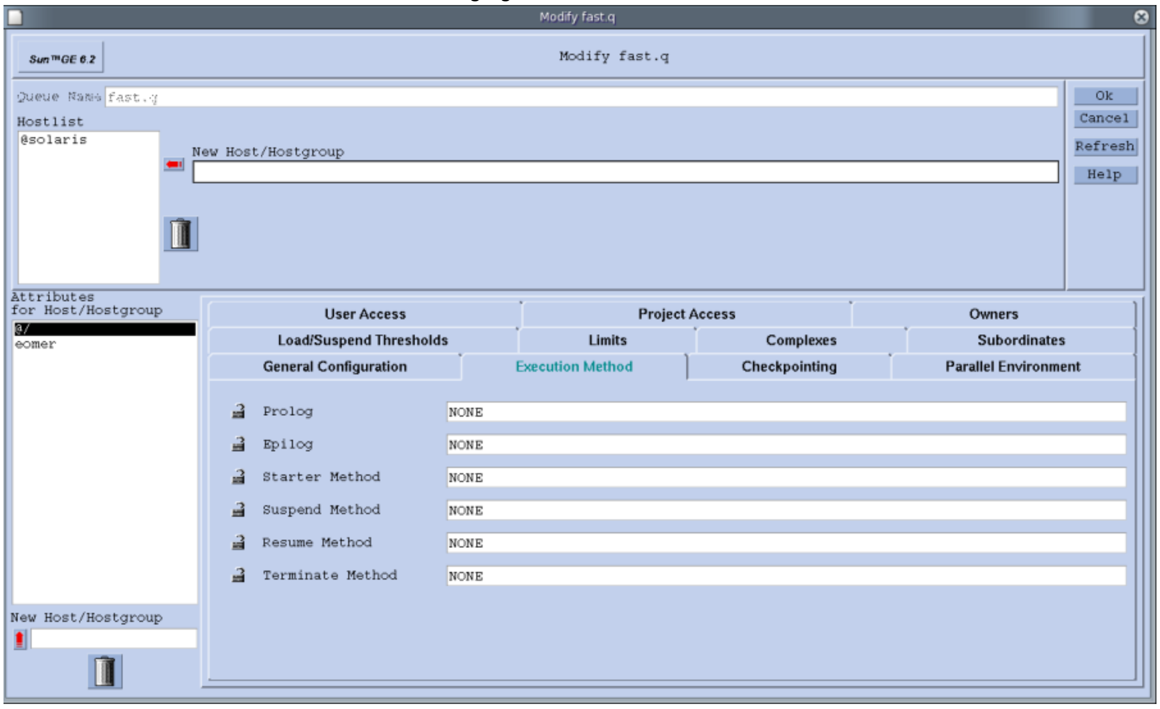
You can configure the following execution method parameters from within the editor:

- `prolog` – A queue-specific prolog script. The prolog script is run with the same environment as the job before the job script is started. This procedure is intended as a means for the Sun Grid Engine administrator to automate the execution of general site specific tasks like the preparation of temporary file systems with the need for the same context information as the job.
- `epilog` – A queue-specific epilog script. The eplilog script is run with the same environment as the job after the job script is started. This procedure is intended as a means for the Sun Grid Engine administrator to automate the execution of general site specific tasks like the cleaning up of temporary file systems with the need for the same context information as the job.
  A queue-specific epilog script. The epilog script is run with the same environment as the job after the job is finished.
- `starter_method` – The specified executable path will be used as a job starter facility responsible for starting batch jobs.
- `suspend_method` – Used to overwrite the default method used by Grid Engine for suspension of a job.
- `resume_method` – Used to overwrite the default method used by Grid Engine for the release of a suspension for a job.
- `terminate_method` – Used to overwrite the default method used by Grid Engine for suspension of a job.

How to Configure Execution Method Parameters With QMON

1. On the QMON Main Control window, click the Queue Control button.
   The Cluster Queues dialog box appears.

2. Select a queue and then click Modify.
   The Queue Configuration dialog box appears.

3. To configure execution method parameters, click the Execution Method tab.
   The Execution Method tab is shown in the following figure.
   
   Modify fast.q
   Sun™GE 6.2
   Modify fast.q
   Queue Name fast.q
   Hostlist
   @solaris
   New Host/Hostgroup
   Attributes for Host/Hostgroup
   @/
   eomer
   New Host/Hostgroup
   User Access | Project Access | Owners
   Load/Suspend Thresholds | Limits | Complexes | Subordinates
   General Configuration | Execution Method | Checkpointing | Parallel Environment
   Prolog NONE
   Epilog NONE
   Starter Method NONE
   Suspend Method NONE
   Resume Method NONE
   Terminate Method NONE

4. From the Execution Method tab, you can modify the following parameters:
   - Prolog – A queue-specific prolog script. The prolog script is run with the same environment as the job before the job script is started.
   - Epilog – A queue-specific epilog script. The epilog script is run with the same environment as the job after the job is finished.
   - Starter Method, Suspend Method, Resume Method, Terminate Method – Use these fields to override the default methods for applying these actions to jobs.

5. Click Ok to save your changes.
   Click Cancel to leave the dialog box without saving your changes.

See the queue_conf(5) man page for detailed information about these parameters.

How to Configure Parallel Environment Parameters From the Command Line

Note
To enable a queue to operate correctly in a parallel environment (PE), you must associate the queue with the PE. This association enables more control of the resources and lets you assign specific queues to handle the parallel workload.

To configure parallel environment parameters from the command line, use the following arguments for the `qconf` command:

To modify a cluster queue, type the following command:

```
qconf -mq <cluster_queue>
```

The `-mq` option (modify cluster queue) modifies the specified cluster queue. The `-mq` option displays an editor containing the configuration of the cluster queue to be changed. The editor is either the default `vi` editor or an editor defined by the `EDITOR` environment variable. Modify the cluster queue by changing the configuration and then saving your changes.

You can configure the following parallel environment parameter from within the editor:

- `pe_list` – The list of administrator-defined parallel environment interface names. The default is NONE. For more information, see the queue_conf(5) man page.

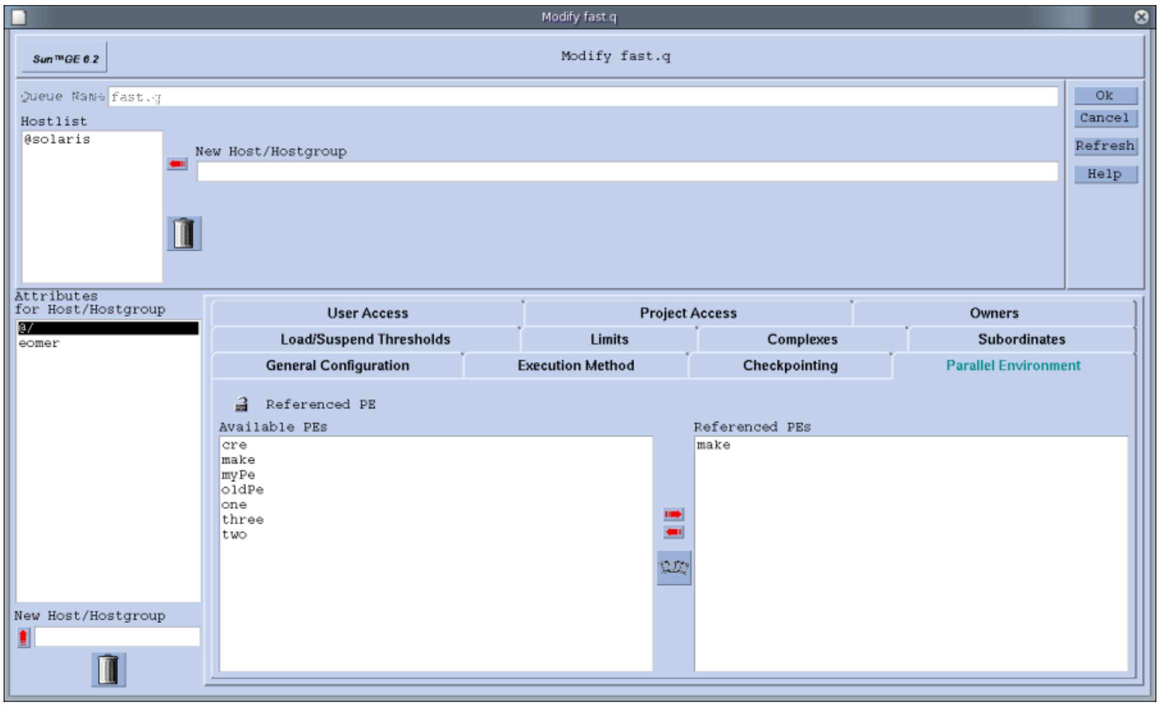
How to Configure Parallel Environment Parameters With QMON

Note
To enable a queue to operate correctly in a parallel environment (PE), you must associate the queue with the PE. This association enables more control of the resources and lets you assign specific queues to handle the parallel workload.

1. On the QMON Main Control window, click the Queue Control button.
   The Cluster Queues dialog box appears.

2. Select a queue and then click Modify.
   The Queue Configuration dialog box appears.

3. To configure parallel environment parameters, click the Parallel Environment tab.
   The Parallel Environment tab is shown in the following figure.



4. You can perform the following tasks from the Parallel Environment tab:
   - To reference a parallel environment from the queue, select the name of a parallel environment from the Available PEs list, and then click the right arrow to add it to the Referenced PEs list.
   - To remove a checkpointing environment from the Referenced PEs list, select it, and then click the left arrow.*
   - To add or modify parallel environments, click the button below the red arrows to open the Parallel Environment Configuration dialog box.*

5. Click Ok to save your changes.
   Click Cancel to leave the dialog box without saving your changes.

For more information, see Configuring Parallel Environments With QMON.
See the `queue_conf(5)` man page for detailed information about this parameter.

How to Configure Checkpointing Parameters From the Command Line

To configure checkpointing parameters from the command line, use the following arguments for the `qconf` command:

To modify a cluster queue, type the following command:

```
qconf -mq <cluster_queue>
```

The `-mq` option (modify cluster queue) modifies the specified cluster queue. The `-mq` option displays an editor containing the configuration of the cluster queue to be changed. The editor is either the default `vi` editor or an editor defined by the `EDITOR` environment variable. Modify the cluster queue by changing the configuration and then saving your changes. For more information, see the `qconf(1)` man page.

You can configure the following checkpointing parameter from within the editor:

- `ckpt_list` – The list of administrator-defined checkpointing interface names. The default is NONE.
- `min_cpu_interval` – The time between two automatic checkpoints in the case of transparent checkpointing jobs.

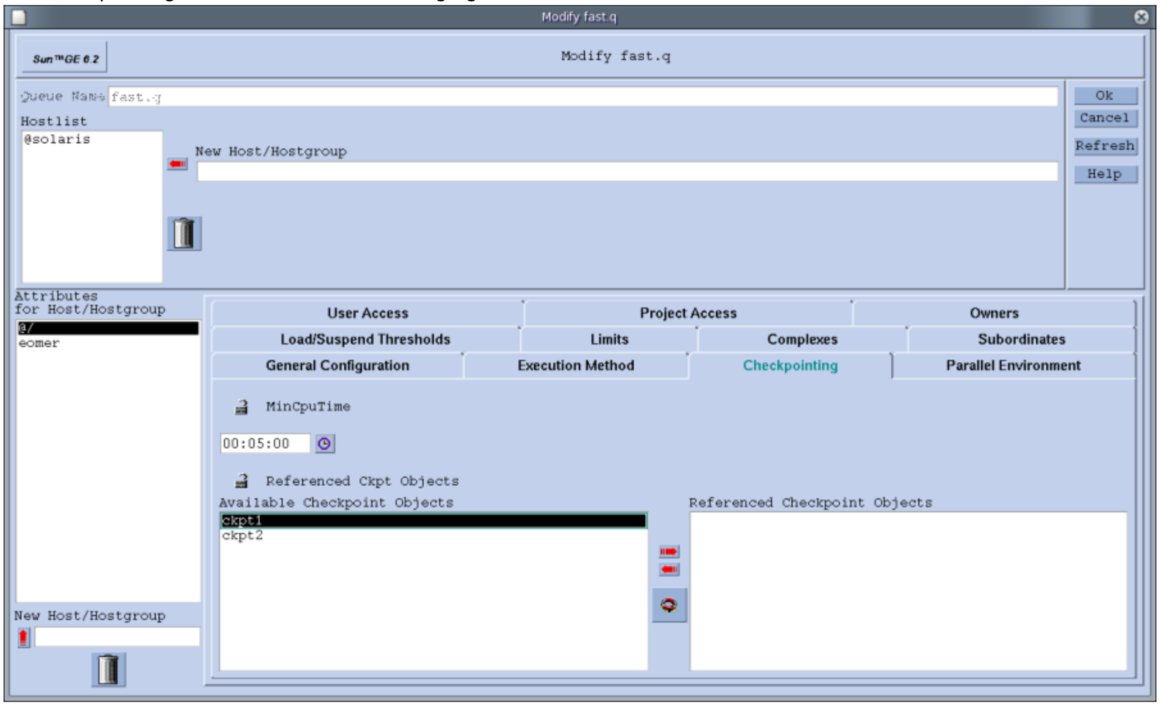
For more information, see the `queue_conf(5)` man page.

How to Configure the Checkpointing Parameters With QMON

1. On the QMON Main Control window, click the Queue Control button.
   The Cluster Queues dialog box appears.

2. Select a queue and then click Modify.
   The Queue Configuration dialog box appears.

3. To configure checkpointing parameters, click the Checkpointing tab.
   The Checkpointing tab is shown in the following figure.



Modify fast.q

Queue Name fast.q

Hostlist
@solaris

New Host/Hostgroup

Attributes for Host/Hostgroup
@/
eomer

New Host/Hostgroup

MinCpuTime
00:05:00

Referenced Ckpt Objects

Available Checkpoint Objects
ckpt1
ckpt2

Referenced Checkpoint Objects

4. From the Checkpointing tab, you can choose from the following tasks:
   - To modify the MinCpuTime, click on the clock icon and then adjust the time.
   - To reference a checkpointing environment from the queue, select the name of a checkpointing environment from the Available list, and then click the right arrow to add it to the Referenced list.
   - To remove a checkpointing environment from the Referenced list, select it, and then click the left arrow.
   - To add or modify checkpointing environments, click the button below the red arrows to open the Checkpointing Configuration dialog box.

5. Click Ok to save your changes.
   Click Cancel to leave the dialog box without saving your changes.

For more information, see Configuring Checkpointing Environments With QMON.

See the `queue_conf(5)` man page for detailed information about these parameters.

How to Configure Load and Suspend Thresholds From the Command Line

To modify a cluster queue, type the following command:

```
qconf -mq <cluster_queue>
```

The -mq option (modify cluster queue) modifies the specified cluster queue. The -mq option displays an editor containing the configuration of the cluster queue to be changed. The editor is either the default vi editor or an editor defined by the EDITOR environment variable. Modify the cluster queue by changing the configuration and then saving your changes.
You can configure the following load and suspend parameters from within the editor:

- `load_thresholds` – A list of load thresholds. If one of the thresholds is exceeded, no additional jobs will be scheduled to the queues and the master daemon will signal an overload condition for this node.
- `suspend_thresholds` – A list of load thresholds with the same semantics as that of the `load_thresholds` parameter (see above) except that exceeding one of the denoted thresholds initiates suspension of one of multiple jobs in the queue.
- `nsuspend` – The number of jobs which are suspended/enabled per time interval if at least one of the load thresholds in the `suspend_thresholds` list is exceeded or if no `suspend_threshold` is violated anymore respectively.
- `suspend_interval` – The time interval in which further `nsuspend` jobs are suspended if one of the `suspend_thresholds` (see above for both) is exceeded by the current load on the host on which the queue is located.

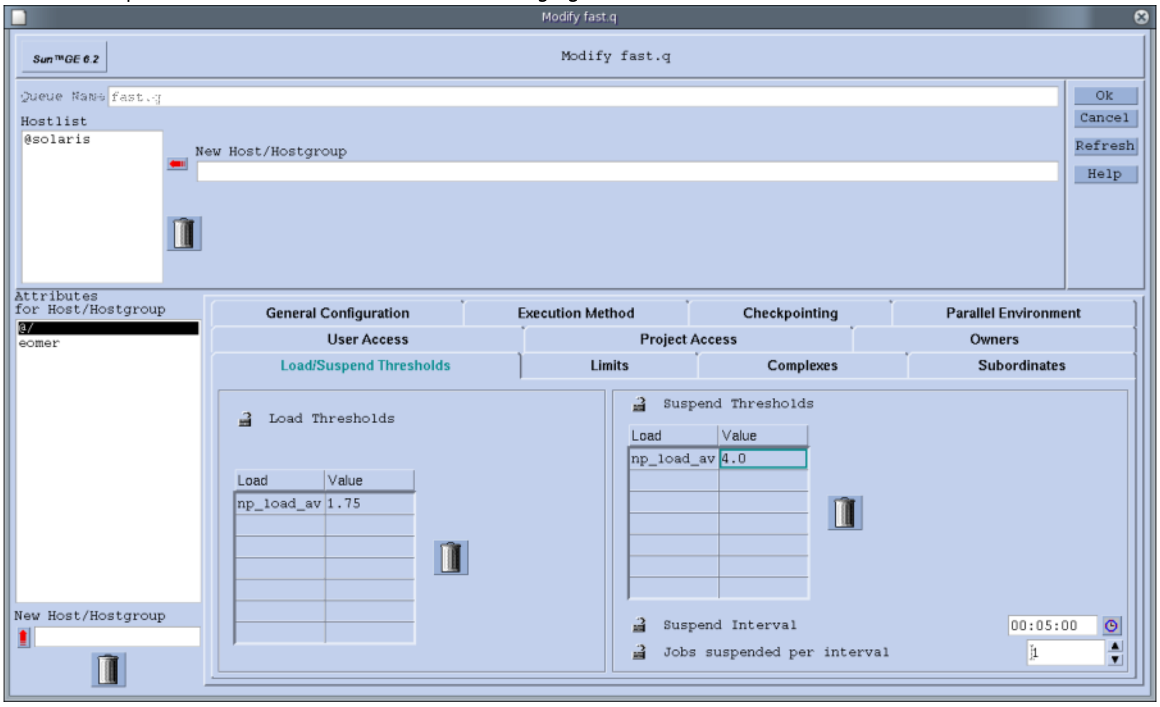
For more information, see the `queue_conf(5)` man page.

How to Configure Load and Suspend Thresholds With QMON

1. On the QMON Main Control window, click the Queue Control button.
   The Cluster Queues dialog box appears.

2. Select a queue and then click Modify.
   The Queue Configuration dialog box appears.

3. To configure load and suspend thresholds, click the Load/Suspend Thresholds tab.
   The Load/Suspend Thresholds tab is shown in the following figure.



Modify fast.q

Sun™GE 6.2 Modify fast.q

Queue Name fast.q

Hostlist
@solaris

New Host/Hostgroup

Attributes for Host/Hostgroup
@/
eomer

General Configuration | Execution Method | Checkpointing | Parallel Environment | User Access | Project Access | Owners | Load/Suspend Thresholds | Limits | Complexes | Subordinates

Load Thresholds

Load	Value
np_load_av	1.75

Suspend Thresholds

Load	Value
np_load_av	4.0

Suspend Interval 00:05:00

Jobs suspended per interval 1

   - The Load Thresholds and the Suspend Thresholds tables display the defined overload thresholds for load parameters and consumable resource attributes. See Configuring Resource Attributes for more information. In the case of load thresholds, overload prevents the queue from receiving further jobs. In the case of suspend thresholds, overload suspends jobs in the queue to reduce the load.
   - Suspend interval. The time interval between suspension of other jobs in case the suspend thresholds are still exceeded.
   - Jobs suspended per interval. The number of jobs to suspend per time interval in order to reduce the load on the system that is hosting the configured queue.
     See the `queue_conf(5)` man page for detailed information about these parameters.

4. From the Load/Suspend thresholds tab, you can perform the following tasks:
   - To change an existing threshold, select it, and then double-click the corresponding Value field.
   - To add new thresholds, do the following:

a. Click Load or Value.
   A selection list appears with all valid attributes that are attached to the queue.
b. To add an attribute to the Load column of the corresponding threshold table, select an attribute, and then click OK.

- To delete an existing threshold, select it, and then type Ctrl-D or click mouse button 3.
  You are prompted to confirm that you want to delete the selection.

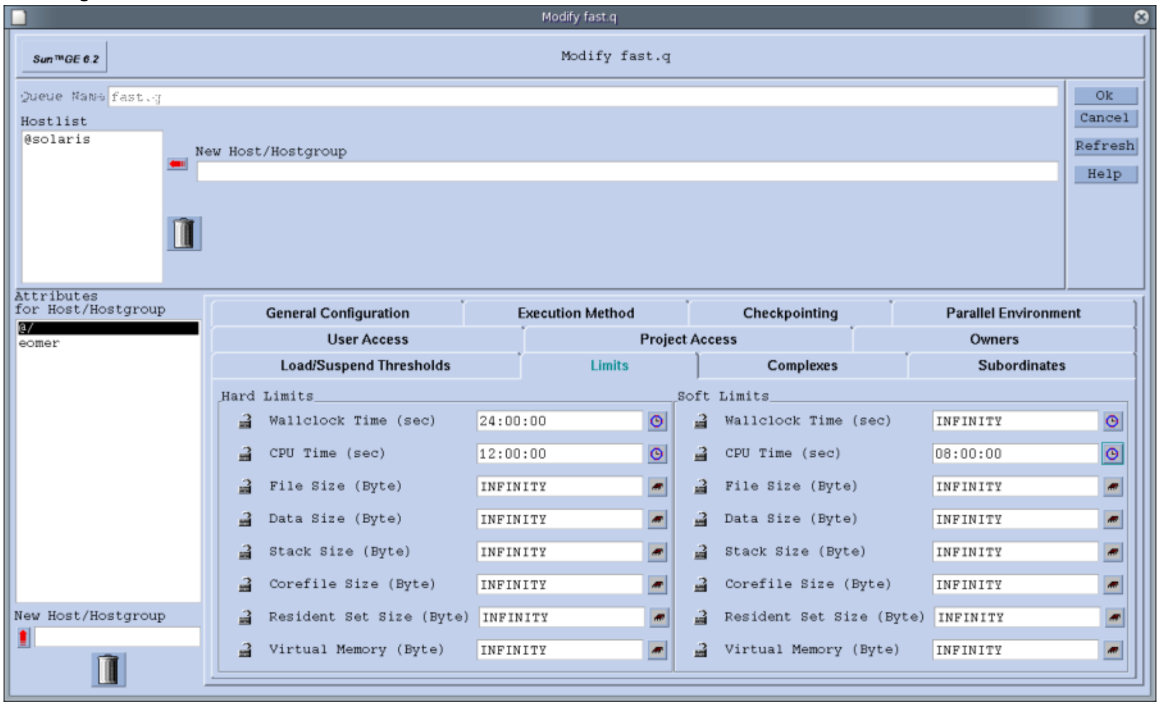
5. Click Ok to save your changes.
   Click Cancel to leave the dialog box without saving your changes.

How to Configure Resource Limits With QMON

1. On the QMON Main Control window, click the Queue Control button.
   The Cluster Queues dialog box appears.

2. Select a queue and then click Modify.
   The Queue Configuration dialog box appears.

3. To configure hard or soft limits, click the Limits tab.



Modify fast.q

Sun™GE 6.2 — Modify fast.q

Queue Name: fast.q

Hostlist: @solaris

New Host/Hostgroup

Attributes for Host/Hostgroup: @/, eomer

Hard Limits		Soft Limits	
Wallclock Time (sec)	24:00:00	Wallclock Time (sec)	INFINITY
CPU Time (sec)	12:00:00	CPU Time (sec)	08:00:00
File Size (Byte)	INFINITY	File Size (Byte)	INFINITY
Data Size (Byte)	INFINITY	Data Size (Byte)	INFINITY
Stack Size (Byte)	INFINITY	Stack Size (Byte)	INFINITY
Corefile Size (Byte)	INFINITY	Corefile Size (Byte)	INFINITY
Resident Set Size (Byte)	INFINITY	Resident Set Size (Byte)	INFINITY
Virtual Memory (Byte)	INFINITY	Virtual Memory (Byte)	INFINITY

4. To change a value of a limit, click the button at the right of the field whose value you want to change.
   A dialog box appears where you can type either Memory or Time limit values.
   When a hard limit is exceeded, the running job in the queue is stopped immediately. When a soft limit is exceeded, a signal is sent that the job can intercept before the job is stopped. The Limits tab is shown in the following figure.

5. Click Ok to save your changes.
   Click Cancel to leave the dialog box without saving your changes.

See the `queue_conf(5)` and the `setrlimit(2)` man pages for detailed information about limit parameters and their interpretation for different operating system architectures.

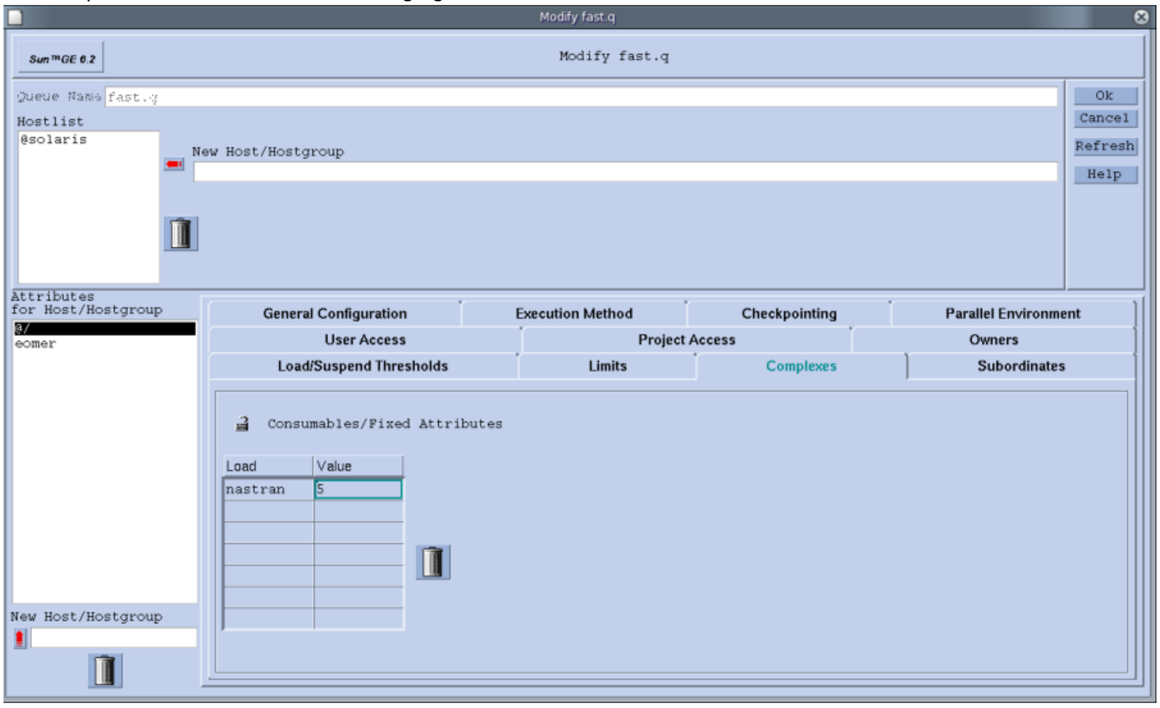
How to Configure Complex Resource Attributes With QMON

1. On the QMON Main Control window, click the Queue Control button.

The Cluster Queues dialog box appears.

2. Select a queue and then click Modify.
   The Queue Configuration dialog box appears.

3. To configure complex resource attributes, click the Complexes tab.
   The Complexes tab is shown in the following figure.



Load	Value
nastran	5

The attributes for which values are explicitly defined are displayed in the Consumable/Fixed Attributes table. The available resource attributes are assembled by default from the complex.
Resource attributes are either consumable or fixed. The definition of a consumable value defines a capacity managed by the queue. The definition of a fixed value defines a queue-specific value. See Configuring Resource Attributes for further details.

4. From the Complexes tab, choose from the following tasks:
   - To modify an attribute, select it, and then double-click the corresponding Value field.
   - To add new attribute definitions, click Load or Value.
     The Attribute Selection dialog box appears with a list of all valid attributes that are attached to the queue.
   - To add an attribute to the Load column of the attribute table, select it, and then click OK.
   - To delete an attribute, select it, and then press Ctrl-D or click mouse button 3. You are prompted to confirm that you want to delete the attribute.

See the `queue_conf`(5) page for detailed information about these attributes.

Next Steps

Use the Complex Configuration dialog box to check or modify the current complex configuration before you attach user-defined resource attributes to a queue or before you detach them from a queue. To access the Complex Configuration dialog box, click the Complex Configuration button on the QMON Main Control window. For more information, see See Configuring Resource Attributes.

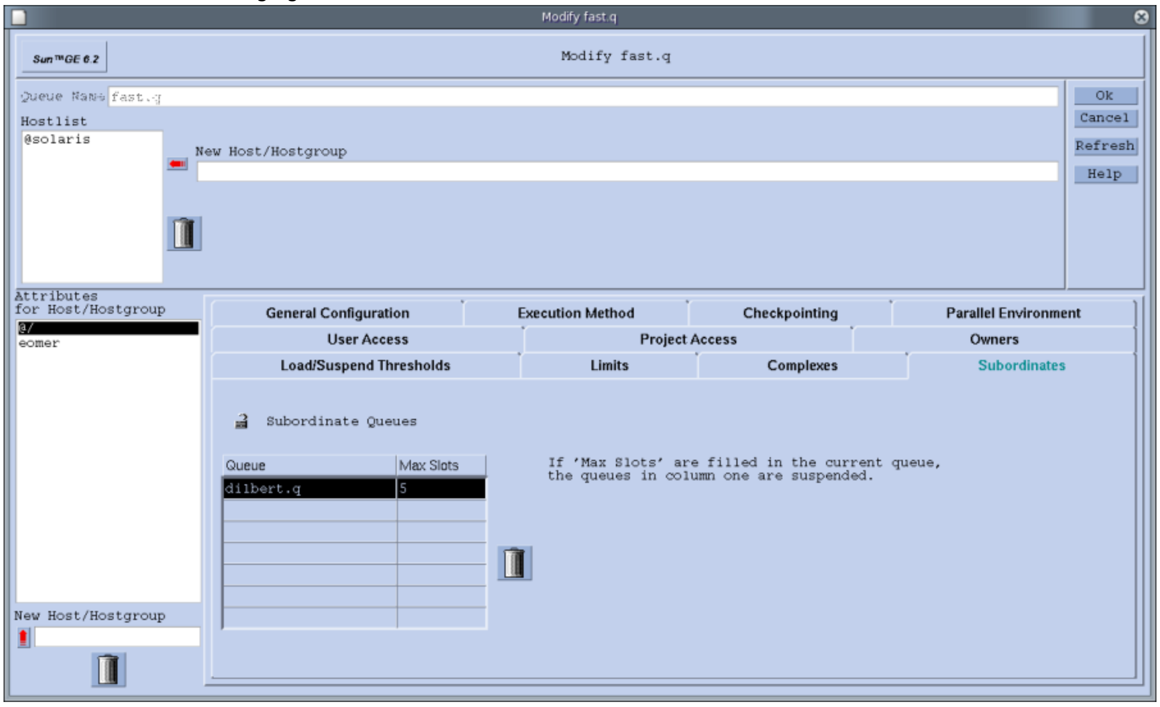
How to Configure Subordinate Queues With QMON

1. On the QMON Main Control window, click the Queue Control button.
   The Cluster Queues dialog box appears.

2. Select a queue and then click Modify.
   The Queue Configuration dialog box appears.

3. To configure subordinate queues, click the Subordinates tab.

Use the subordinate queue facility to implement high priority and low priority queues as well as standalone queues. The Subordinates tab is shown in the following figure.



4. From the Subordinate Queues tab, you can specify the following parameters:
   - Queue – A list of the queues that are subordinated to the configured queue.
     Subordinated queue instances on a host are suspended if the configured queue instance on the same host becomes busy. Subordinated queue instances on that host are resumed when the configured queue instance is no longer busy.
   - Max Slots – For any subordinated queue, you can configure the number of job slots that must be filled in the configured queue to trigger a suspension. If no maximum slot value is specified, all job slots must be filled to trigger suspension of the corresponding queue.

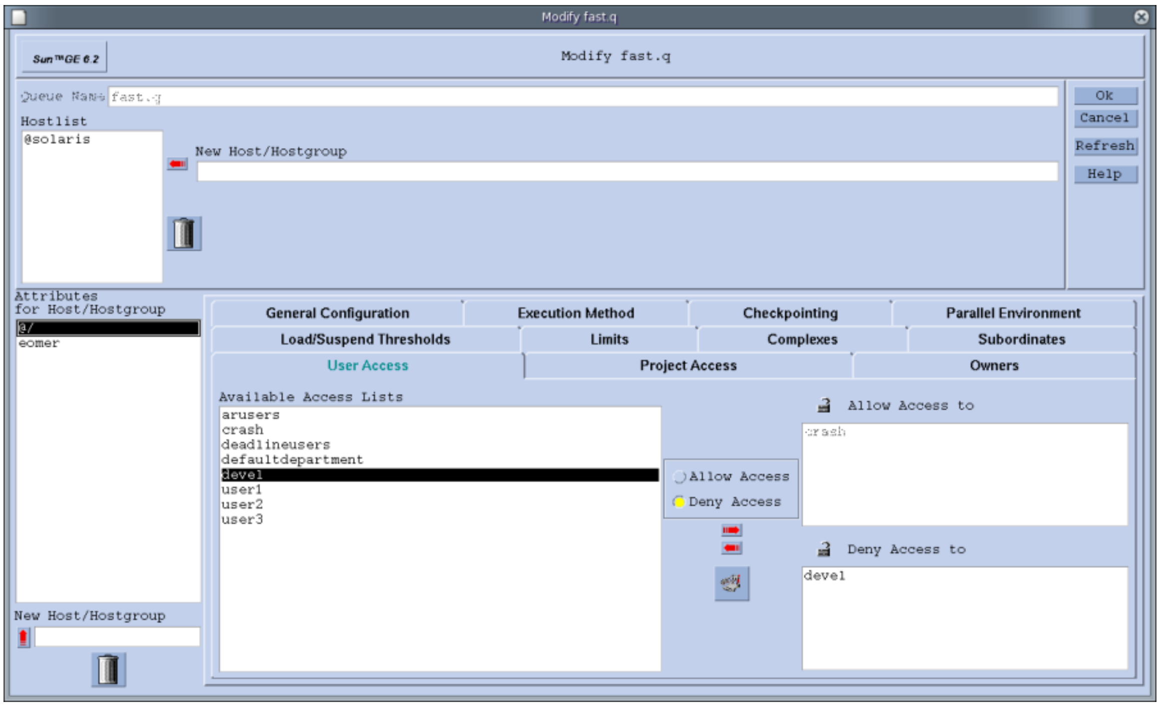
See the `queue_conf(5)` man page for detailed information about these parameters.

How to Configure User Access Parameters With QMON

1. On the QMON Main Control window, click the Queue Control button.
   The Cluster Queues dialog box appears.

2. Select a queue and then click Modify.
   The Queue Configuration dialog box appears.

3. To configure user access parameters, click the User Access tab.
   The User Access tab is shown in the following figure.



Queue Name fast.q

Hostlist
@solaris

Attributes for Host/Hostgroup
@/
eomer

Available Access Lists
arusers
crash
deadlineusers
defaultdepartment
devel
user1
user2
user3

Allow Access to
crash

Deny Access to
devel

Users or user groups belonging to access lists that are included in the Allow Access list have access to the queue. Users who are included in the Deny Access list cannot access the queue. If the Allow Access list is empty, access is unrestricted unless explicitly stated otherwise in the Deny Access list.

4. From the User Access tab, you choose from the following tasks:
   - To add user access lists, do the following:
     a. Click the button below the red arrows.
        The User Configuration dialog box appears.
     b. Click Add.
     c. Click Ok to save your changes.
        Click Cancel to leave the dialog box without saving your changes.
   - To modify user access lists, do the following
     a. Click the button below the red arrows.
        The User Configuration dialog box appears.
     b. Select a user access list and click Modify.
     c. Click Ok to save your changes.
        Click Cancel to leave the dialog box without saving your changes.

For more information about the User Configuration dialog box, see How to Configure User Access Lists With QMON.

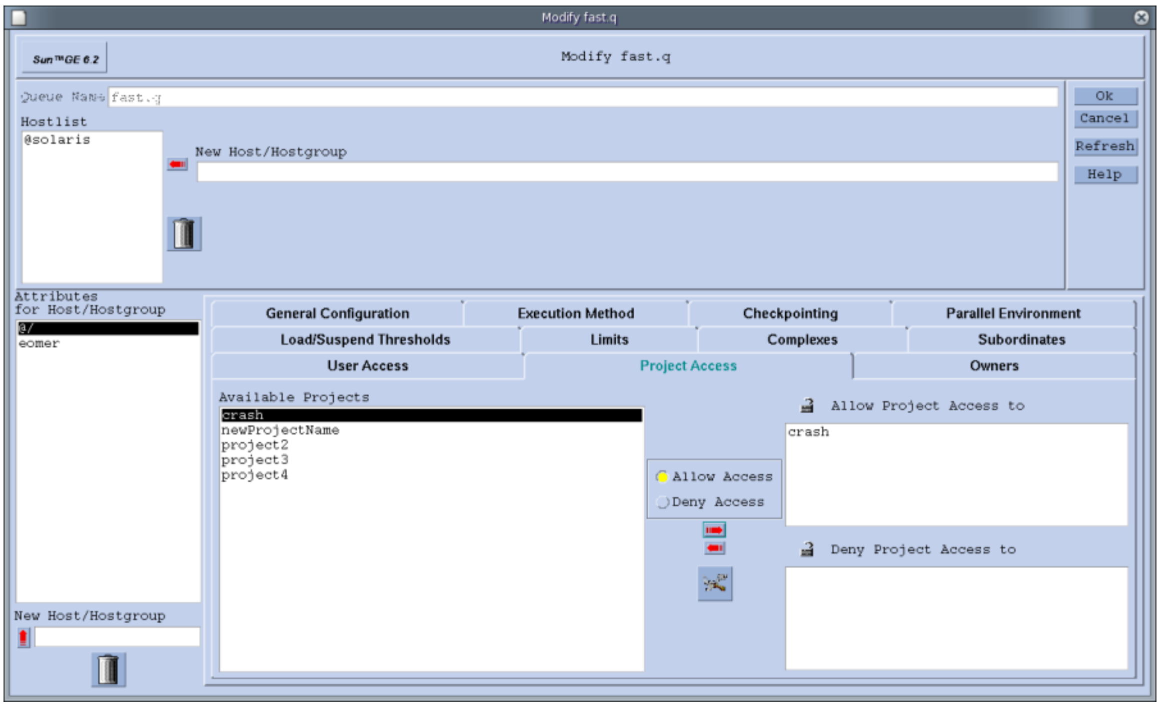
See the `queue_conf(5)` man page for detailed information about these parameters.

How to Configure Project Access Parameters With QMON

1. On the QMON Main Control window, click the Queue Control button.
   The Cluster Queues dialog box appears.

2. Select a queue and then click Modify.
   The Queue Configuration dialog box appears.

3. To configure project access parameters, click the Project Access tab.
   The Project Access tab is shown in the following figure.



Modify fast.q

Queue Name fast.q

Hostlist
@solaris

Attributes for Host/Hostgroup
@/
eomer

Available Projects
crash
newProjectName
project2
project3
project4

[✓ Allow Access] Deny Access

Allow Project Access to
crash

Deny Project Access to

Jobs that are submitted to a project that belongs to the list of allowed projects have access to the queue. Jobs that are submitted to denied projects are not dispatched to the queue.

4. *From the Project Access Tab, choose from the following tasks:
   - To add project access, do the following:
     a. Click the button below the red arrows.
        The Project Configuration dialog box appears.
     b. Click Add.
     c. Click Ok to save any changes.
        Click Cancel to leave the dialog box without saving your changes.
   - To modify project access, do the following:
     a. Click the button below the red arrows.
        The Project Configuration dialog box appears.
     b. Select a queue and click modify.
     c. Click Ok to save your changes.
        Click Cancel to leave the dialog box without saving your changes.

For more information, see How to Configure Projects With QMON.

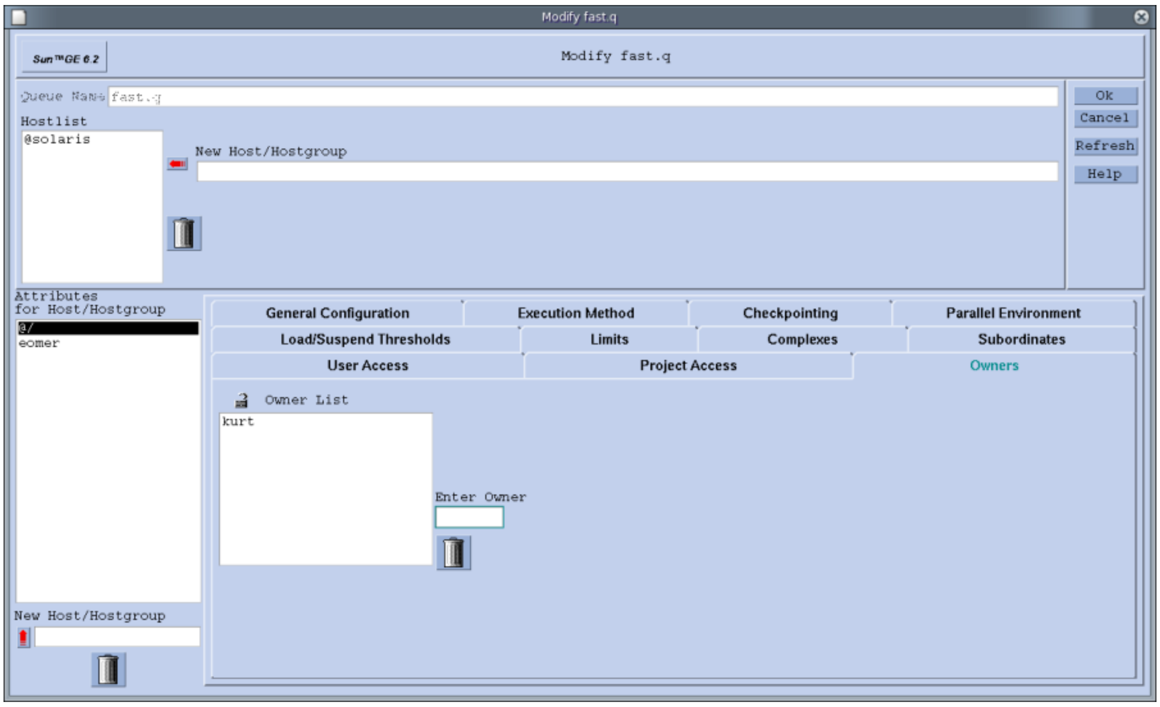
See the `queue_conf(5)` man page for detailed information about these parameters.

How to Configure Owners Parameters With QMON

1. On the QMON Main Control window, click the Queue Control button.
   The Cluster Queues dialog box appears.

2. Select a queue and then click Modify.
   The Queue Configuration dialog box appears.

3. To configure owners parameters, click the Owners tab.
   The Owners tab is shown in the following figure.



Queue Name fast.q

Hostlist
@solaris

@/
eomer

Owner List
kurt

4. From the Owners tab, you can choose from the following tasks:
   - To add any user account to the owner list, enter a username into the Enter Owner field and click Return.
     You can add any user account to the owner list. Typically, you assign users to be owners of certain queue instances to enable them to suspend or disable jobs when necessary. For example, users might occasionally need certain machines for important work, and those machines might be strongly affected by jobs that are running in the background.
   - To delete a user account from the queue owner list, select it, and then click the trash can icon.

Queue owners can do the following:

- Suspend – Stop execution of all jobs running in the queue and close the queue.

> Note
> Jobs that are suspended explicitly while a queue is suspended are not resumed when the queue is resumed. Explicitly suspended jobs must be resumed explicitly.

- Resume – Unsuspend the queue, and then open it.
- Disable – Close the queue, but do not affect running jobs.
- Enable – Open the queue.

See the `queue_conf`(5) man page for detailed information about these parameters.

Generating Accounting Statistics

An accounting record is written to the Sun Grid Engine accounting file (`$SGE_ROOT/$SGE_CELL/common/accounting`) for each completed job. You can use `qacct` to generate the following accounting statistics:

- To display aggregate resource usage information for all machines in the cluster, type the following command:

```
qacct
```

  Statistics are generated by all jobs that have completed and that are contained in the in the cluster accounting file `$SGE_ROOT/$SGE_CELL/common/accounting`. In this case, `qacct` reports on the following:

  - `WALLCLOCK` – Wall clock time, which is the time between when the job starts and when it finishes
  - `UTIME` – CPU time spent in user processes

    - `STIME` – CPU time spent in system calls
    - `CPU` – CPU time usage in seconds
    - `MEMORY` – The integral memory usage in Gbytes CPU seconds
    - `IO` – The amount of data transferred in input/output operations
    - `IOW` – The input/output wait time in seconds
- To display all resource usage information for a completed job or completed jobs, type the following command:

  ```
  qacct -j \[<job_id>\|<job_name>\|<pattern>\]
  ```

  If no argument is given, all jobs contained in the referenced accounting file are displayed. If a job ID is specified, and if more than one entry is displayed, one of the following is true:

    - Job ID numbers have wrapped around. The range for job IDs is 1 through 999999.
    - If a checkpointing job migrated, then it is displayed.
- To display resource usage information for all completed jobs that match a resource requirement specification, type the following command:

  ```
  qacct -l <attr=val,...>
  ```

For more information, see the `qacct(1)` man page. For more information on the information found in the accounting files, see the `accounting(5)` man page.

Managing Advance Reservations

Managers, operators, and users who are referenced in the `arusers` access list can use the advance reservations feature to reserve specific consumable resources in the cluster for future use. If granted, an advance reservation causes the requested resources to be reserved for the specified user, administrator, or job.

You might better understand the concept of an advance reservation if you think about a travel reservation system. Using the Sun Grid Engine resource reservation capability, all passengers are guaranteed to get on a plane flight in the order in which the passengers arrive at the airport. What you really want is to be able to reserve your flights in advance so that you can arrange your specific flight schedule before you arrive at the airport. Grid Engine enables you to make those arrangements in advance based on an allocation scheme that the scheduler uses.

After an advance reservation is submitted, the Grid Engine scheduler selects the best suited queues according to the request. All queues that are not in an orphaned state are considered available. If those resources are available, then the reservation is granted and assigned an ID.

When the reservation is first requested, the requesting user can include a list of users and groups who are also allowed to user the reservation.

Once a user has been granted an advance reservation, the requesting user or users designated on the access list can do the following:

- Submit jobs to the given reservation. If the reservation is not yet active, the job will remain pending until the reservation's start time.
- Block off resources for an out-of-band purpose, such as maintenance.

For more details on managing advance reservations, see the `qrsub(1)` man page.

Advance Reservations Tasks

Task	User Interface	Description
How to Enable a User to Create Advance Reservations	CLI or QMON	Learn how to add users to the `arusers` access list so that they can use advance reservations.
How to Configure Advance Reservations	CLI or QMON	Learn how to add, show, and delete advance reservations.

Additional Resources

Topic	Description
ARCo Queries for Advance Reservations	Learn which ARCo queries apply specifically to advance reservations.

How to Enable a User to Create Advance Reservations From the Command Line

The ability to create an advance reservation is limited to members of the `arusers` list. The `arusers` list is created during Grid Engine installation.

Only the Grid Engine administrator and those people explicitly included in the list can create advance reservations.

To add a specific user to the access list from the command line, type a command similar to the following example:

```
# qconf -au <username> arusers
```

You should see a confirmation message similar to the following:

```
added "username-to-add" to access list "arusers"
```

Tip
You can also perform this task from the QMON graphical interface. For information, see How to Configure User Access Lists with QMON.

How to Enable a User to Create Advance Reservations With QMON

The ability to create an advance reservation is limited to manager, operators, and members of the `arusers` access list. The `arusers` access list is created during Grid Engine installation.

Only the Grid Engine administrator and those people explicitly included in the list can create advance reservations.

To enable a user to create advance reservations, do the following:

1. On the QMON main control window, click the User Configuration icon.
   The User Configuration dialog box appears.

2. Click the Userset tab.

3. Select the **`arusers`** access list and click Modify.
   The Access List Definition dialog box appears.

4. You can add a new access list definition, a new user, or a new group, by using one of the following tasks:
   - To add a new access list definition, type the name of the access list in the Userset Name field.
   - To add a new user or group (with an @ prefix) to the access list, type a user or group name in the User/Group field.

5. Click Ok to save.
   Click Cancel to close the dialog box without saving your changes.

How to Configure Advance Reservations From the Command Line

This page provides information on the following:

- Creating Advance Reservations
- Monitoring Advance Reservations
- Deleting Advance Reservations

Note
To request advance reservations, you must be a manager, an operator, or a member of the `arusers` access list. For more information, see How to Enable a User to Create Advance Reservations From the Command Line.

Tip
An advance reservation must have a start time and an end time. If you do not provide a start time, the advance reservation starts as soon as you submit the definition.

Creating Advance Reservations

Use the `qrsub` command to create an advance reservation and submit it to the Sun Grid Engine queuing system.

You may define default request files (analogous to `sge_request` for qsub) that can contain any of the possible command line options. The file names are `$SGE_ROOT/$SGE_CELL/common/sge_ar_request` (global defaults file) and `$HOME/.sge_ar_request` (user private defaults file).

To define the time at which the advance reservation will start and end, use the following arguments for the `qrsub` command:

```
qrsub -a <date_time> -e <date_time>
```

The start and end times are in [[CC]YY]MMDDhhmm[.SS] format.

To define the time at which the advance reservation will start and end, using a duration, use the following arguments for the `qrsub` command:

```
qrsub -a <date_time> -d <time>
```

The start time is in [[CC]YY]MMDDhhmm[.SS] format. The duration is in hhmm[.SS] format.

For information on additional configuration parameters, see the `qrsub(1)` man page. Many of the options for `qrsub` are the same as those for `qsub`. For more information, see the `submit(1)` man page.

`qrsub` Examples

The following example reserves an slot in the queue `all.q` on `host1` or `host2` or `host3`.

- With new upcoming RESTRING syntax

```
qrsub -q all.q -l "h=host1|host2|host3" -u $user -a 01121200 -d 1:0:0
```

- With current syntax

```
qrsub -q "*@host1,*@host2,*@host3" -u $user -a 01121200 -d 1:0:0
```

The following example reserves 4 slots on a host with `arch=sol-sparc64`.

```
qrsub -pe alloc_pe_slots 4 -l a=sol-sparc64 -u $user -a 01121200 -d 1:0:0
```

Monitoring Advance Reservations

Use the `qrstat` command to view the current status of the granted Sun Grid Engine advance reservations. You can get information about specific advance reservations or users. Without any options, `qrstat` displays an overview of all reservations.

To show a list of all currently-configured advance reservations, type the following:

```
qrstat
```

To show the configuration for a specific advance reservation, type the following:

```
qrstat -ar <ar_id>
```

`qrstat` Examples

The following example shows information about all advance reservations:

```
% qrstat
AR-ID   name       owner        state start at            end at              duration
----------------------------------------------------------------------------------------
    192 project_xy user1        r     12/14/2006 14:47:23 12/14/2006 14:57:33 0:10:10
    193            user2        w     12/18/2006 10:00:00 12/19/2006 10:00:10 24:0:10
```

The following example shows detailed information about the advance reservation whose ar_id is 193:

```
% qrstat -ar 193
==============================================================
id:                          193
ar_name:
submission_time:             Mon Nov 27 17:11:34 2006
owner:                       user1
acl_list:                    user1,user2
start_time:                  Mon Dec 18 10:00:00 2006
end_time:                    Tue Dec 19 10:00:10 2006
duration:                    24:0:10
granted_slots:               all.q@host1=2,all.q@host2=1
resource_list:               myapp=2,myapp=1
...
```

For more information, see the `qrstat(1)` man page.

Deleting Advance Reservations

Use the `qrdel` command to delete an advance reservation. The `qrdel` command requires at least one advance reservation identifier, which can be either an ar_id or an ar_name. The `qrdel` command deletes advance reservations in the order in which their identifiers are presented.

Jobs referring to a advance reservation that is tagged for deletion will also be removed. After all jobs that refer to an advance reservation are removed from the Sun Grid Engine database, the reservation will be removed.

To delete an advance reservation, type the following:

```
qrdel <ar_id>
```

`qrdel` Example

The following example deletes the advance reservation 193:

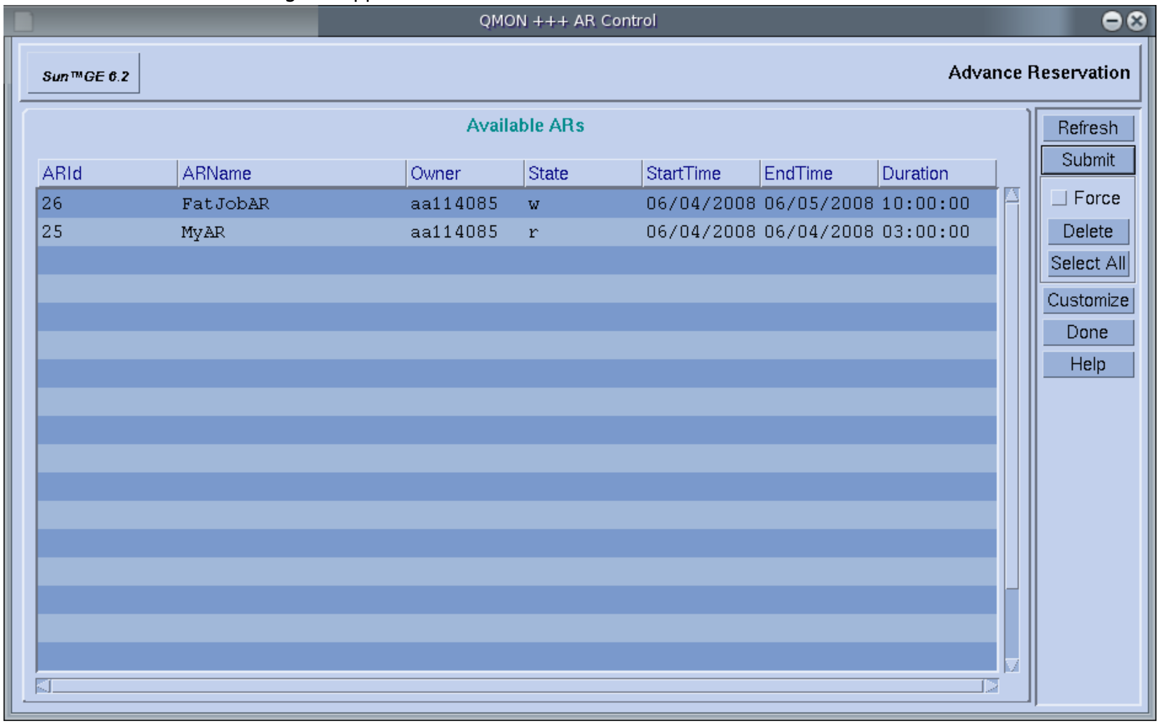
```
qrdel 193
```

For more information, see the `qrdel(1)` man page.

How to Configure Advance Reservations With QMON

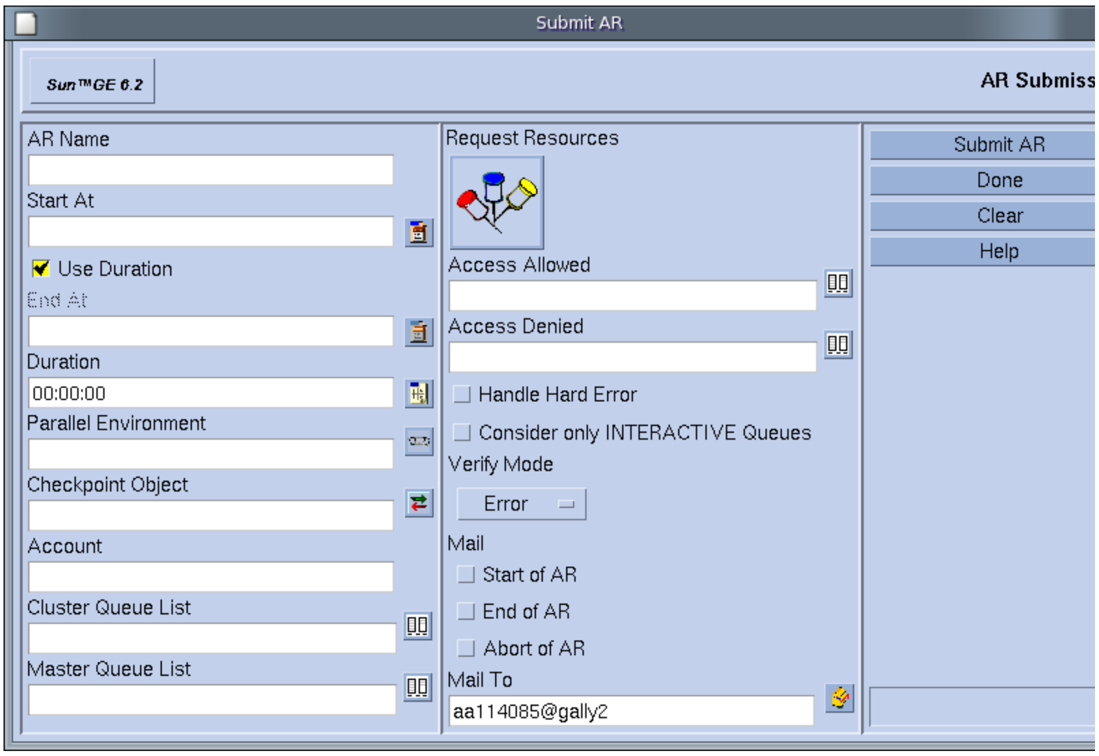
Note
To request advance reservations, you must be a manager, an operator, or a member of the `arusers` access list. For more information, see How to Enable a User to Create Advance Reservations With QMON.

1. On the QMON main control window, click the Advance Reservation icon.
   The Advance Reservation dialog box appears.



ARId	ARName	Owner	State	StartTime	EndTime	Duration
26	FatJobAR	aa114085	w	06/04/2008	06/05/2008	10:00:00
25	MyAR	aa114085	r	06/04/2008	06/04/2008	03:00:00

2. From the Advance Reservation dialog box, you can view all currently configured advance reservations and choose from the following tasks:
   - To add an advance reservation, do the following:
     a. Click the Submit button.
        The AR Definition window appears, as shown in this example.



AR Name:
Start At:
[✓ Use Duration]
End At:
Duration: 00:00:00
Parallel Environment:
Checkpoint Object:
Account:
Cluster Queue List:
Master Queue List:
Request Resources
Access Allowed:
Access Denied:
☐ Handle Hard Error
☐ Consider only INTERACTIVE Queues
Verify Mode: Error
Mail
☐ Start of AR
☐ End of AR
☐ Abort of AR
Mail To: aa114085@gally2

b. Define the time at which the advance reservation will start.

> Tip
> If you do not provide a start time, the advance reservation starts as soon as you submit the definition.

c. Define the time at which the advance reservation will end or provide a duration.

> Note
> A duration or end time is the only required parameter for an advance reservation.

d. (Optional) Provide additional parameters as desired.
   For more information about the possible parameters, click the Help button in the QMON window or see the `qrsub(1)` man page.
e. When you are satisfied with the parameters for your advance reservation, click the Submit AR button.
   If you see an error similar to the following, confirm that the user who is running QMON is in the `arusers` access list.

- To delete an advance reservation, do the following:
  a. Select one or more advance reservations.
  b. Click Delete.

The page How to Manage Advance Reservations From the Command Line does not exist.

ARCo Queries for Advance Reservations

The Accounting and Reporting Console (ARCo) provides several queries that specifically apply to advance reservations:

- Accounting per AR
- Advanced Reservation Attributes
- Advanced Reservation Log
- Advanced Reservation Time Usage
- Advanced Reservation by User
- Number of Jobs Completed per AR

For more information about ARCo, see Accounting and Reporting Console.

Managing Checkpointing Environments

Checkpointing Overview

Checkpointing is a facility that does the following tasks:

- Freezes the status of an running job or application
- Saves this status (the checkpoint) to disk
- Restarts the job or application from the checkpoint if the job or application has otherwise not finished, for example, due to a system shutdown

If you move a checkpoint from one host to another host, checkpointing can migrate jobs or applications in a cluster without significant loss of resources. Hence, dynamic load balancing can be provided with the help of a checkpointing facility.

The Grid Engine system supports two levels of checkpointing:

- User-level checkpointing – At this level, providing the checkpoint generation mechanism is entirely the responsibility of the user or the application. Examples of user-level checkpointing include the following:
  - The periodic writing of restart files that are encoded in the application at prominent algorithmic steps, combined with proper processing of these files when the application is restarted
  - The use of a checkpoint library that must be linked to the application and that thereby installs a checkpointing mechanism

  > Note
  > A variety of third-party applications provides an integrated checkpoint facility that is based on the writing of restart files. Checkpoint libraries are available from hardware vendors or from the public domain. Refer to the Condor project of the University of Wisconsin, for example.

- Kernel-level transparent checkpointing – This level of checkpointing must be provided by the operating system, or by enhancements to it, that can be applied to any job. No source code changes or relinking of your application need to be provided to use kernel-level checkpointing.
  Kernel-level checkpointing can be applied to complete jobs, that is, the process hierarchy created by a job. By contrast, user-level checkpointing is usually restricted to single programs. Therefore the job in which such programs are embedded needs to properly handle cases where the entire job gets restarted.
  Kernel-level checkpointing, as well as checkpointing based on checkpointing libraries, can consume many resources. The complete virtual address space that is in use by the job or application at the time of the checkpoint must be dumped to disk. By contrast, user-level checkpointing based on restart files can restrict the data that is written to the checkpoint on the important information only.
- Transparent Checkpointing – This method of checkpointing uses a checkpointing library, such as the one provided by the public domain package, Condor.

Checkpointing Environments Overview

The Grid Engine software provides a configurable attribute description for each checkpointing method used. Different attribute descriptions reflect the different checkpointing methods and the potential variety of derivatives from these methods on different operating system architectures.

This attribute description is called a checkpointing environment. Default checkpointing environments are provided with the distribution of the Grid Engine system and can be modified according to the site's needs.

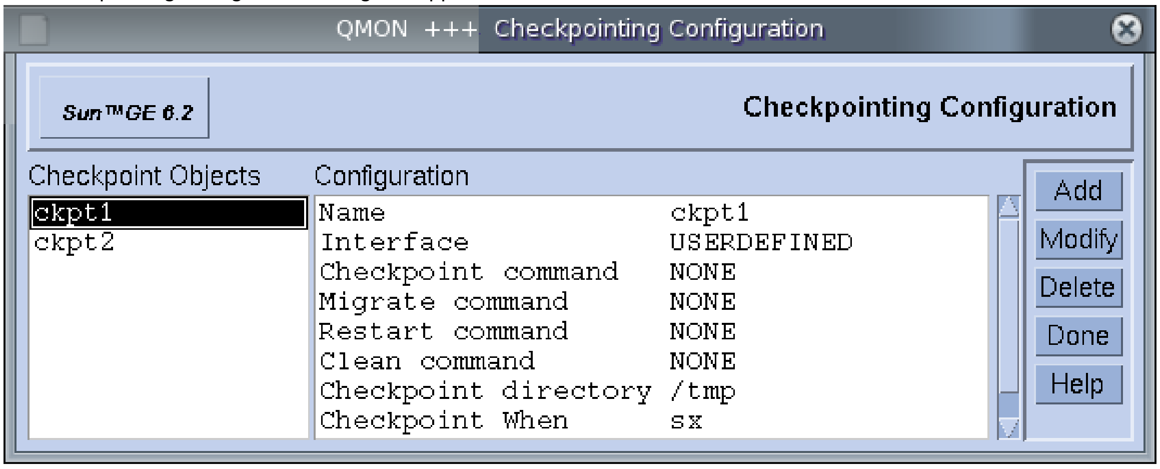
New checkpointing methods can be integrated in principal. However, the integration of new methods can be a challenging task. This integration should be performed only by experienced personnel or by your Grid Engine system support team.

Checkpointing Tasks

Task	User Interface	Description
How to Configure Checkpointing Environments	CLI or QMON	Learn how to display, add, modify, and delete checkpointing environments.

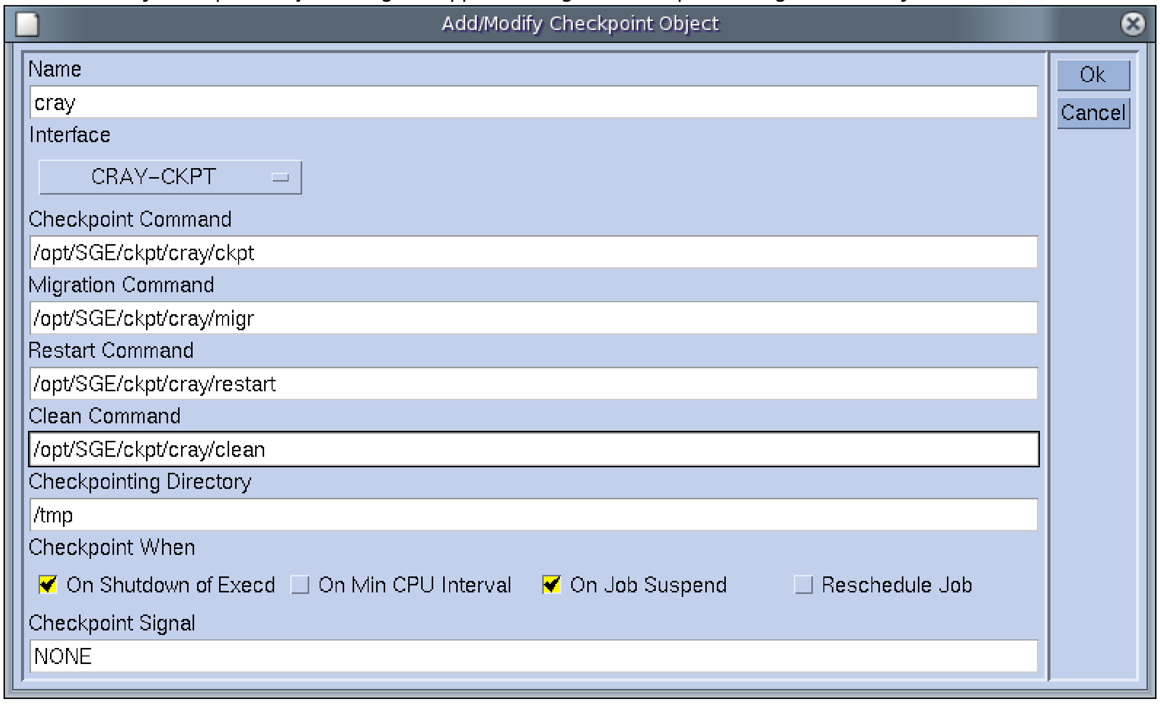
Unknown macro: {excerpt-minclude}

How to Configure Checkpointing Environments With QMON

1. On the QMON Main Control window, click the Checkpoint Configuration button.
   The Checkpointing Configuration dialog box appears.
   

   ```
   ckpt1
   ckpt2
   ```

   ```
   Name                 ckpt1
   Interface            USERDEFINED
   Checkpoint command   NONE
   Migrate command      NONE
   Restart command      NONE
   Clean command        NONE
   Checkpoint directory /tmp
   Checkpoint When      sx
   ```

2. From the Checkpointing Configuration dialog box, you can perform the following functions:
   - To display configured checkpointing environments, select one of the checkpointing environment names listed under Checkpoint Objects.
     The corresponding configuration is displayed under Configuration.
   - To delete a configured checkpointing environment, select it, and then click Delete.
   - To add a checkpointing environment, click Add.
     The Add/Modify Checkpoint Object dialog box appears, along with a template configuration that you can edit.
     

     Name: cray
     Interface: CRAY-CKPT
     Checkpoint Command: /opt/SGE/ckpt/cray/ckpt
     Migration Command: /opt/SGE/ckpt/cray/migr
     Restart Command: /opt/SGE/ckpt/cray/restart
     Clean Command: /opt/SGE/ckpt/cray/clean
     Checkpointing Directory: /tmp
     Checkpoint When: [✓ On Shutdown of Execd] On Min CPU Interval [✓ On Job Suspend] Reschedule Job
     Checkpoint Signal: NONE

     Fill out the template with the requested information.
   - To modify checkpointing environments, select the name of the configured checkpointing environment you want to modify, and then click Modify.
     The Add/Modify Checkpoint Object dialog box appears, along with the current configuration of the selected checkpointing environment.
     The Add/Modify Checkpoint Object dialog box enables you to change the following information:

- Name
- Interface or checkpointing method
- Checkpoint, Migration, Restart, and Clean command strings
- Directory where checkpointing files are stored
- Occasions when checkpoints must be initiated
- Signal to send to job or application when a checkpoint is initiated
  See the `checkpoint(5)` man page for details about these parameters.

> Note
> - You must define the Interface or checkpointing method to use. From the Interface list under Name, select an Interface. See the `checkpoint(5)` man page for details about the meaning of the different interfaces. For the checkpointing environments provided with the distribution of the Grid Engine system, change only the Name parameter and the Checkpointing Directory parameter.

3. Click OK to register your changes with **`sge_qmaster`**.
   Click Cancel to close the dialog box without saving changes.

4. When you have configured all checkpointing environments, click Done to close the Checkpointing Configuration dialog box.

Managing Parallel Environments

A parallel environment (PE) is a software package that enables concurrent computing on parallel platforms in networked environments.

A variety of systems have evolved over the past years into viable technology for distributed and parallel processing on various hardware platforms. The following are two examples of the most common message-passing environments:

- Parallel Virtual Machine (PVM) from Oak Ridge National Laboratories
- Message Passing Interface (MPI) from the Message Passing Interface Forum

Public domain as well as hardware vendor-provided implementations exist for both tools.

All these systems show different characteristics and have separate requirements. To handle parallel jobs running on top of such systems, the Grid Engine system provides a flexible, powerful interface that satisfies various needs.

The Grid Engine system enables you to run parallel jobs through the following programs:

- Arbitrary message-passing environments such as PVM or MPI. See the PVM User's Guide and the MPI User's Guide for details.
- Shared memory parallel programs on multiple slots, either in single queues or distributed across multiple queues and across machines for distributed memory parallel jobs.

Any number of different parallel environment interfaces can be configured concurrently.

Interfaces between parallel environments and the Grid Engine system can be implemented if suitable startup and stop procedures are provided. The startup procedure and the stop procedure are described in Parallel Environment Startup Procedure and in Termination of the Parallel Environment.

Parallel Environment Tasks

Task	User Interface	Description
How to Configure Parallel Environments	CLI or QMON	Learn how to configure parallel environments.

Additional Topics

Topic	Description
Parallel Environment Startup Procedure	Learn about the parallel environment startup procedure.
Parallel Environment Termination Procedure	Learn about the parallel environment termination procedure.

Tight Integration of Parallel Environments and Grid Engine	Learn how to tightly integrate parallel environments.
Configuring and Using the Hadoop Integration	Learn how to set up Hadoop as a tightly integrated parallel environment

How to Configure Parallel Environments From the Command Line

To configure the parallel environment from the command line, use the following arguments for the `qconf` command:

> Note
> To run parallel jobs, you must also associate a queue with the parallel environment. See Configuring Queues for more information. Use the `queue_conf(5)` attribute `pe_list` to identify the suited parallel environments. Then, to link the parallel environment and queues, use either the QMON utility or the following form of the `qconf` command:
>
> ```
> # qconf -mq <queue_name>
> ```

- To display a parallel environment, type the following command:

  ```
  qconf -sp <pename>
  ```

  The `-sp` option (show parallel environment) prints the configuration of the specified parallel environment to standard output.

- To display a list of all currently-configured parallel environments, type the following command:

  ```
  qconf -spl
  ```

  The `-spl` option (show parallel environment list) lists the names of all currently configured parallel environments.

- To add a parallel environment, type the following command:

  ```
  qconf -ap <pename>
  ```

  The `-ap` option (add parallel environment) displays an editor containing a parallel environment configuration template. The editor is either the default `vi` editor or the editor defined by the `EDITOR` environment variable. pe-name specifies the name of the parallel environment. The name is already provided in the corresponding field of the template. To configure the parallel environment, change and save the template. See the `sge_pe(5)` man page for a detailed description of the template entries to change.

- To add a parallel environment from file, type the following command:

  ```
  qconf -AP <filename>
  ```

  The -Ap option (add parallel environment from file) parses the specified file and adds the new parallel environment configuration. The file must have the format of the parallel environment configuration template.

- To modify a parallel environment, type the following command:

  ```
  qconf -mp <pename>
  ```

  The `-mp` option (modify parallel environment) displays an editor containing the specified parallel environment as a configuration template. The editor is either the default `vi` editor or the editor defined by the `EDITOR` environment variable. To modify the parallel environment, change and save the template. See the sge_pe(5) man page for a detailed description of the template entries to change.

- To modify a parallel environment from file, type the following command:

  ```
  qconf -Mp <filename>
  ```

  The -Mp option (modify parallel environment from file) parses the specified file and modifies the existing parallel environment configuration. The file must have the format of the parallel environment configuration template.

- To delete a parallel environment, type the following command:

  ```
  qconf -dp <pename>
  ```

  The -dp option (delete parallel environment) deletes the specified parallel environment.

Example - Configuring a Parallel Environment From the Command Line

The qsub command that corresponds to the parallel job specification in Example - Displaying Configured Parallel Environment Interfaces With QMON is as follows:

```
% qsub -N Flow -p -111 -P devel -a 200012240000.00 -cwd \
 -S /bin/tcsh -o flow.out -j y -pe mpi 4-16 \
 -v SHARED_MEM=TRUE,MODEL_SIZE=LARGE \
 -ac JOB_STEP=preprocessing,PORT=1234 \
 -A FLOW -w w -r y -m s,e -q big_q\
 -M me@myhost.com,me@other.address \
 flow.sh big.data
```

This example shows how to use the qsub -pe command to formulate an equivalent request. The qsub(1) man page provides more details about the -pe option.

Select a suitable parallel environment interface for a parallel job, keeping the following considerations in mind:

- Parallel environment interfaces can use different message-passing systems or no message systems.
- Parallel environment interfaces can allocate processes on single or multiple hosts.
- Access to the parallel environment can be denied to certain users.
- Only a specific set of queues can be used by a parallel environment interface.
- Only a certain number of queue slots can be occupied by a parallel environment interface at any point of time.

Ask your Grid Engine administrator for the available parallel environment interfaces best suited for your types of parallel jobs.

You can specify resource requirements along with your parallel environment request. The specifying of resource requirements further reduces the set of eligible queues for the parallel environment interface to those queues that fit the requirement. See Managing Resource Quotas.

For example, assume that you run the following command:

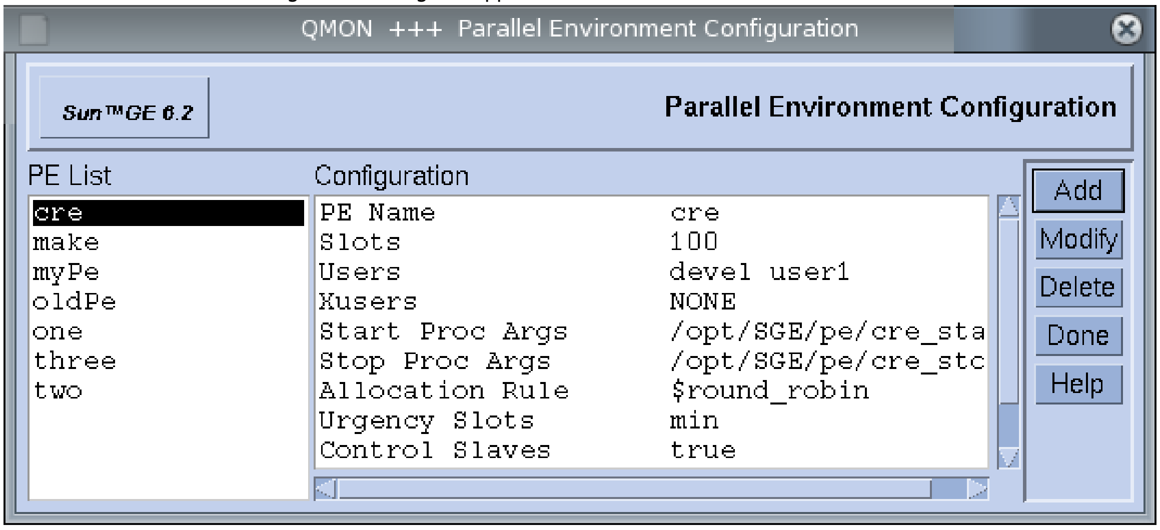
```
% qsub -pe mpi 1,2,4,8 -l nastran,arch=osf nastran.par
```

The queues that are suitable for this job are queues that are associated with the parallel environment interface mpi by the parallel environment configuration. Suitable queues also satisfy the resource requirement specification specified by the qsub -l command.

Note
The parallel environment interface facility is highly configurable. In particular, the administrator can configure the parallel environment startup and stop procedures to support site-specific needs. See the sge_pe(5) man page for details.

Use the `qsub -v` and `qsub -V` commands to pass information from the user who submits the job to the startup and stop procedures. These two options export environment variables. If you are unsure, ask the administrator whether you are required to export certain environment variables.

How to Configure Parallel Environments With QMON

1. On the QMON Main Control window, click the Parallel Environment Configuration button.
   The Parallel Environment Configuration dialog box appears.
   
   QMON +++ Parallel Environment Configuration
Sun™GE 6.2
Parallel Environment Configuration
PE List
Configuration
Add
Modify
Delete
Done
Help

   cre
   make
   myPe
   oldPe
   one
   three
   two

```
PE Name              cre
Slots                100
Users                devel user1
Xusers               NONE
Start Proc Args      /opt/SGE/pe/cre_sta
Stop Proc Args       /opt/SGE/pe/cre_stc
Allocation Rule      $round_robin
Urgency Slots        min
Control Slaves       true
```

   Currently configured parallel environments are displayed under PE List.

2. To display the contents of a parallel environment, select it.
   The selected parallel environment configuration is displayed under `Configuration`.

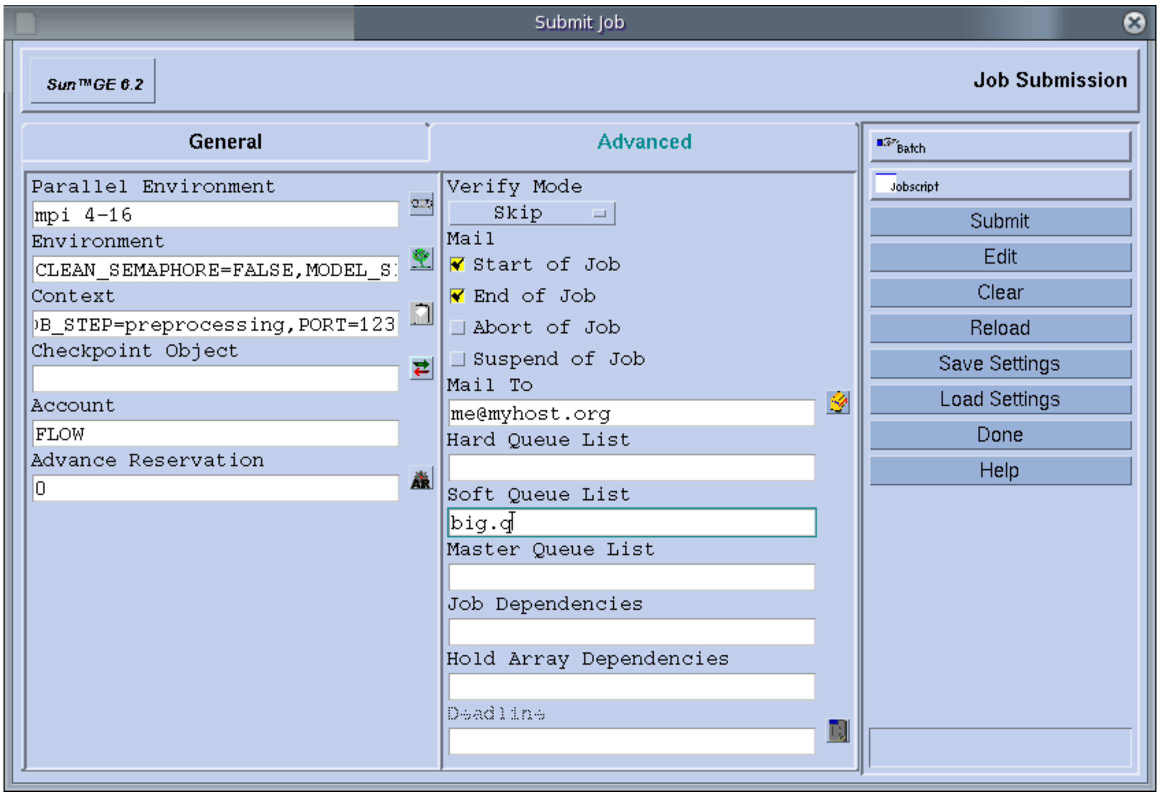
3. To add a new parallel environment, click Add.
   The Add/Modify PE dialog box appears. See How to Configure Parallel Environments With QMON.

4. To modify a parallel environment, select it, and then click Modify.
   The Add/Modify PE dialog box appears. See How to Configure Parallel Environments With QMON.

5. To delete a parallel environment, select it, and then click Delete.

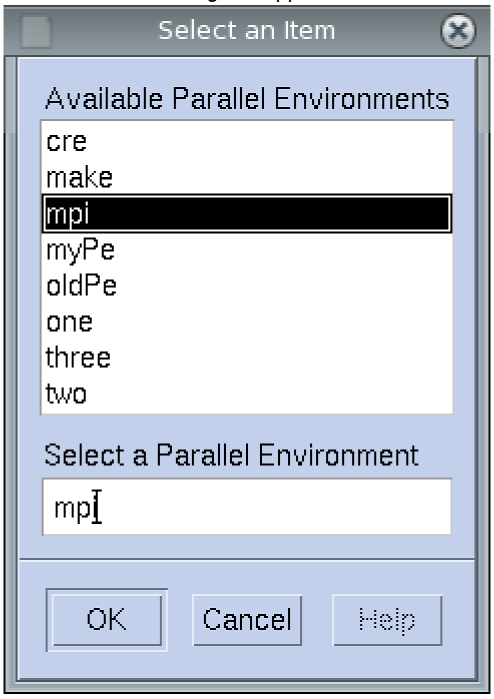
Example - Displaying Configured Parallel Environment Interfaces With QMON

The following example defines a parallel job to be submitted. The job requests that the parallel environment interface `mpi` (message passing interface) be used with from 4 to 16 processes, with 16 being preferable.



Parallel Environment: mpi 4-16
Environment: CLEAN_SEMAPHORE=FALSE,MODEL_S
Context: B_STEP=preprocessing,PORT=123
Checkpoint Object:
Account: FLOW
Advance Reservation: 0
Verify Mode: Skip
Mail: [✓ Start of Job] [✓ End of Job] Abort of Job, Suspend of Job
Mail To: me@myhost.org
Hard Queue List:
Soft Queue List: big.q
Master Queue List:
Job Dependencies:
Hold Array Dependencies:
Deadline:

- To select a parallel environment from a list of available parallel environments, click the button at the right of the Parallel Environment field. A selection dialog box appears.


Select an Item
Available Parallel Environments
cre
make
mpi
myPe
oldPe
one
three
two
Select a Parallel Environment
mpi
OK
Cancel
Help

- You can add a range for the number of parallel tasks initiated by the job after the parallel environment name in the Parallel Environment field.

Parallel Environment Startup Procedure

The Grid Engine software starts the parallel environment by using the exec system call to invoke a startup procedure. The name of the startup executable and the parameters passed to this executable are configurable from within the Grid Engine software.

An example for such a startup procedure for the PVM environment is contained in the distribution tree of the Grid Engine software. The startup procedure is made up of a shell script and a C program that is invoked by the shell script. The shell script uses the C program to start up PVM cleanly. All other required operations are handled by the shell script.

The shell script is located under `$SGE_ROOT/pvm/startpvm.sh`. The C program file is located under `$SGE_ROOT/pvm/src/start_pvm.c`.

> Note
> The startup procedure could have been a single C program. The use of a shell script enables easier customization of the sample startup procedure.

The example script `startpvm.sh` requires the following three arguments:

- The path of a host file generated by Grid Engine software, containing the names of the hosts from which PVM is to be started
- The host on which the `startpvm.sh` procedure is invoked
- The path of the PVM root directory, usually contained in the PVM_ROOT environment variable

These parameters can be passed to the startup script as described in How to Configure Parallel Environments With QMON. The parameters are among the parameters provided to parallel environment startup and stop scripts by the Grid Engine software during runtime. The required host file, as an example, is generated by the Grid Engine software. The name of the file can be passed to the startup procedure in the parallel environment configuration by the special parameter name `$pe_hostfile`. A description of all available parameters is provided in the `sge_pe` (5) man page.

The host file has the following format:

- Each line of the file refers to a queue on which parallel processes are to run.
- The first entry of each line specifies the host name of the queue.
- The second entry specifies the number of parallel processes to run in this queue.
- The third entry denotes the queue.
- The fourth entry denotes a processor range to use in case of a multiprocessor machine.

This file format is generated by the Grid Engine software. The file format is fixed. Parallel environments that need a different file format must translate it within the startup procedure. See the `startpvm.sh` file. PVM is an example of a parallel environment that needs a different file format.

When the Grid Engine software starts the parallel environment startup procedure, the startup procedure launches the parallel environment. The startup procedure should exit with a zero exit status. If the exit status of the startup procedure is not zero, Grid Engine software reports an error and does not start the parallel job.

> Note
> You should test any startup procedures first from the command line, without using the Grid Engine system. Doing so avoids all errors that can be hard to trace if the procedure is integrated into the Grid Engine software framework.

Parallel Environment Termination Procedure

When a parallel job finishes or is aborted, for example, by `qdel`, a procedure to halt the parallel environment is called. The definition and semantics of this procedure are similar to the procedures described for the startup program. The stop procedure can also be defined in a parallel environment configuration. See, for example, How to Configure Parallel Environments With QMON.

The purpose of the stop procedure is to shut down the parallel environment and to reap all associated processes.

> Note
> If the stop procedure fails to clean up parallel environment processes, the Grid Engine system might have no information about processes that are running under parallel environment control. Therefore the stop procedure cannot clean up these processes. The Grid Engine software, of course, cleans up the processes directly associated with the job script that the system has launched.

The distribution tree of the Grid Engine software also contains an example of a stop procedure for the PVM parallel environment. This example resides under `$SGE_ROOT/pvm/stoppvm.sh`. It takes the following two arguments:

- The path to the host file generated by the Grid Engine system
- The name of the host on which the stop procedure is started

Similar to the startup procedure, the stop procedure is expected to return a zero exit status on success and a nonzero exit status on failure.

> Note
> You should test any stop procedures first from the command line, without using the Grid Engine software. Doing so avoids all errors that can be hard to trace if the procedure is integrated into the Grid Engine framework.

Tight Integration of Parallel Environments and Grid Engine

How to Configure Parallel Environments With QMON mentions that using `sge_execd` and `sge_shepherd` to create parallel tasks offers benefits over parallel environments that create their own parallel tasks. The UNIX operating system allows reliable resource control only for the creator of a process hierarchy. Features such as correct accounting, resource limits, and process control for parallel applications, can be enforced only by the creator of all parallel tasks.

Most parallel environments do not implement these features. Therefore parallel environments do not provide a sufficient interface for the integration with a resource management system like the Grid Engine system. To overcome this problem, the Grid Engine system provides an advanced parallel environment interface for tight integration with parallel environments. This parallel environment interface transfers the responsibility for creating tasks from the parallel environment to the Grid Engine software.

The distribution of the Grid Engine system contains two examples of such a tight integration, one for the PVM public domain version, and one for the MPICH MPI implementation from Argonne National Laboratories. The examples are contained in the directories `$SGE_ROOT/pvm` and `$SGE_ROOT/mpi`, respectively. The directories also contain README files that describe the usage and any current restrictions. Refer to those README files for more details.

For the purpose of comparison, the `$SGE_ROOT/mpi/sunhpc/loose-integration` directory contains a loose integration sample with Sun HPC ClusterTools software, and the `$SGE_ROOT/mpi` directory contains a loosely integrated variant of the interfaces for comparison.

> Note
> The performance of a tight integration with a parallel environment is an advanced task that can require expert knowledge of the parallel environment and the Grid Engine software parallel environment interface. You might want to contact your Sun support representative for assistance.

Managing Policies

The Grid Engine software orchestrates the delivery of computational power, based on enterprise resource policies that the administrator manages. The software uses these policies to examine available computer resources in the grid. The software gathers these resources, and then it allocates and delivers them automatically, in a way that optimizes usage across the grid.

To enable cooperation in the grid, project owners must do the following:

- Negotiate policies
- Ensure that policies for manual overrides for unique project requirements are flexible
- Automatically monitor and enforce policies

Policy management automatically controls the use of shared resources in the cluster to achieve your goals. High-priority jobs are dispatched preferentially. These jobs receive greater CPU entitlements when they are competing with other, lower-priority jobs. The Grid Engine software monitors the progress of all jobs. It adjusts their relative priorities correspondingly, and with respect to the goals that you define in the policies.

This policy-based resource allocation grants each user, team, department, and all projects an allocated share of system resources. This allocation of resources extends over a specified period of time, such as a week, a month, or a quarter.

Grid Engine policies are implemented in conjunction with the Grid Engine scheduler. For information, see Managing the Scheduler.

Topic	Description
Specifying Policy Priority	Learn how to specify policy priority.
Configuring the Share-Based Policy	Learn how to configure the share-based policy.
Configuring the Functional Policy	Learn how to configure the functional policy.
Configuring the Override Policy	Learn how to configure the override policy.
Configuring the Urgency Policy	Learn how to configure the urgency policy.
Configuring Ticket-Based Policies	Learn how to configure ticket-based policies.

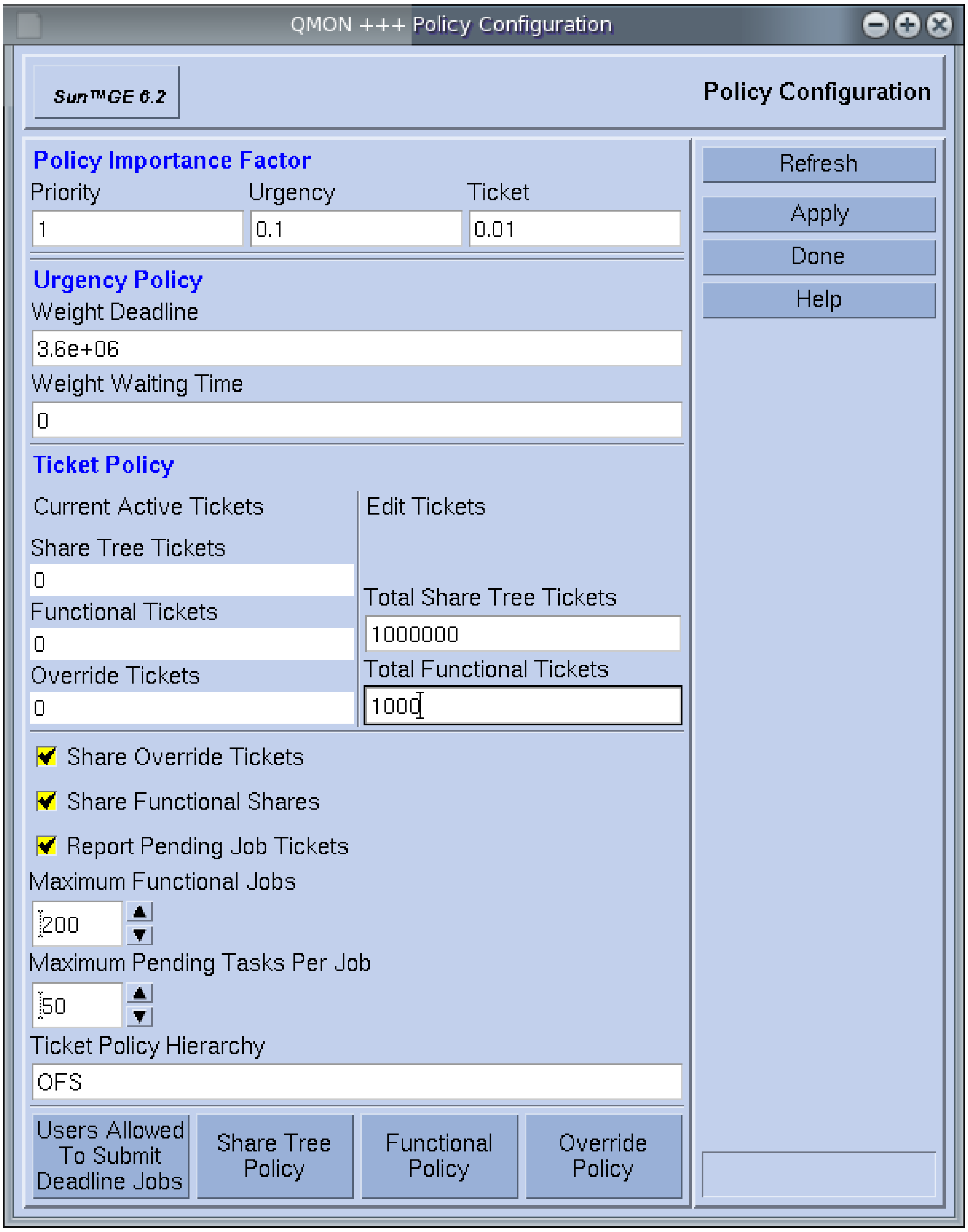
Policy Configuration Tasks

Note
You must use the QMON graphical interface to perform certain policy configuration tasks, such as configuring the share tree policy in detail. However, you can use the command line to perform some simple functions.

Task	User Interface	Description
How to Configure the Share-Based Policy	CLI	Learn how to configure the share-based policy.
How to Configure the Functional Share Policy	CLI	Learn how to configure the functional share policy.
How to Configure the Override Policy	CLI	Learn how to configure the override policy.
How to Configure Policy-Based Resource Management	QMON	Learn how to configure policy-based resource management.
How to Create Project-Based Share-Tree Scheduling	QMON	Learn how to create project-based, share-tree scheduling.
How to Create User-Based, Project-Based, and Department-Based Functional Scheduling	QMON	Learn how to create user-based, project-based, and department-based functional scheduling.

How to Configure Policy-Based Resource Management With QMON

1. On the QMON Main Control window, click the Policy Configuration button.
   The Policy Configuration dialog box appears, as shown below.


QMON +++ Policy Configuration

Sun™GE 6.2

Policy Configuration

Policy Importance Factor

Priority	Urgency	Ticket
1	0.1	0.01

Urgency Policy

Weight Deadline: 3.6e+06

Weight Waiting Time: 0

Ticket Policy

Current Active Tickets		Edit Tickets	
Share Tree Tickets	0	Total Share Tree Tickets	1000000
Functional Tickets	0	Total Functional Tickets	1000
Override Tickets	0		

✓ Share Override Tickets

✓ Share Functional Shares

✓ Report Pending Job Tickets

Maximum Functional Jobs: 200

Maximum Pending Tasks Per Job: 50

Ticket Policy Hierarchy: OFS

2. From the Policy Configuration dialog box, you can edit the following information:
   If you need to refresh the information displayed in the Policy Configuration dialog box, click Refresh.
   - Policy Importance Factor – Specifies the relative importance of the priority types that control the sorting order of jobs. See Specifying Policy Priority.
   - Urgency Policy – Defines an urgency value for each job. See Configuring the Urgency Policy.
   - Ticket Policy – Lets you modify the number of tickets that are allocated to the share tree policy and the functional policy. See Configuring Ticket-Based Policies. You can also access detailed configuration dialog boxes for the three ticket-based policies.
     - Share-based policy – See Configuring the Share-Based Policy
     - Functional policy – See Configuring the Functional Policy
     - Override policy – See Configuring the Override Policy

3. Click Apply to save any changes that you make to the Policy Configuration.
   Click Done to close the dialog box without saving changes.

Specifying Policy Priority

Before the Grid Engine system dispatches jobs, the jobs are brought into priority order, highest priority first. Without any administrator influence, the order is first-in, first-out (FIFO).

On the Policy Configuration dialog box, under Policy Importance Factor, you can specify the relative importance of the three priority types that control the sorting order of jobs. For example, if you specify Priority as 1, Urgency as 0.1, and Ticket as 0.01, job priority that is specified by the `qsub -p` command is given the most weight, job priority that is specified by the Urgency Policy is considered next, and job priority that is specified by the Ticket Policy is given the least weight.

- Priority – Also called POSIX priority. The `-p` option of the `qsub` command specifies site-specific priority policies.
- Urgency Policy – Jobs can have an urgency value that determines their relative importance. Pending jobs are sorted according to their urgency value.
- Ticket Policy – Jobs are always treated according to their relative importance as defined by the number of tickets that the jobs have. Pending jobs are sorted in ticket order.

For more information about job priorities, see Job Sorting.

You can specify a weighting factor for each priority type. This weighting factor determines the degree to which each type of priority affects overall job priority. To make it easier to control the range of values for each priority type, normalized values are used instead of the raw ticket values, urgency values, and POSIX priority values.

The following formula expresses how a job's priority values are determined:

```
Job priority = Urgency * normalized urgency value +
Ticket * normalized ticket value +
Priority * normalized priority value
```

Configuring the Urgency Policy

The Urgency Policy defines an urgency value for each job. This urgency value is determined by the sum of the following three contributing elements:

- Resource requirement – Each resource attribute defined in the complex can have an urgency value. For information about the setting urgency values for resource attributes, see Configuring Complex Resource Attributes With QMON. Each job request for a resource attribute adds the attribute's urgency value to the total.
- Deadline – The urgency value for deadline jobs is determined by dividing the Weight Deadline specified in the Policy Configuration dialog box by the free time, in seconds, until the job's deadline initiation time specified by the `qsub -dl` command. Users must be added to the `deadlineusers` access list in order to use this policy. For more information, see How to Configure User Access Lists.
- Waiting time – The urgency value for a job's waiting time is determined by multiplying the job's waiting time by the Weight Waiting Time specified in the Policy Configuration dialog box. The job's waiting time is measured in seconds.

For details about how the Grid Engine system arrives at the urgency value total, see About the Urgency Policy.

Configuring Ticket-Based Policies

The tickets that are currently assigned to individual policies are listed under Current Active Tickets in the Policy Configuration dialog box. The numbers reflect the relative importance of the policies. The numbers indicate whether a certain policy currently dominates the cluster or whether policies are in balance.

Tickets provide a quantitative measure. For example, you might assign twice as many tickets to the share-based policy as you assign to the functional policy. This means that twice the resource entitlement is allocated to the share-based policy than is allocated to the functional policy. In this sense, tickets behave very much like stock shares.

The total number of all tickets has no particular meaning. Only the relations between policies counts. Hence, total ticket numbers are usually quite high to allow for fine adjustment of the relative importance of the policies.

Under Edit Tickets, you can modify the number of tickets that are allocated to the share tree policy and the functional policy. For details, see Editing Tickets.

Select the Share Override Tickets check box to control the total ticket amount distributed by the override policy. Deselect the Share Override Tickets check box to control the importance of individual jobs relative to the ticket pools that are available for the other policies and override categories. With this setting, the number of jobs that are under a category member does not matter. The jobs always get the same number of tickets. However, the total number of override tickets in the system increases as the number of jobs with a right to receive override tickets increases. Other policies can lose importance in such cases. For detailed information, see Sharing Override Tickets.

Select the Share Functional Tickets check box to give a category member a constant entitlement level for the sum of all its jobs. Deselect the check box to give each job the same entitlement level, based on its category member's entitlement. For detailed information, see Sharing Functional Ticket Shares.

You can set the maximum number of jobs that can be scheduled in the functional policy. The default value is 200.

You can set the maximum number of pending subtasks that are allowed for each array job. The default value is 50. Use this setting to reduce scheduling overhead.

You can specify the Ticket Policy Hierarchy to resolve certain cases of conflicting policies. The resolving of policy conflicts applies particularly to pending jobs. For detailed information, see Setting the Ticket Policy Hierarchy.

Editing Tickets

You can edit the total number of share-tree tickets and functional tickets. Override tickets are assigned directly through the override policy configuration. The other ticket pools are distributed automatically among jobs that are associated with the policies and with respect to the actual policy configuration.

> Note
> All share-based tickets and functional tickets are always distributed among the jobs associated with these policies. Override tickets might not be applicable to the currently active jobs. Consequently, the active override tickets might be zero, even though the override policy has tickets defined.

Sharing Override Tickets

The administrator assigns tickets to the different members of the override categories, that is, to individual users, projects, departments, or jobs. Consequently, the number of tickets that are assigned to a category member determines how many tickets are assigned to jobs under that category member. For example, the number of tickets that are assigned to user A determines how many tickets are assigned to all jobs of user A.

> Note
> The number of tickets that are assigned to the job category does not determine how many tickets are assigned to jobs in that category.

Use the Share Override Tickets check box to set the `share_override_tickets` parameter of `sched_conf`(5). This parameter controls how job ticket values are derived from their category member ticket value. When you select the Share Override Tickets check box, the tickets of the category members are distributed evenly among the jobs under this member. If you deselect the Share Override Tickets check box, each job inherits the ticket amount defined for its category member. In other words, the category member tickets are replicated for all jobs underneath.

Select the Share Override Tickets check box to control the total ticket amount distributed by the override policy. With this setting, ticket amounts that are assigned to a job can become negligibly small if many jobs are under one category member. For example, ticket amounts might diminish if many jobs belong to one member of the user category.

Deselect the Share Override Tickets check box to control the importance of individual jobs relative to the ticket pools that are available for the other policies and override categories. With this setting, the number of jobs that are under a category member does not matter. The jobs always get the same number of tickets. However, the total number of override tickets in the system increases as the number of jobs with a right to receive override tickets increases. Other policies can lose importance in such cases.

Sharing Functional Ticket Shares

The functional policy defines entitlement shares for the functional categories. Then the policy defines shares for all members of each of these categories. The functional policy is thus similar to a two-level share tree. The difference is that a job can be associated with several categories at the same time. The job belongs to a particular user, for instance, but the job can also belong to a project or a department.

However, as in the share tree, the entitlement shares that a job receives from a functional category is determined by the following:

- The shares that are defined for its corresponding category member (for example, its project)
- The shares that are given to the category (project instead of user, department, and so on)

Use the Share Functional Tickets check box to set the `share_functional_shares` parameter of `sched_conf`(5). This parameter defines how the category member shares are used to determine the shares of a job. The shares assigned to the category members, such as a particular user or project, can be replicated for each job. Alternatively, shares can be distributed among the jobs under the category member.

- Selecting the Share Functional Tickets check box means that functional shares are replicated among jobs.
- Deselecting the Share Functional Tickets check box means that functional shares are distributed among jobs.

Those shares are comparable to stock shares. Such shares have no effect for the jobs that belong to the same category member. All jobs under the same category member have the same number of shares in both cases. But the share number has an effect when comparing the share amounts within the same category. Jobs with many siblings that belong to the same category member receive relatively small share portions if you select the Share Functional Tickets check box. On the other hand, if you clear the Share Functional Tickets check box, all sibling jobs receive the same share amount as their category member.

Select the Share Functional Tickets check box to give a category member a constant entitlement level for the sum of all its jobs. The entitlement of an individual job can get negligibly small, however, if the job has many siblings.

Deselect the Share Functional Tickets check box to give each job the same entitlement level, based on its category member's entitlement. The number of job siblings in the system does not matter.

> Note
> A category member with many jobs underneath can dominate the functional policy.

Be aware that the setting of share functional shares does not determine the total number of functional tickets that are distributed. The total number is always as defined by the administrator for the functional policy ticket pool. The share functional shares parameter influences only how functional tickets are distributed within the functional policy.

Example – Functional Policy

The following example describes a common scenario where a user wishes to translate the Sun Grid Engine 5.3 Scheduler Option `-user_sort true` to a Sun Grid Engine 6.2 configuration but does not understand the share override functional policy ticket feature.

For a plain user-based equal share, you configure your global configuration `sge_conf(5)` with

`-enforce_user auto`
`-auto_user_fshare 100`

Then you use `-weight_tickets_functional 10000` in the scheduler configuration `sched_conf(5)`. This action causes the functional policy to be used for user-based equal share scheduling with 100 shares for each user.

Tuning Scheduling Run Time

Pending jobs are sorted according to the number of tickets that each job has, as described in Job Sorting. The scheduler reports the number of tickets each pending job has to the master daemon `sge_qmaster`. However, on systems with very large numbers of jobs, you might want to turn off ticket reporting. When you turn off ticket reporting, you disable ticket-based job priority. The sort order of jobs is based only on the time each job is submitted.

To turn off the reporting of pending job tickets to `sge_qmaster`, clear the `Report Pending Job Tickets` check box on the `Policy Configuration` dialog box. Doing so sets the `report_pjob_tickets` parameter of `sched_conf`(5) to false.

Setting the Ticket Policy Hierarchy

Ticket policy hierarchy provides the means to resolve certain cases of conflicting ticket policies. The resolving of ticket policy conflicts applies particularly to pending jobs.

Such cases can occur in combination with the share-based policy and the functional policy. With both policies, assigning priorities to jobs that belong to the same leaf-level entities is done on a first-come, first-served basis. Leaf-level entities include the following:

- User leaves in the share tree
- Project leaves in the share tree
- Any member of the following categories in the functional policy: user, project, department, or queue

Members of the job category are not included among leaf-level entities. So, for example, the first job of the same user gets the most, the second gets the next most, the third next, and so on.

A conflict can occur if another policy mandates an order that is different. So, for example, the override policy might define the third job as the most important, whereas the first job that is submitted should come last.

A policy hierarchy might gives the override policy higher priority over the share-tree policy or the functional policy. Such a policy hierarchy ensures that high-priority jobs under the override policy get more entitlements than jobs in the other two policies. Such jobs must belong to the same leaf level entity (user or project) in the share tree.

The Ticket Policy Hierarchy can be a combination of up to three letters. These letters are the first letters of the names of the following three ticket policies:

- S – Share-based
- F – Functional
- O – Override

Use these letters to establish a hierarchy of ticket policies. The first letter defines the top policy. The last letter defines the bottom of the hierarchy. Policies that are not listed in the policy hierarchy do not influence the hierarchy. However, policies that are not listed in the hierarchy can still be a source for tickets of jobs. However, those tickets do not influence the ticket calculations in other policies. All tickets of all policies are added up for each job to define its overall entitlement.

The following examples describe two settings and how they influence the order of the pending jobs:

- Override, Share-Based (OS) Hierarchy Setting

  ```
  policy_hierarchy=OS
  ```

  1. The override policy assigns the appropriate number of tickets to each pending job.
  2. The number of tickets determines the entitlement assignment in the share tree in case two jobs belong to the same user or to the same leaf-level project. Then the share tree tickets are calculated for the pending jobs.
  3. The tickets from the override policy and from the share-tree policy are added together, along with all other active policies not in the hierarchy. The job with the highest resulting number of tickets has the highest entitlement.
- Override, Functional (OF) Hierarchy Setting

  ```
  policy_hierarchy=OF
  ```

  1. The override policy assigns the appropriate number of tickets to each pending job. Then the tickets from the override policy are added up.
  2. The resulting number of tickets influences the entitlement assignment in the functional policy in case two jobs belong to the same functional category member. Based on this entitlement assignment, the functional tickets are calculated for the pending jobs.
  3. The resulting value is added to the ticket amount from the override policy. The job with the highest resulting number of tickets has the highest entitlement.

All combinations of the three letters are theoretically possible, but only a subset of the combinations are meaningful or have practical relevance. The last letter should always be S or F, because only those two policies can be influenced due to their characteristics described in the examples.

The following form is recommended for `policy_hierarchy` settings:

```
[O][S|F]
```

If the override policy is present, O should occur as the first letter only, because the override policy can only influence. The share-based policy and the functional policy can only be influenced. Therefore S or F should occur as the last letter.

Configuring the Share-Based Policy

Share-based scheduling grants each user and project its allocated share of system resources during an accumulation period such as a week, a month, or a quarter. Share-based scheduling is also called share tree scheduling. It constantly adjusts each user's and project's potential resource share for the near term, until the next scheduling interval. Share-based scheduling is defined for user or for project, or for both.

Share-based scheduling ensures that a defined share is guaranteed to the instances that are configured in the share tree over time. Jobs that are associated with share-tree branches where fewer resources were consumed in the past than anticipated are preferred when the system dispatches jobs. At the same time, full resource usage is guaranteed, because unused share proportions are still available for pending jobs associated with other share-tree branches.

By giving each user or project its targeted share as far as possible, groups of users or projects also get their targeted share. Departments or divisions are examples of such groups. Fair share for all entities is attainable only when every entity that is entitled to resources contends for those resources during the accumulation period. If a user, a project, or a group does not submit jobs during a given period, the resources are shared among those who do submit jobs.

Share-based scheduling is a feedback scheme. The share of the system to which any user or user-group, or project or project-group, is entitled is a configuration parameter. The share of the system to which any job is entitled is based on the following factors:

- The share allocated to the job's user or project
- The accumulated past usage for each user and user group, and for each project and project group. This usage is adjusted by a decay factor. "Old" usage has less impact.

The Grid Engine software keeps track of how much usage users and projects have already received. At each scheduling interval, the Scheduler adjusts all jobs' share of resources. Doing so ensures that all users, user groups, projects, and project groups get close to their fair share of the system during the accumulation period. In other words, resources are granted or are denied to keep everyone more or less at their targeted share of usage.

The Half-Life Factor

Half-life is how fast the system "forgets" about a user's resource consumption. The administrator decides whether to penalize a user for high resource consumption, be it six months ago or six days ago. The administrator also decides how to apply the penalty. On each node of the share tree, Grid Engine software maintains a record of users' resource consumption.

With this record, the system administrator can decide how far to look back to determine a user's under-usage or over-usage when setting up a share-based policy. The resource usage in this context is the mathematical sum of all the computer resources that are consumed over a "sliding window of time."

The length of this window is determined by a "half-life" factor, which in the Grid Engine system is an internal decay function. This decay function reduces the impact of accrued resource consumption over time. A short half-life quickly lessens the impact of resource overconsumption. A longer half-life gradually lessens the impact of resource overconsumption.

This half-life decay function is a specified unit of time. For example, consider a half-life of seven days that is applied to a resource consumption of 1,000 units. This half-life decay factor results in the following usage "penalty" adjustment over time:

- 500 after 7 days
- 250 after 14 days
- 125 after 21 days
- 62.5 after 28 days

The half-life-based decay diminishes the impact of a user's resource consumption over time, until the effect of the penalty is negligible.

> Note
> Override tickets that a user receives are not subjected to a past usage penalty, because override tickets belong to a different policy system. The decay function is a characteristic of the share-tree policy only.

Compensation Factor

Sometimes the comparison shows that actual usage is well below targeted usage. In such a case, the adjusting of a user's share or a project's share of resource can allow a user to dominate the system. Such an adjustment is based on the goal of reaching target share. This domination might not be desirable.

The compensation factor enables an administrator to limit how much a user or a project can dominate the resources in the near term.

For example, a compensation factor of two limits a user's or project's current share to twice its targeted share. Assume that a user or a project should get 20 percent of the system resources over the accumulation period. If the user or project currently gets much less, the maximum that it can get in the near term is only 40 percent.

The share-based policy defines long-term resource entitlements of users or projects as per the share tree. When combined with the share-based policy, the compensation factor makes automatic adjustments in entitlements.

If a user or project is either under or over the defined target entitlement, the Grid Engine system compensates. The system raises or lowers that user's or project's entitlement for a short term over or under the long-term target. This compensation is calculated by a share tree algorithm.

The compensation factor provides an additional mechanism to control the amount of compensation that the Grid Engine system assigns. The additional compensation factor (CF) calculation is carried out only if the following conditions are true:

- Short-term-entitlement is greater than long-term-entitlement multiplied by the CF.
- The CF is greater than 0.

If either condition is not true, or if both conditions are not true, the compensation as defined and implemented by the share-tree algorithm is used.

The smaller the value of the CF, the greater is its effect. If the value is greater than 1, the Grid Engine system's compensation is limited. The upper limit for compensation is calculated as long-term-entitlement multiplied by the CF. And as defined earlier, the short-term entitlement must exceed this limit before anything happens based on the compensation factor.

If the CF is 1, the Grid Engine system compensates in the same way as with the raw share-tree algorithm. So a value of one has an effect that is similar to a value of zero. The only difference is an implementation detail. If the CF is one, the CF calculations are carried out without an effect. If the CF is zero, the calculations are suppressed.

If the value is less than 1, the Grid Engine system overcompensates. Jobs receive much more compensation than they are entitled to based on the share-tree algorithm. Jobs also receive this overcompensation earlier, because the criterion for activating the compensation is met at lower short-term entitlement values. The activating criterion is `short-term-entitlement > long-term-entitlement * CF`.

Hierarchical Share Tree

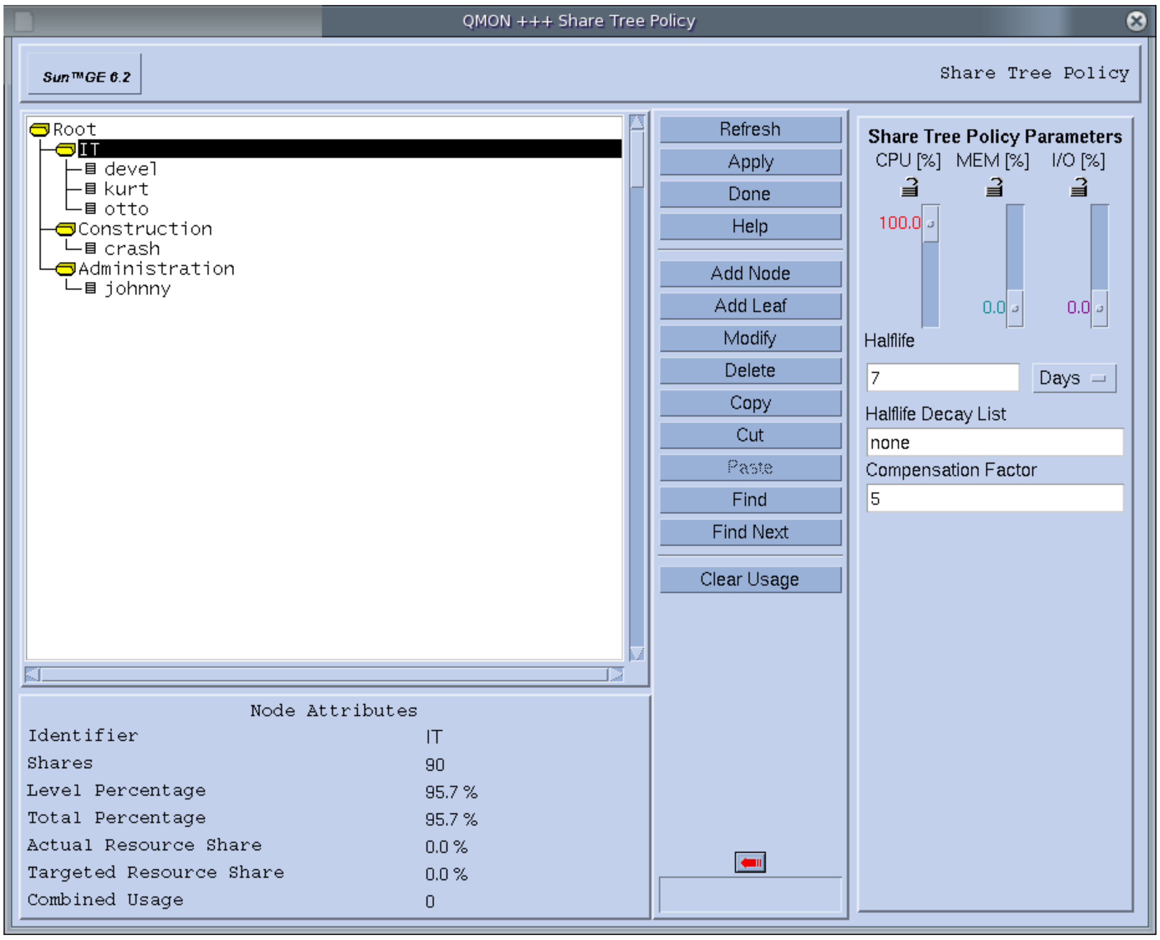
The share-based policy is implemented through a hierarchical share tree. The share tree specifies, for a moving accumulation period, how system resources are to be shared among all users and projects. The length of the accumulation period is determined by a configurable decay constant. The Grid Engine system bases a job's share entitlement on the degree to which each parent node in the share tree reaches its accumulation limit. A job's share entitlement is based on its leaf node share allocation, which in turn depends on the allocations of its parent nodes. All jobs associated with a leaf node split the associated shares.

The entitlement derived from the share tree is combined with other entitlements, such as entitlements from a functional policy, to determine a job's net entitlement. The share tree is allotted the total number of tickets for share-based scheduling. This number determines the weight of share-based scheduling among the four scheduling policies.

The share tree is defined during installation. The share tree can be altered at any time. When the share tree is edited, the new share allocations take effect at the next scheduling interval.

Configuring the Share-Tree Policy With QMON

On the `QMON Policy Configuration` dialog box, click `Share Tree Policy`. The `Share Tree Policy` dialog box appears.



Share Tree Policy Parameters

CPU [%]	MEM [%]	I/O [%]
100.0	0.0	0.0

Halflife: 7 Days

Halflife Decay List: none

Compensation Factor: 5

Node Attributes

Identifier	IT
Shares	90
Level Percentage	95.7 %
Total Percentage	95.7 %
Actual Resource Share	0.0 %
Targeted Resource Share	0.0 %
Combined Usage	0

Node Attributes

Under Node Attributes, the attributes of the selected node are displayed:

- Identifier – A user, project, or agglomeration name.
- Shares – The number of shares that are allocated to this user or project.

> Note
> Shares define relative importance. They are not percentages. Shares also do not have quantitative meaning. The specification of hundreds or even thousands of shares is generally a good idea, as high numbers allow fine tuning of importance relationships.

- Level Percentage – This node's portion of the total shares at the level of the same parent node in the tree. The number of this node's shares divided by the sum of its and its sibling's shares.
- Total Percentage – This node's portion of the total shares in the entire share tree. The long-term targeted resource share of the node.
- Actual Resource Usage – The percentage of all the resources in the system that this node has consumed so far in the accumulation period. The percentage is expressed in relation to all nodes in the share tree.
- Targeted Resource Usage – Same as Actual Resource Usage, but only taking the currently active nodes in the share tree into account. Active nodes have jobs in the system. In the short term, the Grid Engine system attempts to balance the entitlement among active nodes.
- Combined Usage – The total usage for the node. Combined Usage is the sum of the usage that is accumulated at this node. Leaf nodes accumulate the usage of all jobs that run under them. Inner nodes accumulate the usage of all descendant nodes. Combined Usage includes CPU, memory, and I/O usage according to the ratio specified under Share Tree Policy Parameters. Combined usage is decayed at the half-life decay rate that is specified by the parameters.

When a user node or a project node is removed and then added back, the user's or project's usage is retained. A node can be added back either at the same place or at a different place in the share tree. You can zero out that usage before you add the node back to the share tree. To do so, first remove the node from the users or projects configured in the Grid Engine system. Then add the node back to the users or projects there.

Users or projects that were not in the share tree but that ran jobs have nonzero usage when added to the share tree. To zero out usage when you add such users or projects to the tree, first remove them from the users or projects configured in the Grid Engine system. Then add them to the tree.

To add an interior node under the selected node, click Add Node. A blank Node Info window appears, where you can enter the node's name and number of shares. You can enter any node name or share number.

To add a leaf node under the selected node, click Add Leaf. A blank Node Info window appears, where you can enter the node's name and number of shares. The node's name must be an existing Grid Engine user (Configuring User Objects With QMON) or project (Defining Projects).

The following rules apply when you are adding a leaf node:

- All nodes have a unique path in share tree.
- A project is not referenced more than once in a share tree.
- A user appears only once in a project subtree.
- A user appears only once outside of a project subtree.
- A user does not appear as a nonleaf node.
- All leaf nodes in a project subtree reference a known user or the reserved name `default`. See a detailed description of this special user in About the Special User default.
- Project subtrees do not have sub-projects.
- All leaf nodes not in a project subtree reference a known user or known project.
- All user leaf nodes in a project subtree have access to the project.

To edit the selected node, click Modify. A Node Info window appears. The window displays the mode's name and its number of shares.

To cut or copy the selected node to a buffer, click Cut or Copy. To paste under the selected node the contents of the most recently cut or copied node, click Paste.

To delete the selected node and all its descendants, click Delete.

To clear the entire share-tree hierarchy, click Clear Usage. Clear the hierarchy when the share-based policy is aligned to a budget and needs to start from scratch at the beginning of each budget term. The Clear Usage facility also is handy when setting up or modifying test Grid Engine software environments.

QMON periodically updates the information displayed in the Share Tree Policy dialog box. Click Refresh to force the display to refresh immediately.

To save all the node changes that you make, click Apply. To close the dialog box without saving changes, click Done.

To search the share tree for a node name, click Find, and then type a search string. Node names are indicated which begin with the case sensitive search string. Click Find Next to find the next occurrence of the search string.

Click Help to open the online help system.

Share Tree Policy Parameters

To display the Share Tree Policy Parameters, click the arrow at the right of the Node Attributes.

- CPU [%] slider — This slider's setting indicates what the percentage of Combined Usage CPU is. When you change this slider, the MEM and I/O sliders change to compensate for the change in CPU percentage.
- MEM [%] slider — This slider's setting indicates what the percentage of Combined Usage memory is. When you change this slider, the CPU and I/O sliders change to compensate for the change in MEM percentage.
- I/O [%] slider — This slider's setting indicates what the percentage of Combined Usage I/O is. When you change this slider, the CPU and MEM sliders change to compensate for the change in I/O percentage.

> Note
> CPU %, MEM %, and I/O % always adds up to 100%.

- Lock Symbol — When a lock is open, the slider that it guards can change freely. The slider can change either because the slider was moved or because it is compensating for another slider's being moved. When a lock is closed, the slider that it guards cannot change. If two locks are closed and one lock is open, no sliders can be changed.
- Half-life — Use this field to specify the half-life for usage. Usage is decayed during each scheduling interval so that any particular contribution to accumulated usage has half the value after a duration of half-life.
- Days/Hours selection menu — Select whether half-life is to be measured in days or hours.
- Compensation Factor — This field accepts a positive integer for the compensation factor. Reasonable values range between 2 and 10.

The actual usage of a user or project can be far below its targeted usage. The compensation factor prevents such users or projects from dominating resources when they first get those resources. See Compensation Factor for more information.

About the Special User default

You can use the special user `default` to reduce the amount of share-tree maintenance for sites with many users. Under the share-tree policy, a job's priority is determined based on the node that the job maps to in the share tree. Users who are not explicitly named in the share tree are mapped to the `default` node, if it exists.

The specification of a single `default` node allows for a simple share tree to be created. Such a share tree makes user-based fair sharing possible.

You can use the `default` user also in cases where the same share entitlement is assigned to most users. Same share entitlement is also known as equal share scheduling.

The `default` user configures all user entries under the `default` node, giving the same share amount to each user. Each user who submits jobs receives the same share entitlement as that configured for the `default` user. To activate the facility for a particular user, you must add this user to the list of Grid Engine users.

The share tree displays "virtual" nodes for all users who are mapped to the `default` node. The display of virtual nodes enables you to examine the usage and the fair-share scheduling parameters for users who are mapped to the `default` node.

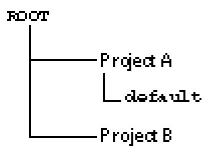
You can also use the `default` user for "hybrid" share trees, where users are subordinated under projects in the share tree. The `default` user can be a leaf node under a project node.

The short-term entitlements of users vary according to differences in the amount of resources that the users consume. However, long-term entitlements of users remain the same.

You might want to assign lower or higher entitlements to some users while maintaining the same long-term entitlement for all other users. To do so, configure a share tree with individual user entries next to the `default` user for those users with special entitlements.

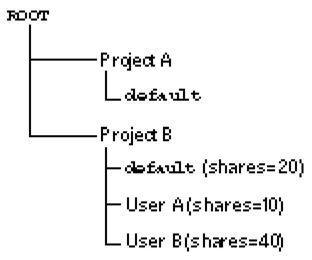
In Example A, all users submitting to Project A get equal long-term entitlements. The users submitting to Project B only contribute to the accumulated resource consumption of Project B. Entitlements of Project B users are not managed.

Example A



Compare Example A with Example B:

Example B



In Example B, treatment for Project A is the same as for Example A. But all default users who submit jobs to Project B, except users A and B, receive equal long-term resource entitlements. Default users have 20 shares. User A, with 10 shares, receives half the entitlement of the default users. User B, with 40 shares, receives twice the entitlement as the default users.

How to Create Project-Based Share-Tree Scheduling With QMON

The objective of this setup is to guarantee a certain share assignment of all the cluster resources to different projects over time.

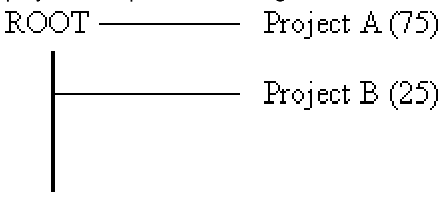
1. Specify the number of share-tree tickets (for example, 1000000) in the scheduler configuration.
   See How to Configure Policy-Based Resource Management With QMON, and the `sched_conf(5)` man page.

2. (Optional) Add one user for each scheduling-relevant user.
   See How to Configure User Objects With QMON, and the `user(5)` man page.

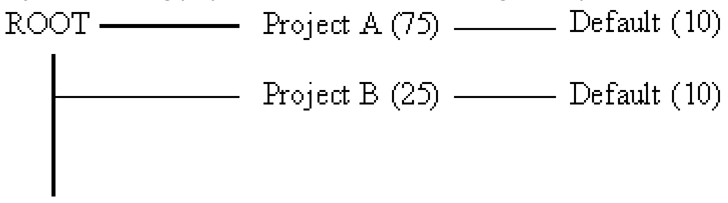
3. Add one project for each scheduling-relevant project.
   See Configuring the Share-Tree Policy With QMON, and the `project(5)` man page.

4. Use QMON to set up a share tree that reflects the structure of all scheduling-relevant projects as nodes.
   See Configuring the Share-Tree Policy With QMON.

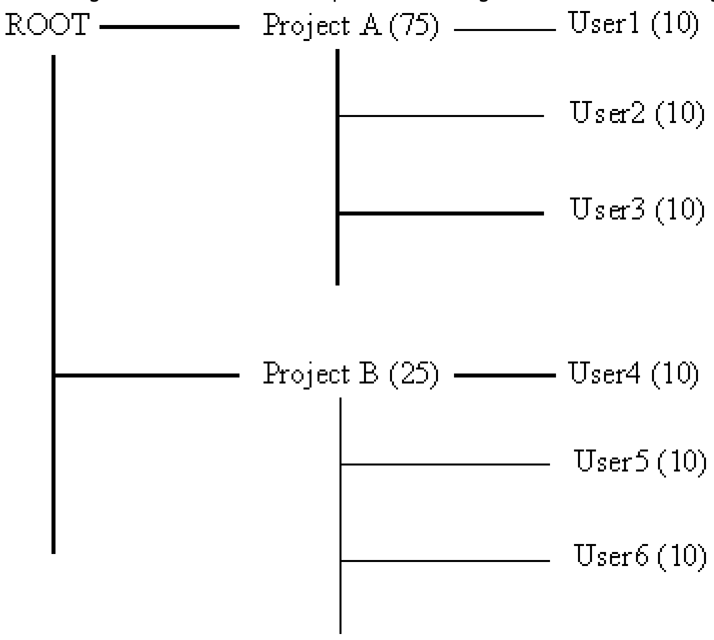
5. Assign share-tree shares to the projects.
   For example, if you are creating project-based share-tree scheduling with first-come, first-served scheduling among jobs of the same project, a simple structure might look like the following:



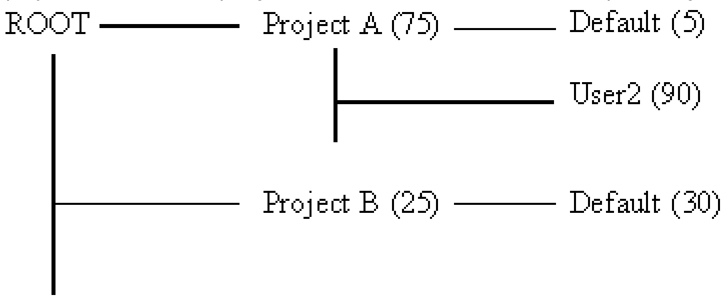
If you are creating project-based share-tree scheduling with equal shares for each user, a simple structure might look like the following:



If you are creating project-based share-tree scheduling with individual user shares in each project, add users as leaves to their projects. Then assign individual shares. A simple structure might look like the following:



If you want to assign individual shares to only a few users, designate the user `default` in combination with individual users below a project node. For example, you can condense the tree illustrated previously into the following:



Configuring the Functional Policy

Functional scheduling is a nonfeedback scheme for determining a job's importance. Functional scheduling associates a job with the submitting user, project, or department. Functional scheduling is sometimes called priority scheduling. The functional policy setup ensures that a defined share is guaranteed to each user, project, job, or department at any time. Jobs of users, projects, or departments that have used fewer resources than anticipated are preferred when the system dispatches jobs to idle resources.

At the same time, full resource usage is guaranteed, because unused share proportions are distributed among those users, projects, departments, and jobs that need the resources. Past resource consumption is not taken into account.

Functional policy entitlement to system resources is combined with other entitlements in determining a job's net entitlement. For example, functional policy entitlement might be combined with override policy entitlement.

The total number of tickets that are allotted to the functional policy determines the weight of functional scheduling among the scheduling policies. During installation, the administrator divides the total number of functional tickets among the functional categories of user, department, project, and job.

Functional Shares

Functional shares are assigned to every member of each functional category: user, department, project, and job. These shares indicate the proportion of the tickets for a category to which each job associated with a member of the category is entitled. For example, user `davidson` has 200 shares, and user `donlee` has 100. A job submitted by `davidson` is entitled to twice as many user-functional-tickets as a job submitted by `donlee`.

The functional tickets that are allotted to each category are shared among all the jobs that are associated with a particular category.

Configuring the Functional Share Policy With QMON

At the bottom of the QMON Policy Configuration dialog box, click Functional Policy. The Functional Policy dialog box appears.

Function Category List

Select the functional category for which you are defining functional shares: user, project, department, or job.

Functional Shares Table

The table under Functional Shares is scrollable. The table displays the following information:

- A list of the members of the category currently selected from the Function Category list.
- The number of functional shares for each member of the category. Shares are used as a convenient indication of the relative importance of each member of the functional category. You can edit this field.
- The percentage of the functional share allocation for this category of functional ticket that this number of functional shares represents. This field is a feedback device and is not editable.

QMON periodically updates the information displayed in the Functional Policy dialog box. Click Refresh to force the display to refresh immediately.

To save all node changes that you make, click Apply. To close the dialog box without saving changes, click Done.

Changing Functional Configurations

Click the jagged arrow above the Functional Shares table to open a configuration dialog box.

- For User functional shares, the User Configuration dialog box appears. Use the User tab to switch to the appropriate mode for changing the configuration of Grid Engine users. See How to Configure User Objects With QMON.
- For Department functional shares, the User Configuration dialog box appears. Use the Userset tab to switch to the appropriate mode for changing the configuration of departments that are represented as usersets. The `defaultdepartment` userset automatically includes all users who are not assigned to an administrator-created department.
- For Project functional shares, the Project Configuration dialog box appears. See How to Configure Projects With QMON.
- For Job functional shares, the `Job Control` dialog box appears. See Monitoring and Controlling Jobs With QMON.

Ratio Between Sorts of Functional Tickets

To display the Ratio Between Sorts Of Functional Tickets, click the arrow at the right of the Functional Shares table.

User [%], Department [%], Project [%], and Job [%] always add up to 100%.

When you change any of the sliders, all other unlocked sliders change to compensate for the change.

When a lock is open, the slider that it guards can change freely. The slider can change either because it is moved or because the moving of another slider causes this slider to change. When a lock is closed, the slider that it guards cannot change. If four locks are closed and one lock is open, no sliders can change.

- User slider – Indicates the percentage of the total functional tickets to allocate to the users category.
- Departments slider – Indicates the percentage of the total functional tickets to allocate to the departments category. The `defaultdepartment` userset automatically includes all users who are not assigned to an administrator-created department.
- Project slider – Indicates the percentage of the total functional tickets to allocate to the projects category.
- Job slider – Indicates the percentage of the total functional tickets to allocate to the jobs category.

How to Create User-Based, Project-Based, and Department-Based Functional Scheduling With QMON

Use this setup to create a certain share assignment of all the resources in the cluster to different users, projects, or departments. First-come, first-served scheduling is used among jobs of the same user, project, or department.

1. In the Scheduler Configuration dialog box, select the Share Functional Tickets check box.
   See Sharing Functional Ticket Shares, and the `sched_conf(5)` man page.

2. Specify the number of functional tickets (for example, 1000000) in the scheduler configuration.
   See How to Configure Policy-Based Resource Management With QMON, and the `sched_conf(5)` man page.

3. Add scheduling-relevant items:
   - Add one user for each scheduling-relevant user. See How to Configure User Objects With QMON, and the `user(5)` man page.
   - Add one project for each scheduling-relevant project. See How to Configure Projects With QMON, and the `project(5)` man page.
   - Add each scheduling-relevant department.
4. Assign functional shares to each user, project, or department.
   See How to Configure User Access Lists With QMON, and the `access_list(5)` man page.
   Assign the shares as a percentage of the whole. Examples follow:
   - For users:
     - UserA (10)
     - UserB (20)
     - UserC (20)
     - UserD (20)
   - For projects:
     - ProjectA (55)
     - ProjectB (45)
   - For departments:
     - DepartmentA (90)
     - DepartmentB (5)
     - DepartmentC (5)

How to Configure the Override Policy From the Command Line

To configure the override policy from the command line, use the `qconf` command with the following arguments:

- To modify the user category, type the following command:

```
qconf -muser
```

The `-muser` option modifies the `oticket` parameter of the user entry file. See the `user(5)` man page for information about the user entry file.

- To modify the department category, type the following command:

```
qconf -mu
```

The `-mu` option modifies the `oticket` parameter of the access list file. See the `access_list(5)` man page for information about the access list file, which is used to represent departments.

- To modify the project category, type the following command:

```
qconf -mprj
```

The `-mprj` option modifies the `oticket` parameter of the project entry file. See the `project(5)` man page for information about the project entry file.

To change the number of override tickets for the specified job, use the `qalter -ot` *`override_tickets`* command.

How to Configure the Share-Based Policy From the Command Line

Note
Use QMON to configure the share tree policy, because a hierarchical tree is well-suited for graphical display and for editing. However, if you need to integrate share tree modifications in shell scripts, for example, you can use the `qconf` command and its options.

To configure the share-based policy from the command line, use the `qconf` command with the following arguments:

- To display the share tree configuration, type the following command:

```
qconf -sstree
```

- To add a new share tree, type the following command:

```
qconf -astree
```

- To modify an existing share tree, type the following command:

```
qconf -mstree
```

- To delete a share tree, type the following command:

```
qconf -dstree
```

See the `qconf(1)` man page for details about these options. The `share_tree(5)` man page contains a description of the format of the share tree configuration.

- The -astnode, -mstnode, -dstnode, and -sstnode options do not address the entire share tree, but only a single node. The node is referenced as path through all parent nodes down the share tree, similar to a directory path. The options enable you to add, modify, delete, and display a node. The information contained in a node includes its name and the attached shares.
- The weighting of the usage parameters CPU, memory, and I/O are contained in the scheduler configuration as usage_weight. The weighting of the half-life is contained in the scheduler configuration as halftime. The compensation factor is contained in the scheduler configuration as compensation_factor. You can access the scheduler configuration from the command line by using the -msconf and the -ssconf options of qconf. See the sched_conf(5) man page for details about the format.

How to Configure the Functional Share Policy From the Command Line

To configure the functional share policy from the command line, use the qconf command with the following arguments:

- To configure the user category, type the following command:

  ```
  qconf -muser
  ```

  The -muser option modifies the fshare parameter of the user entry file. See the user(5) man page for information about the user entry file.

- To modify the department category, type the following command:

  ```
  qconf -mu
  ```

  The -mu option modifies the fshare parameter of the access list file. See the access_list(5) man page for information about the access list file, which is used to represent departments.

- To modify the project category, type the following command:

  ```
  qconf -mprj
  ```

  The -mprj option modifies the fshare parameter of the project entry file. See the project(5) man page for information about the project entry file.

The weighting between different categories is defined in the scheduler configuration sched_conf and can be changed using qconf -msconf. The parameters to change are weight_user, weight_department, weight_project, and weight_job. The parameter values range between 0 and 1, and the total sum of parameters must add up to 1.

To assign functional shares to jobs, use the -js job_share option with the qsub, qsh, qrsh, qlogin, and qalter commands. The -js job_share option defines or redefines the job share of the job relative to other jobs. job_share is an unsigned integer value. The default job_share value for jobs is 0.

Configuring the Override Policy

Override scheduling enables a Grid Engine system manager or operator to dynamically adjust the relative importance of one job or of all jobs that are associated with a user, a department, or a project. This adjustment adds tickets to the specified job, user, department, or project. By adding override tickets, override scheduling increases the total number of tickets that a user, department, project, or job has. As a result, the overall share of resources is increased.

The addition of override tickets also increases the total number of tickets in the system. These additional tickets deflate the value of every job's tickets.

You can use override tickets for the following two purposes:

- To temporarily override the share-based policy or the functional policy without having to change the configuration of these policies.
- To establish resource entitlement levels with an associated fixed amount of tickets. The establishment of entitlement levels is appropriate for scenarios like high, medium, or low priority classes.

Override tickets that are assigned directly to a job go away when the job finishes. All other tickets are inflated back to their original value. Override tickets that are assigned to users, departments, projects, and jobs remain in effect until the administrator explicitly removes the tickets.

The Policy Configuration dialog box displays the current number of override tickets that are active in the system.

> Note
> Override entries remain in the Override dialog box. These entries can influence subsequent work if they are not explicitly deleted by the administrator when they are no longer needed.

Configuring the Override Policy With QMON

At the bottom of the Policy Configuration dialog box, click Override Policy. The Override Policy dialog box appears.

Override Category List

Select the category for which you are defining override tickets: user, project, department, or job.

Override Table

The override table is scrollable. It displays the following information:

- A list of the members of the category for which you are defining tickets. The categories are user, project, department, and job.
- The number of override tickets for each member of the category. This field is editable.

QMON periodically updates the information that is displayed in the Override Policy dialog box. Click Refresh to force the display to refresh immediately.

To save all override changes that you make, click Apply. To close the dialog box without saving changes, click Done.

Changing Override Configurations

Click the jagged arrow above the override table to open a configuration dialog box.

- For User override tickets, the User Configuration dialog box appears. Use the User tab to switch to the appropriate mode for changing the configuration of Grid Engine users. See How to Configure User Objects With QMON for more details.
- For Department override tickets, the User Configuration dialog box appears. The `defaultdepartment` userset automatically includes all users who are not assigned to an administrator-created department. Use the Userset tab to switch to the appropriate mode for changing the configuration of departments that are represented as usersets.
- For Project override tickets, the Project Configuration dialog box appears. See How to Configure Projects With QMON for more details.
- For Job override tickets, the Job Control dialog box appears. See Monitoring and Controlling Jobs With QMON.

Managing Resource Quotas

Resource Quota Overview

To prevent users from consuming all available resources, the Grid Engine software supports complex attributes that you can configure on a global, queue or host layer. While this layered resource management approach is powerful, the approach leaves gaps that become particularly important in large installations that consist of many different custom resources, user groups, and projects. The resource quota feature closes this gap by enabling you to manage these enterprise environments to the extent that you can control which project or department must abdicate when single bottleneck resources run out.

The resource quota feature enables you to apply limits to several kinds of resources and resource consumers, to all jobs in the cluster, and to combinations of consumers. In this context, resources are any defined complex attribute known by the Sun Grid Engine configuration. For more information about complex attributes, see the `complex(5)` man page. Resources can be `slots`, `arch`, `mem_total`, `num_proc`, `swap_total`, built-in resources, or any custom-defined resource like `compiler_license`. Resource consumers are (per) users, (per) queues, (per) hosts, (per) projects, and (per) parallel environments.

The resource quota feature provides a way for you to limit the resources that a consumer can use at any time. This limitation provides an indirect method to prioritize users, departments, and projects. To define directly the priorities by which a user should obtain a resource, use the resource urgency and share-based policies described in Configuring the Urgency Policy and Configuring the Share-Based Policy.

To limit resources through the Grid Engine software, use the `qquota` and `qconf` commands, or the QMON graphical interface. For more information, see the `qquota(1)` and `qconf(1)` man pages.

Resource Quota Sets

Resource quota sets enable you to specify the maximum resource consumption for any job requests. Once you define the resource quota sets, the scheduler uses them to select the next possible jobs to be run by watching that the quotas will not be exceeded. The ultimate result of setting resource quotas is that only those jobs that do not exceed their resource quotas will be scheduled and run.

A resource quota set defines a maximum resource quota for a particular job request. All of the configured rule sets apply all of the time. If multiple resource quota sets are defined, the most restrictive set applies. Every resource quota set consists of one or more resource quota rules. These rules are evaluated in order, and the first rule that matches a specific request is used. A resource quota set always results in at most one effective resource quota rule for a specific request.

A resource quota set consists of the following information:

- Name – The name of the resource quota set.
- Enabled – A boolean value that indicates whether the resource set should be considered in scheduling decisions. If enabled is true, the resource quota set is active and will be considered for scheduling decisions. The default value is false.
- Description – An optional field that contains an arbitrary string that describes the set. The default value is NONE.
- Limit rule – Every resource quota set needs at least one limit rule, which is contained in the limit field. For example, the following limit rule limits all users together to 10 slots: `limit users * to slots=10`. The limit rule contains the following information:
  - Name – An optional name for the rule. If used, the name must be unique within the resource quota set.
  - Filter scope – The filter scope identifies the list of resource consumers to which the quota applies. A resource consumer contains a keyword followed by a comma-separated list of consumers. Use the following keywords: `users`, `projects`, `queues` (cluster queues), `hosts`, or `pes` (parallel environments).
    The following example shows a resource consumer and a filter scope. The defined filter scope limits `user1` and `user2` to the maximum number of the configured limit independently from the host.

    ```
    users {user1, user2}
    users {user1, user2} hosts *
    ```

    To include an expandable list in the resource quota definition, use curly braces around the resource consumer list, as shown in the resource consumer and filter scope example above.
    To exclude one of a specific resource type from a list, use `!` (the exclamation point, sometimes referred to as the "not" symbol).

  - Limit – An attribute-value pair that defines the actual limit for the resource. For example, `virtual_free=2G`. You can also combine pairs into a comma-separated list of attribute-value pairs. For example, `virtual_free=2G,swap_free=1.5G`.

Example – Sample Resource Quota Set

The following example resource quota set restricts `user1` and `user2` to 2 Gbytes of free virtual space on each host in the host group `lx_hosts`.

```
     {
        name         max_virtual_free_on_lx_hosts
        description  "resource quota for virtual_free restriction"
        enabled      true
        limit        users {user1,user2} hosts {@lx_host} to virtual_free=2g
     }
```

Static and Dynamic Resource Quotas

Resource quota rules always define a maximum value of a resource that can be used. In most cases, these values are static and equal for all matching filter scopes. Although you could define several different rules to apply to different scopes, you would then have several rules that are nearly identical. Instead of duplicating rules, you can instead define a dynamic limit.

A dynamic limit uses an algebraic expression to derive the rule limit value. The algebraic formula can reference a complex attribute whose value is used to calculate the resulting limit.

Example – Dynamic Limit Example

The following example illustrates the use of dynamic limits. Users are allowed to use five slots per CPU on all Linux hosts.

```
limit hosts {@linux_hosts} to slots=$num_proc*5
```

The value of `num_proc` is the number of processors on the host. The limit is calculated by the formula `$num_proc*5`, and can be different on each host. Expanding the example above, you could have the following resulting limits:

- On a host with two CPUs, users can use ten slots to run jobs.
- On a host with one CPU, users can use only five slots to run jobs.

Instead of `num_proc`, you could use any other complex attribute known for a host as either a load value or a consumable resource.

Performance Considerations and Command Line Examples

Topic	Description
Performance Considerations	Learn how to create efficient rule sets to improve performance.

Resource Management Tasks

Task	User Interface	Description
How to Configure Resource Quotas	CLI or QMON	Learn how to display, add, modify, and delete resource quotas.
How to Monitor Resource Quota Utilization	CLI	Learn how to monitory resource quota utilization from the command line.

Performance Considerations

Efficient Rule Sets

To provide the most efficient processing of jobs and resources in queues, put the most restrictive rule at the first position of a rule set. Following this convention helps the Sun Grid Engine scheduler to restrict the amount of suited queue instances in a particularly efficient manner, because the first rule is never shadowed by any subsequent rule in the same rule set and thus always stands for itself.

To illustrate this rule, consider an environment similar to the following:

- Four queues named Q001-Q004.
- Four managed resources named F001-F004.
- Jobs that require a specific managed resource, such as F001, are constrained to run in the associated queue, such as Q001.
- Jobs are submitted into one of five projects named P001-P005.

In such an environment, you might define a single rule set as follows:

```
{
      name         30_for_each_project
      description  "not more than 30 per project"
      enabled      TRUE
      limit projects {*} queues Q001 to F001=30
      limit projects {*} queues Q002 to F002=30
      limit projects {*} queues Q003 to F003=30
      limit projects {*} queues Q004 to F004=30
      limit to F001=0,F002=0,F003=0,F004=0
   }
```

The single rule set limits the utilization of each managed resource to 30 for each project and constrains the jobs in eligible queues at the same time. This will work fine, but in a larger cluster with many hosts, the single rule set would become the cause of slow job dispatching.

To help the Sun Grid Engine scheduler to foreclose as many queue instances as possible during matchmaking, use four separate rule sets.

```
{
      name         30_for_each_project_in_Q001
      description  "not more than 30 per project of F001 in Q001"
      enabled      TRUE
      limit queues !Q001 to F001=0
      limit projects {*} queues Q001 to F001=30
   }

   {
      name         30_for_each_project_in_Q002
      description  "not more than 30 per project of F002 in Q002"
      enabled      TRUE
      limit queues !Q002 to F002=0
      limit projects {*} queues Q002 to F002=30
   }

   {
      name         30_for_each_project_in_Q003
      description  "not more than 30 per project of F003 in Q003"
      enabled      TRUE
      limit queues !Q003 to F003=0
      limit projects {*} queues Q003 to F003=30
   }

   {
      name         30_for_each_project_in_Q004
      description  "not more than 30 per project of F004 in Q004"
      enabled      TRUE
      limit queues !Q004 to F004=0
      limit projects {*} queues Q004 to F004=30
   }
```

These four rule sets constrain the very same per project resource quotas as the single rule set. However, the four rule sets can be processed much more efficiently due to unsuitable queue instances being shielded first. Consolidating these shields into a single resource quota set would not be doable in this case.

Note
The purpose of the sample above is not to recommend one cluster queue per resource. In fact, the opposite is true, because fewer queues finally always enable fewer, more powerful shields as shown here:

```
  {
      name         30_for_each_project_in_Q001
      description  "not more than 30 per project of F001/F002 in Q001"
      enabled      TRUE
      limit queues !Q001 to F001=0,F002=0
      limit projects {*} queues Q001 to F001=30,F002=30
   }

   {
      name         30_for_each_project_in_Q002
      description  "not more than 30 per project of F003/F004 in Q002"
      enabled      TRUE
      limit queues !Q002 to F003=0,F004=0
      limit projects {*} queues Q002 to F003=30,F004=30   }
```

In this example, the queues are consolidated from Q001-Q004 down to Q001-Q002. However, this actually increases overall cluster utilization and throughput.

Example 2 – Rule Set

The rule set defines the following limit:

- All users together should never take more than 20 slots.
- All users should take at most 5 slots on all Linux hosts.
- Every user is restricted to 1 slot per Linux host except `MyUser` who is restricted to 2 slots, and all other slots on hosts are restricted to 0.
- The host group `@linux` includes `host1` and `host2`.

To configure the rule set, use one of the following forms of the `qconf` command:

- `qconf -arqs` rule-set-name for each rule set
- `qconf -arqs` to run all rule sets at once

```
{
 name maxujobs
 limit users * to slots=20
}

{
 name max_linux
 limit users * hosts @linux to slots=5
}

{
 name max_per_host
 limit users MyUser hosts {@linux} to slots=2
 limit users {*} hosts {@linux} to slots=1
 limit users * hosts * to slots=0
}
```

After jobs are submitted for different users, the `qstat` command shows output similar to the example shown below:

```
$ qstat
job-ID  prior   name       user       state submit/start at     queue       slots ja-task-ID
---------------------------------------------------------------------------------------------
     27 0.55500 Sleeper    MyUser     r     02/21/2006 15:53:10 all.q@host1   1
     29 0.55500 Sleeper    MyUser     r     02/21/2006 15:53:10 all.q@host1   1
     30 0.55500 Sleeper    MyUser     r     02/21/2006 15:53:10 all.q@host2   1
     26 0.55500 Sleeper    MyUser     r     02/21/2006 15:53:10 all.q@host2   1
     28 0.55500 Sleeper    user1      r     02/21/2006 15:53:10 all.q@host2   1
```

How to Configure Resource Quotas From the Command Line

To configure resource quotas from the command line, use the following arguments for the `qconf` command:

- To display a list of defined resource quota sets, type the following command:

  ```
  $ qconf -srqsl
  ```

- To display a specific resource quota set, type the following command:

  ```
  $ qconf -srqs [name_list]
  ```

- To add a resource quota set by invoking a text editor, use the following command:

  ```
  qconf -arqs [name]
  ```

- To add a resource quota set from file, type the following command:

  ```
  qconf -Arqs <filename>
  ```

- To modify a resource quota set, type the following command:

  ```
  qconf -mrqs [name]
  ```

  To modify a resource set from file, type the following command:

  ```
  qconf -Mrqs <filename>
  ```

  Note
  If you use the `-mrqs` or `-Mrqs` option without a name, the new rule set replaces all the currently configured rule sets.

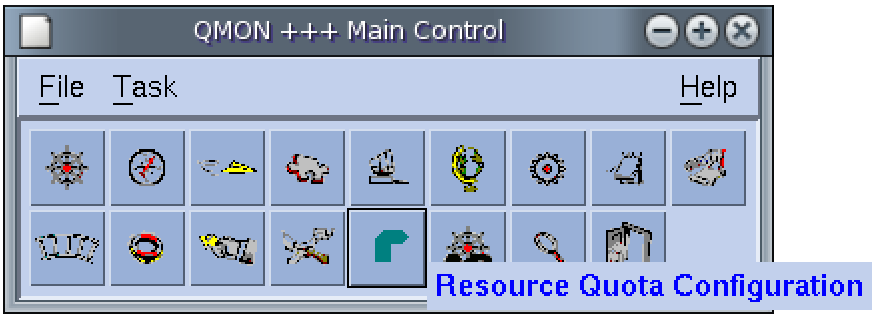
- To delete a resource quota set, type the following command:

  ```
  $ qconf -drqs [name_list]
  ```

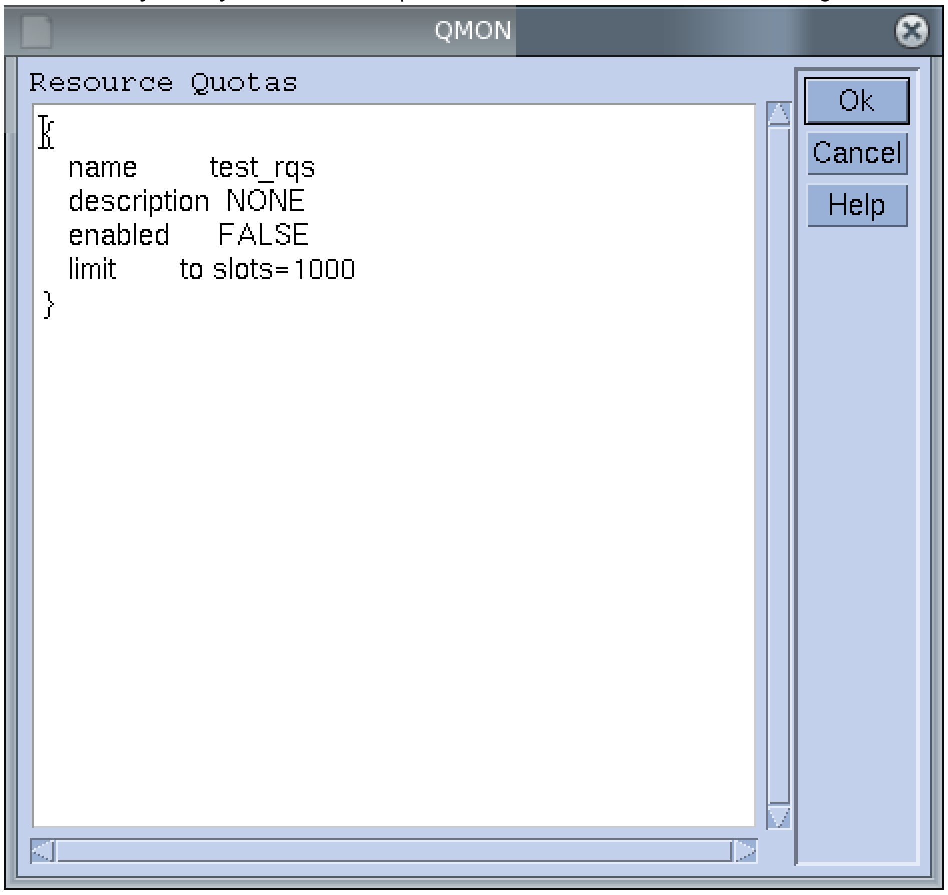
For more information about `qconf`, see the `qconf(1)` man page.

How to Configure Resource Quotas With QMON

1. In the QMON Main Control window, click the Resource Quota Configuration button.



2. Type the Resource Quota information in the text field.
   Use the same syntax as you would for the `qconf` command, as illustrated in the following screen example.



```
{
   name        test_rqs
   description  NONE
   enabled      FALSE
   limit        to slots=1000
}
```

How to Monitor Resource Quota Utilization From the Command Line

To monitor resource quota utilization from the command line, use the following command:

```
qquota
```

The `qquota` command includes several options that you can use to limit the information to a specific host, cluster queue, project, parallel environments, resource, or user. If you use no options, `qquota` displays information about resource sets that apply to the user name from which you invoke the command. For more information, see the `qquota(1)` man page.

The `qquota` command displays information about the current Sun Grid Engine resource quotas. The `qquota` command lists each resource quota that is being used at least once or that defines a static limit. For each applicable resource quota, `qquota` displays the following information:

- Resource quota rule – The name of the rule set name and the name or number of the rule
- Limit – The resource name, the number of available items for that resource, and the number of used items for that resource
- Filter – The effective resource quota set filter, which results from applying the filter scope explained in About Resource Quota Sets

Examples – Sample **qquota** Commands

The following example shows information about the resource quota sets that apply to user `user1`

```
$ qquota -u user1
resource quota    limit                filter
--------------------------------------------------------------------------------
maxujobs/1         slots=5/20           -
max_linux/1        slots=5/5            hosts @linux
max_per_host/1     slots=1/2            users user1 hosts host2
```

The following example shows information about the resource quota sets that apply to user `MyUser`:

```
$ qquota # as user MyUser
resource quota rule    limit                filter
--------------------------------------------------------------------------------
maxujobs/1         slots=5/20           -
max_linux/1        slots=5/5            hosts @linux
max_per_host/1     slots=2/2            users MyUser hosts host2
max_per_host/1     slots=2/2            users MyUser hosts host1

$ qquota -h host2 # as user MyUser
resource quota    limit                filter
--------------------------------------------------------------------------------
maxujobs/1         slots=5/20           -
max_linux/1        slots=5/5            hosts @linux
max_per_host/1     slots=2/2            users MyUser hosts host2

$ qquota -u user1
resource quota    limit                filter
--------------------------------------------------------------------------------
maxujobs/1         slots=5/20           -
max_linux/1        slots=5/5            hosts @linux
max_per_host/1     slots=1/2            users user1 hosts host2

$ qquota -u *
resource quota    limit                filter
--------------------------------------------------------------------------------
maxujobs/1         slots=5/20           -
max_linux/1        slots=5/5            hosts @linux
max_per_host/1     slots=2/2            users MyUser hosts host1
max_per_host/1     slots=2/2            users MyUser hosts host2
max_per_host/1     slots=1/2            users user1 hosts host2
```

Managing the Scheduler

The scheduler, the scheduling function for the master daemon, is responsible for assigning the highest priority pending jobs to the best available cluster resources. After a user submits a job, the master daemon assigns it to a pending list. The master daemon then periodically triggers scheduling runs in an attempt to assign these jobs to available cluster resources.

> Previous to the release of Sun Grid Engine 6.2, the scheduler was a separate daemon that communicated directly with the master daemon. Now, "the scheduler" is simply the scheduling function of the master deamon.

This section contains information about scheduling Grid Engine system policy implementation through the scheduler. For information about Grid Engine policies, see Managing Policies.

Topic	Description
How Jobs Are Scheduled	Learn how the Sun Grid Engine scheduling function schedules jobs.
Scheduling Strategies	Learn what strategies you can use to maximize your resources and accomplish your goals.
Configuring the Scheduler	Learn how to configure the scheduler.
Configuring Exclusive Scheduling	Learn how to request exclusive access to a host.

How to Monitor the Scheduler From the Command Line

To record detailed information on scheduler decisions for all jobs, type the following:

```
qconf -tsm
```

This command forces the scheduler to write trace output to the `$SGE_ROOT/$SGE_CELL/common/schedd_runlog` file.

For more information, see the `qconf(1)` man page.

Configuring the Scheduler

To better understand how the scheduling function works, check out the following topics:

- Default Scheduling
- Scheduling Alternatives
- Changing the Scheduling Algorithm
- Scaling System Load
- Selecting Queue by Sequence Number
- Selecting Queue by Share
- Restricting the Number of Jobs per User or Group
- Scheduling Tasks

Refer to How to Configure Policy-Based Resource Management With QMON for details on the scheduling administration of resource-sharing policies of the Grid Engine system. The following sections focus on administering the scheduler configuration `sched_conf(5)` and related issues.

Default Scheduling

The default scheduling is a first-in, first-out policy. In other words, the first job that is submitted is the first job the scheduler examines to dispatch it to a queue. If the first job in the list of pending jobs finds a queue that is suitable and available, that job is started first. A job ranked behind the first job can be started first only if the first job fails to find a suitable free resource.

The default strategy is to select queue instances on the least-loaded host, provided that the queues deliver suitable service for the job's resource requirements. If several suitable queues share the same load, the queue to be selected is unpredictable.

Scheduling Alternatives

You can modify the job scheduling and queue selection strategy in various ways:

- Changing the scheduling algorithm

- Scaling system load
- Selecting queue by sequence number
- Selecting queue by share
- Restricting the number of jobs per user or per group

The following sections explore these alternatives in detail.

Changing the Scheduling Algorithm

The scheduler configuration parameter `algorithm` provides a selection for the scheduling algorithm in use. See the `sched_conf`(5) man page for further information. Currently, `default` is the only allowed setting.

Scaling System Load

To select the queue to run a job, the Grid Engine system uses the system load information on the machines that host queue instances. This queue selection scheme builds up a load-balanced situation, thus guaranteeing better use of the available resources in a cluster.

However, the system load may not always tell the truth. For example, if a multi-CPU machine is compared to a single CPU system, the multiprocessor system usually reports higher load figures, because it probably runs more processes. The system load is a measurement strongly influenced by the number of processes trying to get CPU access. But multi-CPU systems are capable of satisfying a much higher load than single-CPU machines. This problem is addressed by processor-number-adjusted sets of load values that are reported by default by `sge_execd`. Use these load parameters instead of the raw load values to avoid the problem described earlier. See [Load Parameters] and the `$SGE_ROOT/doc/load_parameters.asc` file for details.

Another example of potentially improper interpretation of load values is when systems have marked differences in their performance potential or in their price performance ratio. In both cases, equal load values do not mean that arbitrary hosts can be selected to run a job. In this situation, the administrator should define load scaling factors for the relevant execution hosts and load parameters. See [Configuring Execution Hosts With QMON], and related sections.

> Note
> The scaled load parameters are also compared against the load threshold lists load-thresholds and migr-load-thresholds. See the `queue_conf`(5) man page for details.

Another problem associated with load parameters is the need for an application-dependent and site-dependent interpretation of the values and their relative importance. The CPU load might be dominant for a certain type of application that is common at a particular site. By contrast, the memory load might be more important for another site and for the application profile to which the site's compute cluster is dedicated. To address this problem, the Grid Engine system enables the administrator to specify a load formula in the scheduler configuration file `sched_conf`. See the `sched_conf`(5) man page for more details. Site-specific information on resource usage and capacity planning can be taken into account by using site-defined load parameters and consumable resources in the load formula. See the sections [Adding Site-Specific Load Parameters] and [Consumable Resources].

Finally, the time dependency of load parameters must be taken into account. The load that is imposed by the jobs that are running on a system varies in time. Often the load, for example, the CPU load, requires some amount of time to be reported in the appropriate quantity by the operating system. If a job recently started, the reported load might not provide an accurate representation of the load that the job has imposed on that host. The reported load adapts to the real load over time. But the period of time in which the reported load is too low might lead to an over-subscription of that host. The Grid Engine system enables the administrator to specify load adjustment factors that are used in the scheduler to compensate for this problem. See the `sched_conf`(5) man page for detailed information on how to set these load adjustment factors.

Load adjustments are used to virtually increase the measured load after a job is dispatched. In the case of oversubscribed machines, this helps to align with load thresholds. If you do not need load adjustments, you should turn them off. Load adjustments impose additional work on the scheduler in connection with sorting hosts and load thresholds verification.

To disable load adjustments, on the Load Adjustment tab of the Scheduler Configuration dialog box, set the Decay Time to zero, and delete all load adjustment values in the table. See How to Modify the Scheduler Configuration With QMON.

Selecting Queue by Sequence Number

Another way to change the default scheme for queue selection is to set the global cluster configuration parameter `queue_sort_method` to `seq_no` instead of to the default `load`. In this case, the system load is no longer used as the primary method to select queues. Instead, the sequence numbers that are assigned to the queues by the queue configuration parameter seq_no define a fixed order for queue selection. The queues must be suitable for the considered job, and they must be available. See the queue_conf(5) and sched_conf(5) man pages for more details.

This queue selection policy is useful if the machines that offer batch services at your site are ranked in a monotonous price per job order. For example, a job running on Machine A costs 1-unit of money. The same job costs 10-units on Machine B. And on Machine C the job costs 100-units. Thus the preferred scheduling policy is to first fill up Host A and then to use Host B. Host C is used only if no alternative remains.

Note
If you have changed the method of queue selection to seq_no, and the considered queues all share the same sequence number, queues are selected by the default load.

Selecting Queue by Share

The goal of this method is to place jobs so as to attempt to meet the targeted share of global system resources for each job. This method takes into account the resource capability represented by each host in relation to all the system resources. This method tries to balance the percentage of tickets for each host (that is, the sum of tickets for all jobs running on a host) with the percentage of the resource capability that particular host represents for the system. See [Configuring Execution Hosts With QMON] for instructions on how to define the capacity of a host.

The host's load, although of secondary importance, is also taken into account in the sorting. Choose this sorting method for a site that uses the share-tree policy.

Restricting the Number of Jobs per User or Group

The administrator can assign an upper limit to the number of jobs that any user or any UNIX group can run at any time. To enforce this feature, do one of the following:

- Set maxujobs as described in the sched_conf(5) man page.
- On the General Parameters tab of the Scheduler Configuration dialog box, use the Max Jobs/User field to set the maximum number of jobs a user or user group can run concurrently.

Scheduling Tasks

Task	User Interface	Description
How to Monitor the Scheduler	CLI	
How to Modify the Scheduler Configuration	CLI or QMON	Learn how to show and modify the current scheduler configuration.

How to Modify the Scheduler Configuration From the Command Line

To configure the scheduler from the command line, use the following arguments for the qconf command:

- To show the the current scheduler configuration, type the following command:

```
qconf -ssconf
```

- To modify the current scheduler configuration using an editor, type the following command:

```
qconf -msconf
```

You can configure the following parameters in the scheduler configuration:

    - `default_duration` – This value specifies the assumed runtime for jobs with neither `-l h_rt=...` nor `-l s_rt=`... and is used for advance and resource reservations.
    - `flush_finish_sec` – Timespan (in seconds) after which the scheduler is informed of finished jobs.
    - `flush_submit_sec` – Timespan (in seconds) after which the scheduler is informed of new jobs.
    - `max_reservation` – The maximum number of resource reservations considered within a schedule interval.
    - `params` – The following additional parameter is available:
        - DURATION_OFFSET – This parameter specifies the offset between the net job run times and the total time until resources become available again used when computing advance or resource reservations. The default is 60 seconds. If the offset is too low, reservations (see `max_reservation`) can be delayed repeatedly due to an overly optimistic job circulation time.

- To shut down the scheduler, type the following command:

```
qconf -kt scheduler
```

- To start up the scheduler, type the following command:

```
qconf -at scheduler
```

Example – Modifying the Schedule Interval

The following example modifies the schedule interval of the scheduler:

```
#!/bin/ksh
# sched_int.sh: modify the schedule interval
# usage: sched_int.sh <n>, where <n> is
# the new interval, in seconds. n < 60

TMPFILE=/tmp/sched_int.$$
if [ $MOD_SGE_SCHED_INT ]; then
     grep -v schedule_interval $1 > $TMPFILE
     echo "schedule_interval 0:0:$MOD_SGE_SCHED_INT" >> $TMPFILE
# sleep to ensure modification time changes
     sleep 1
     mv $TMPFILE $1
else
     export EDITOR=$0
     export MOD_SGE_SCHED_INT=$1
     qconf -msconf
fi
```

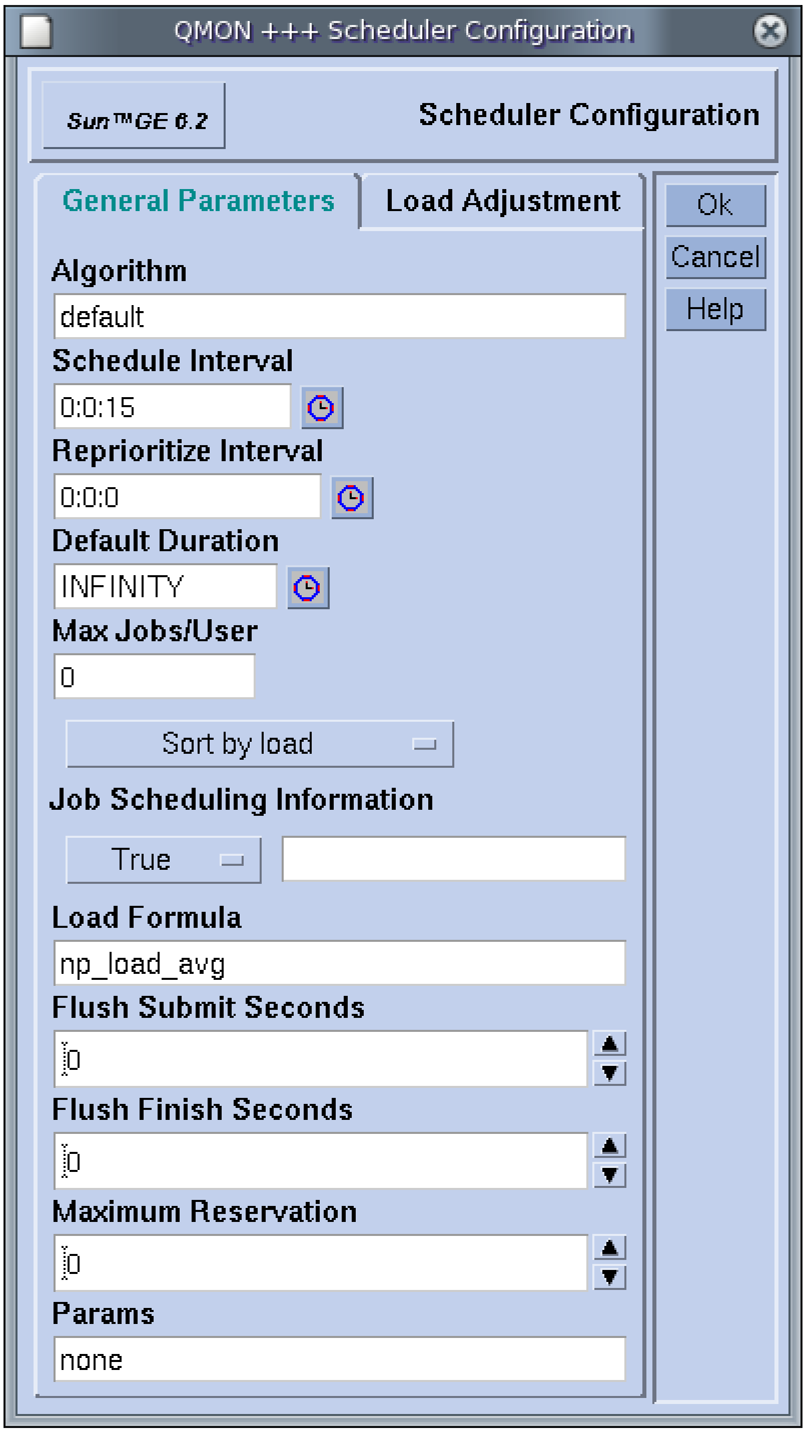
This script modifies the EDITOR environment to point to itself. The script then calls the `qconf -msconf` command. This second nested invocation of the script modifies the temporary file specified by the first argument and then exits. The Grid Engine system automatically reads in the changes, and the first invocation of the script terminates.

For more information on how to configure the cluster from file or how to modify many objects at once, see Using qconf.

How to Modify the Scheduler Configuration With QMON

1. Click the Scheduler Configuration button on the QMON Main Control window.
   The Scheduler Configuration dialog box appears.

2. From the Scheduler dialog box, you can choose from the following tasks:
    - To modify general scheduling parameters, click the `General Parameters` tab.
      The General Parameters tab looks like the following figure.


QMON +++ Scheduler Configuration

Sun™GE 6.2

Scheduler Configuration

General Parameters | Load Adjustment

Algorithm: default

Schedule Interval: 0:0:15

Reprioritize Interval: 0:0:0

Default Duration: INFINITY

Max Jobs/User: 0

Sort by load

Job Scheduling Information: True

Load Formula: np_load_avg

Flush Submit Seconds: 0

Flush Finish Seconds: 0

Maximum Reservation: 0

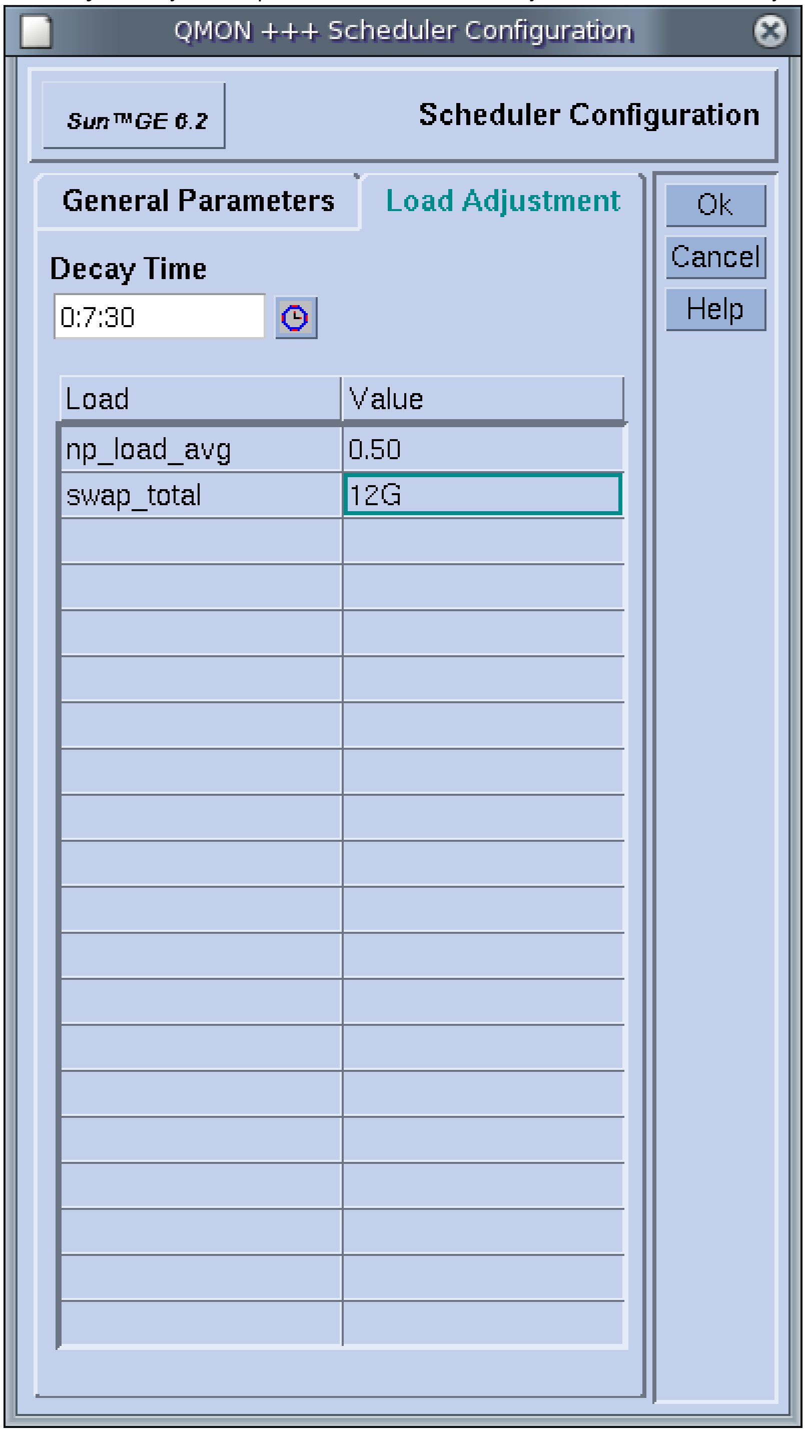
Params: none

Use the `General Parameters` tab to set the following parameters:

- Schedule Interval – The regular time interval between scheduler runs.
- Reprioritize Interval – The regular time interval to reprioritize jobs on the execution hosts, based on the current ticket amount for running jobs. To turn reprioritizing off, set this parameter to zero.
- Max Jobs/User – The maximum number of jobs that are allowed to run concurrently per user and per UNIX group.
- Sort by – The queue sorting scheme, either sorting by load or sorting by sequence number.
- Job Scheduling Information - Job scheduling information is generated in every scheduling interval and is accessible through `qstat -j` <job_id>. Note: This option is deprecated and has been replaced by `qalter -w p` <job_id>.
- Load Formula – The load formula to use to sort hosts and queues.
- Flush Submit Seconds – The number of seconds that the scheduler waits after a job is submitted before the scheduler is triggered. To disable the flush after a job is submitted, set this parameter to zero.
- Flush Finish Seconds – The number of seconds that the scheduler waits after a job has finished before the scheduler is triggered. To disable the flush after a job has finished, set this parameter to zero.
- Maximum Reservation – The maximum number of resource reservations that can be scheduled within a scheduling interval.
- Params – Use this setting to specify additional parameters to pass to the scheduler. Params can be `PROFILE` or `MONITOR`. If you set `PROFILE=1`, the scheduler will log profiling information that summarizes each scheduling run. If you set `MONITOR=1`, the scheduler will record information for each scheduling run in the file

$SGE_ROOT/$SGE_CELL/common/schedule.
By default, the Grid Engine system schedules job runs in a fixed schedule interval. You can use the `Flush Submit Seconds` and `Flush Finish Seconds` parameters to configure immediate scheduling.

- To modify load adjustment parameters, click the Load Adjustment tab. The Load Adjustment tab looks like the following figure:



Decay Time
0:7:30

Load	Value
np_load_avg	0.50
swap_total	12G

The Load Adjustment tab displays following parameters:
- Decay Time – The decay time for the load adjustment.
- A table of load adjustment values listing all load and consumable attributes for which an adjustment value is currently defined. From the Load Adjustment tab, you can choose from the following tasks:
- To add load values to the list, click the Load or the Value column heading. A selection list appears with all resource attributes that are attached to the hosts.
- To add a resource attribute in the Load column of the Consumable/Fixed Attributes table, use the Attribute Selection dialog box to select one of the attributes, and then click OK.
- To modify an existing value, double-click the Value field.
- To delete a resource attribute, select it, and then press `Control-D` or click mouse button 3. A dialog box asks you to confirm the deletion.

See the `sched_conf(5)` man page for more details about the scheduler configuration.

Monitoring and Controlling SMF Services

For information on service names and changed behavior, see Installing SMF Services.

Monitoring SMF Services

You can use `svcs` command to query services present on your system.

```
% svcs
STATE          STIME    FMRI
legacy_run     16:03:54 lrc:/etc/rcS_d/S29wrsmcfg
legacy_run     16:04:11 lrc:/etc/rc2_d/S47pppd
online         16:03:44 svc:/network/loopback:default
online         16:03:47 svc:/system/filesystem/root:default
online         16:03:47 svc:/system/scheduler:default
online         16:03:47 svc:/system/boot-archive:default
online         16:03:48 svc:/system/filesystem/usr:default
online         16:03:49 svc:/network/physical:default
online         16:03:49 svc:/milestone/network:default
...
```

To query Grid Engine services, you can use mask "*sge*":

```
% svcs "*sge*"
online         16:03:47 svc:/application/sge/qmaster:prod_cluster
online         16:03:47 svc:/application/sge/qmaster:test_cluster
online         16:03:47 svc:/application/sge/execd:prod_cluster
online         16:03:47 svc:/application/sge/execd:test_cluster
```

To get a more information about single service use svcs -l <FMRI>:

```
% svcs -l qmaster:prod_cluster
fmri         svc:/application/sge/qmaster:prod_cluster
name         Sun Grid Engine - QMaster service
enabled      true
state        online
next_state   none
state_time   Sun May 19 21:28:39 2008
logfile      /var/svc/log/application-sge-qmaster:prod_cluster.log
restarter    svc:/system/svc/restarter:default
contract_id  4912
dependency   require_all/none svc:/milestone/network (online)
dependency   optional_all/none svc:/system/filesystem/autofs (online)
```

You can see that each SMF service has an additional log file. This log file contains information related to the SMF framework and can contain many useful information when service fails.

Controlling SMF services

You may use `svcadm` command to enable (start), disable (stop) or restart any SMF service. You must have appropriate permissions (solaris.smf.manage.*) to do so (typically root).

Starting qmaster service (will be started on reboot):

```
% svcadm enable qmaster:prod_cluster
```

Stopping qmaster service just for now (will be started on reboot):

```
% svcadm disable -t qmaster:prod_cluster
```

Stopping qmaster service (will NOT be started on reboot):

```
% svcadm disable qmaster:prod_cluster
```

Start qmaster service until next reboot:

```
% svcadm enable -t qmaster:prod_cluster
```

Fine-Tuning Your Environment

The Grid Engine system is a full-function, general-purpose distributed resource management tool. The scheduler component of the system supports a wide range of different compute farm scenarios. To get the maximum performance from your compute environment, you should review the features that are enabled. You should then determine which features you really need to solve your load management problem. Disabling some of these features can improve performance on the throughput of your cluster.

Scheduler Monitoring

Scheduler monitoring can help you understand why certain jobs are not being dispatched by Sun Grid Engine. However, this option consumes a lot of resources because it reports on all jobs at all times. Usually, you do not need all of this information. To request information for one job at a time, use `qalter -w p`.

If you would like to enable scheduler monitoring, set `schedd_job_info` to `true` in the scheduler configuration. For more information, see How to Modify the Scheduler Configuration With QMON and the `sched_conf(5)` man page.

Finished Jobs

In the case of array jobs, the finished job list in `qmaster` can become quite large. By switching the finished job list off, you save memory and speed up the `qstat` process, because `qstat` also fetches the finished jobs list.

To turn off the finished job list function, set `finished_jobs` to zero in the cluster configuration. See How to Modify the Scheduler Configuration With QMON and the `sge_conf(5)` man page.

Job Validation

Forced validation at job submission time can be a valuable procedure to prevent non-dispatchable jobs from forever remaining in a pending state. However, job validation can also be a time-consuming task. Job validation can be especially time-consuming in heterogeneous environments with different execution nodes and consumable resources, and in which all users have their own job profiles. In homogeneous environments with only a few different jobs, a general job validation usually can be omitted.

To disable job verification, add the `qsub` option `-w n` in the cluster-wide default requests. For more information, see How to Submit an Advanced Job With QMON and the `sge_request(5)` man page.

Load Thresholds and Suspend Thresholds

Load thresholds are needed if you deliberately oversubscribe your machines and you need to prevent excessive system load. Suspend thresholds are also used to prevent overloading the system.

Another case where you want to prevent the overloading of a node is when the execution node is still open for interactive load. Interactive load is not under the control of the Grid Engine system.

A compute farm might be more single-purpose. For example, each CPU at a compute node might be represented by only one queue slot, and no interactive load might be expected at these nodes. In such cases, you can omit `load_thresholds`.

To disable both thresholds, set `load_thresholds` to none and `suspend_thresholds` to none. See Configuring Load and Suspend Thresholds and the `queue_conf(5)` man page.

Load Adjustments

Load adjustments are used to increase the measured load after a job is dispatched. This mechanism prevents over-subscription of machines that is caused by the delay between job dispatching and the corresponding load impact. You can switch off load adjustments if you do not need them. Load adjustments impose on the scheduler some additional work in connection with sorting hosts and load thresholds verification.

To disable load adjustments, set `job_load_adjustments` to none and `load_adjustment_decay_time` to zero in the scheduler configuration. See How to Modify the Scheduler Configuration With QMON and the `sched_conf(5)` man page.

Immediate Scheduling

The default for the Grid Engine system is to start scheduling runs in a fixed schedule interval. A good feature of fixed intervals is that they limit the CPU time consumption of the `qmaster` and the scheduler. A bad feature is that fixed intervals choke the scheduler, artificially resulting in a limited throughput. Many compute farms have machines specifically dedicated to `qmaster` and the scheduler, and such setups provide no reason to choke the scheduler. See `schedule_interval` in `sched_conf`(5).

You can configure immediate scheduling by using the `flush_submit_sec` and `flush_finish_sec` parameters of the scheduler configuration. See How to Modify the Scheduler Configuration With QMON and the `sched_conf(5)` man page.

If immediate scheduling is activated, the throughput of a compute farm is limited only by the power of the machine that is hosting `sge_qmaster` and the scheduler.

Urgency Policy and Resource Reservation

The urgency policy enables you to customize job priority schemes that are resource-dependent. Such job priority schemes include the following:

- A general preference to run the largest parallel jobs first
- A preference for jobs that request particular resources in order to make use of expensive licenses

The implementing of both objectives is especially valuable if you are using resource reservation.

Using DTrace for Performance Tuning

Troubleshooting in a distributed system that spans potentially thousands of active components can challenge even the most experienced system administrator. In practice, Grid Engine administrators have no explicit mechanism for identifying and reproducing issues that lead to degraded performance in their production environments. In the Solaris ^{TM} 10 environment, you can use the DTrace utility to monitor the on-site performance of the Grid Engine master component. DTrace is a comprehensive framework for tracing dynamic events in Solaris 10 environments. For general information about DTrace, see http://www.sun.com/bigadmin/content/dtrace/ and the `dtrace` man page. For detailed information about using DTrace with Grid Engine software, view the `$SGE_ROOT/dtrace/README_dtrace.txt` file.

Tuning Performance From the Command Line Through DTrace

If you can use Solaris 10 DTrace, you can use the `$SGE_ROOT/dtrace/monitor.sh` script to monitor a Grid Engine master and look for any bottlenecks. The `monitor.sh` script supports the following options:

- `-interval` value – Specify statistics interval to use. The default is `15sec`. A larger interval results in coarser statistics, while a smaller value provides more refined results. Most useful values range from `1sec` to `24hours`.
- `-cell` cell-name – Required if `$SGE_CELL` is not default.
- `-spooling` – Display `qmaster` spooling probes in addition to statistics. This option enables you to view more specific information about a presumed spooling bottleneck.
- `-requests` – Shows incoming `qmaster` request probes. This option enables you to view more specific information to evaluate instances in which someone is flooding your `qmaster`.

Note
Any critical, error, or warning messages appear in `monitor.sh` output.

Analyzing Bottlenecks on the Grid Engine Master

To provide effective performance tuning, you must understand the bottlenecks of distributed systems. The `$SGE_ROOT/dtrace/monitor.sh` script measures throughput-relevant data of the running Grid Engine master and compiles this data into a few indices that are printed in a single-line view per interval. This view shows four main categories of information:

- Spooling – Indicates the number of operations that spooled to the `qmaster` process and the elapsed time
- Request handling – Shows the number of messages sent and received of various types, such as reports, GDI requests, and ACK messages
- Scheduling – Indicates the number of scheduling requests sent to the `schedd` process and the elapsed time
- Qmaster processing – Includes information about `qmaster/schedd` communications, `qmaster` request I/O activities, and `qmaster` lock and unlock requests

For more information, see the example below.

Sample DTrace Output for Bottleneck Analysis

The following monitoring output sample illustrates a case where a Grid Engine master bottleneck can be detected. The example shows the following information:

- For `qmaster` spooling activities:
  - `#wrt` – Number of `qmaster` write operations processed through `spool_write_object()` and `spool_delete_object()`. Almost every significant write operation goes through this function.
  - `wrt/ms` – Total time all threads spend in `spool_write_object()` in microseconds.
- For `qmaster` message processing:
  - `#rep` – Number of reports `qmaster` processed through `sge_c_report()`. Most data sent by `execd` functions to `qmaster` are reflected here.
  - `#gdi` – Number of GDI requests `qmaster` processed through `do_gdi_request()`. Almost anything sent from client commands arrives as a GDI request, although GDI requests can also come from `exexd` functions and the scheduler.
  - `#ack` – Number of ACK messages `qmaster` processed through `do_c_ack()`. High numbers of ACK messages might indicate job signaling, although ACK messages are used also for other purposes.
- For scheduling activities:
  - `#dsp` – Number of calls to `dispatch_jobs()`. Each call to `dispatch_jobs()` can be seen as a scheduling run.
  - `dsp/ms` – Total time scheduler spent in all calls to `dispatch_jobs()`.
  - `#sad` – Number of calls to `select_assign_debit()`. Each call to `select_assign_debit()` can be seen as a try of the scheduler to find an assignment or a reservation for a job.
- For `qmaster` processing:
  - `#snd` – Number of event packages that `qmaster` sends to the scheduler. If that number goes down to zero over time, something is wrong.
  - `#rcv` – Number of event packages that the scxheduler receives from `qmaster`. If that number goes down to zero over time, something is wrong.
  - `#in++` – Number of messages added to `qmaster` received messages buffer.
  - `#in--` – Number of messages removed from `qmaster` received messages buffer. If more messages are added than removed during an interval, the total of messages not yet processed is about to grow.
  - `#out++` – Number of messages added to `qmaster` send messages buffer.
  - `#out--` – Number of messages removed from `qmaster` send messages buffer. If more messages are added than removed during an interval, the total of messages not yet delivered is about to grow.
  - `#lck0/#ulck0` – Number of calls to `sge_lock()/sge_unlock()` for `qmaster` "global" lock. This lock must always be obtained, when `qmaster`-internal lists (job list, queue list, etc.) are accessed.
  - `#lck1/#ulck1` – Number of calls to `sge_lock()/sge_unlock()` for `qmaster` "master_config" lock. This lock is a secondary lock, but is also important.

Note
The specific columns displayed on your system might differ from the following example.

In this example, performance degraded between 17:40:32 and 17:41:05.

```
CPU     ID      FUNCTION:NAME
  0      1             :BEGIN                 Time |   #wrt  wrt/ms |#rep #gdi #ack|   #dsp
dsp/ms    #sad|   #snd    #rcv|  #in++   #in--  #out++  #out--|  #lck0  #ulck0   #lck1  #ulck1
  0  36909         :tick-3sec 2006 Nov 24 17:39:23 |      43       3|   0    8    4|      3
691     121|      4       4|     11      11      15      15|     68      68     289     288
  0  36909         :tick-3sec 2006 Nov 24 17:39:26 |      83      16|   0   10    3|      3
699     122|      3       3|     14      13      17      17|     90      90     681     681
  0  36909         :tick-3sec 2006 Nov 24 17:39:29 |     117      24|   0    9    4|      4
1092     198|      4       4|     13      13      17      17|     71      71     591     591
  0  36909         :tick-3sec 2006 Nov 24 17:39:32 |      19       4|   0    9    3|      3
591     147|      3       3|     12      12      15      15|     44      43     249     249
  0  36909         :tick-3sec 2006 Nov 24 17:39:35 |     144      28|   0    9    4|      4
1012     173|      4       4|     13      13      17      17|     61      62    1246    1247
  0  36909         :tick-3sec 2006 Nov 24 17:39:38 |      46       5|   0    8    3|      3
705     122|      3       3|     11      11      14      14|     67      67     293     293
  0  36909         :tick-3sec 2006 Nov 24 17:39:41 |     154      31|   0    9    3|      4
894     198|      3       3|     13      13      16      16|     73      72     968     969
  0  36909         :tick-3sec 2006 Nov 24 17:39:44 |      46       5|   0   10    4|      4
971     162|      4       4|     13      13      17      17|     71      72     304     304
  0  36909         :tick-3sec 2006 Nov 24 17:39:47 |     154      29|   0    8    3|      3
739     158|      3       3|     11      11      14      14|     67      67     990     990
  0  36909         :tick-3sec 2006 Nov 24 17:39:50 |      46       5|   0   10    4|      4
815     162|      4       4|     14      14      18      18|     76      76     692     693
  0  36909         :tick-3sec 2006 Nov 24 17:39:53 |      74      15|   0    8    3|      3
746     136|      3       3|     12      12      15      15|     54      53     571     571
  0  36909         :tick-3sec 2006 Nov 24 17:39:56 |     116      20|   0   11    4|      4
992     184|      4       4|     14      14      18      18|     80      81     669     669
  0  36909         :tick-3sec 2006 Nov 24 17:39:59 |      87      18|   0   11    4|      4
851     176|      5       4|     15      15      21      21|     77      76     670     670
  0  36909         :tick-3sec 2006 Nov 24 17:40:02 |     109      20|   0   12    5|      4
930     184|      4       5|     17      17      20      20|     77      78     624     624
  0  36909         :tick-3sec 2006 Nov 24 17:40:05 |      88      15|   0    9    3|      4
995     176|      3       3|     12      12      15      15|     71      71    1026    1026
  0  36909          :tick-3sec 2006 Nov 24 17:40:08 |     112      20|   0   12    4|      4
927     184|      5       4|     16      16      22      22|     81      81     652     652
  0  36909          :tick-3sec 2006 Nov 24 17:40:11 |      32       6|   0    7    4|      3
618     121|      3       4|     11      11      13      13|     54      53     336     336
  0  36909          :tick-3sec 2006 Nov 24 17:40:14 |     145      30|   0   11    4|      4
988     199|      4       4|     15      15      19      19|     64      65     827     827
  0  36909          :tick-3sec 2006 Nov 24 17:40:17 |      43       3|   0    7    3|      3
618     121|      3       3|     10      10      13      13|     64      64     286     286
  0  36909          :tick-3sec 2006 Nov 24 17:40:20 |     157      31|   0   11    4|      4
977     199|      4       4|     15      15      19      19|     80      80    1406    1408
  0  36909          :tick-3sec 2006 Nov 24 17:40:23 |      43       4|   0    7    3|      3
701     121|      3       3|     10      10      13      13|     64      64     285     285
  0  36909          :tick-3sec 2006 Nov 24 17:40:26 |      73      18|   0   11    4|      4
948     171|      4       4|     15      15      19      19|     77      77     700     700
  0  36909          :tick-3sec 2006 Nov 24 17:40:29 |     127      31|   0   10    4|      4
968     189|      4       4|     14      14      18      18|     74      74     584     584
  0  36909          :tick-3sec 2006 Nov 24 17:40:32 |      10       3|   0    6    0|      1
203      41|      0       0|     58       8      62      62|     23      22     106     106
  0  36909          :tick-3sec 2006 Nov 24 17:40:35 |      19       5|   0    5    0|      0
0       0|      0       0|      8       5      13      13|     30      30     200     200
  0  36909          :tick-3sec 2006 Nov 24 17:40:38 |      16       5|   0    5    1|      0
0       0|      0       0|      5       6      10      10|     27      26     558     559
  0  36909          :tick-3sec 2006 Nov 24 17:40:41 |       1       0|   0    4    0|      0
0       0|      0       0|      7       4      11      11|      9       9      34      34
  0  36909          :tick-3sec 2006 Nov 24 17:40:44 |       0       0|   0    4    0|      0
0       0|      0       0|      7       4      11      11|      8       8      28      28
  0  36909          :tick-3sec 2006 Nov 24 17:40:47 |       0       0|   0    6    0|      1
744      81|      1       1|     10       6      15      15|     14      14      33      33
  0  36909          :tick-3sec 2006 Nov 24 17:40:50 |       1       0|   0    5    1|      0
0       0|      0       0|      8       6      14      14|     11      11      49      49
  0  36909          :tick-3sec 2006 Nov 24 17:40:53 |       0       0|   0    4    0|      0
0       0|      0       0|      9       4      12      12|      6       7      28      28
  0  36909          :tick-3sec 2006 Nov 24 17:40:56 |       0       0|   0    5    0|      0
0       0|      0       0|      8       5      13      13|     12      12     420     420
  0  36909          :tick-3sec 2006 Nov 24 17:40:59 |       0       0|   0    4    0|      0
0       0|      0       0|      8       4      12      12|      9       8      30      30
  0  36909          :tick-3sec 2006 Nov 24 17:41:02 |       0       0|   0    4    1|      0
0       0|      0       0|     12       5      16      16|      7       8      25      25
  0  36909          :tick-3sec 2006 Nov 24 17:41:05 |     165      41|   0   48   60|      0
0       0|      1       1|     23     106      71      71|     96      97    1236    1236
  0  36909          :tick-3sec 2006 Nov 24 17:41:08 |     178      28|   0   15   53|      4
```

```
965      206|       4        4|      68       68       75       75|     130      130     1336     1336
  0  36909           :tick-3sec 2006 Nov 24 17:41:11 |      106       23|   0   27   35|       4
855      166|       4        4|      82       82       91       91|     115      114     1040     1040
  0  36909           :tick-3sec 2006 Nov 24 17:41:14 |      198       37|   0   41   70|       4
1189      196|       4        4|     185      185      185      185|     134      135     1327     1327
  0  36909           :tick-3sec 2006 Nov 24 17:41:17 |       16        5|   0    9    5|       4
940      161|       3        3|      17       17       20       20|      43       42      234      234
  0  36909           :tick-3sec 2006 Nov 24 17:41:20 |      162       35|   0   13    8|       4
958      200|       4        4|      23       23       28       28|      80       81     1018     1018
  0  36909           :tick-3sec 2006 Nov 24 17:41:23 |       44        6|   0    6    3|       2
544       81|       3        3|       8        8       11       11|      63       63      747      747
  0  36909           :tick-3sec 2006 Nov 24 17:41:26 |      150       34|   0   13    6|       4
921      199|       4        4|      21       21       25       25|      73       72      923      923
  0  36909           :tick-3sec 2006 Nov 24 17:41:29 |       43        3|   0    5    2|       2
506       81|       2        2|       7        7        9        9|      57       57      260      260
  0  36909           :tick-3sec 2006 Nov 24 17:41:32 |      157       37|   0    9    3|       4
978      199|       3        3|      13       13       16       16|      73       72      970      970
  0  36909           :tick-3sec 2006 Nov 24 17:41:35 |       43        3|   0    7    3|       2
512       85|       3        3|       9        9       12       12|      61       62      274      274
  0  36909           :tick-3sec 2006 Nov 24 17:41:38 |      127       29|   0    8    3|       4
994      185|       3        3|      11       11       14       14|      68       68     1265     1265
  0  36909           :tick-3sec 2006 Nov 24 17:41:41 |       66       11|   0   10    4|       4
973      171|       4        4|      14       14       18       18|      67       67      354      354
  0  36909           :tick-3sec 2006 Nov 24 17:41:44 |       48       10|   0    8    3|       3
785      128|       3        3|      11       11       14       14|      52       51      399      399
  0  36909           :tick-3sec 2006 Nov 24 17:41:47 |      142       31|   0   12    4|       4
913      192|       5        4|      17       17       23       23|      89       90      830      830
```

```
   0  36909          :tick-3sec 2006 Nov 24 17:41:50 |      64       13|   0   11    5|       4
 853     168|      4       5|     15      15      18      18|     75      75     542     542
```