N1 Grid Engine 6 Administration Guide

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 817–5677–20
May 2005

050701@12762

Contents

Tables

Figures

Examples

Preface

The *N1 Grid Engine 6 Administration Guide* provides background information about how to set up and administer a system of networked computer hosts that run N1™ Grid Engine 6 software. This version of the manual supports Grid Engine 6 Update 4

Who Should Use This Book

The background information and instructions in this guide are intended for experienced system administrators.

How This Book Is Organized

The *N1 Grid Engine 6 Administration Guide* includes eight chapters.

- Chapter 1 provides general background about *hosts* and *clusters*, along with detailed instructions for how to configure them.
- Chapter 2 describes *queues*, which serve as containers for different categories of jobs. The chapter includes complete instructions for how to configure cluster queues and queue instances.
- Chapter 3 explains how the grid engine system uses the *complex* to define all the pertinent information concerning the resource attributes a user can request for a job. The administrator configures complex resource attributes to match the requirements of the environment. This chapter provides detailed instructions for how to configure resource attributes.
- Chapter 4 provides background information about different types of users of the grid engine system. The chapter provides instructions on how to set up and maintain user access and project access.

- Chapter 5 provides full background information about the types of *user policies* that are available. The chapter provides instructions on how to match these policies to the computing environment. Chapter 5 also describes how to configure and modify the scheduler.
- Chapter 6 describes how the grid engine system fits in with *parallel environments*, and provides detailed instructions on how to configure them. The chapter also describes how to set up and use *checkpointing environments*.
- Chapter 7 describes how to gather reporting and accounting statistics, how to automatically back up your grid engine system configuration files, and how to use files and scripts to add or modify objects such as queues, hosts, and environments.
- Chapter 8 describes some ways to fine-tune your grid engine system. It also explains how the grid engine system retrieves error message and describes how to run the software in debug mode.
- Chapter 9, DBWriter describes how you can modify the DBWriter portion of the ARCo feature.

Note – Some of the material in this guide appeared originally in the "How-To" section of the Sun Grid Engine project web site. Updated frequently, this web site is of special value to administrators of the grid engine system and is well worth consulting.

Related Books

Other books in the N1 Grid Engine 6 software documentation collection include:

- *N1 Grid Engine 6 User's Guide*
- *N1 Grid Engine 6 Installation Guide*
- *N1 Grid Engine 6 Release Notes*

Accessing Sun Documentation Online

The docs.sun.com^{SM} Web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. The URL is `http://docs.sun.com`.

Ordering Sun Documentation

Sun Microsystems offers select product documentation in print. For a list of documents and how to order them, see "Buy printed documentation" at `http://docs.sun.com`.

Typographic Conventions

The following table describes the typographic changes that are used in this book.

TABLE P–1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories, and on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`** `Password:`
AaBbCc123	Command-line placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms or terms to be emphasized	Read Chapter 6 in the *User's Guide*. Do *not* save the file.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–2 Shell Prompts

Shell	Prompt
C shell prompt	`machine_name%`
C shell superuser prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
Bourne shell and Korn shell superuser prompt	`#`

CHAPTER 1

Configuring Hosts and Clusters

This chapter provides background information about configuring various aspects of the grid engine system. This chapter includes instructions for the following tasks:

About Hosts and Daemons

Grid engine system hosts are classified into four groups, depending on which daemons are running on the system and on how the hosts are registered at `sge_qmaster`.

- **Master host.** The master host is central for the overall cluster activity. The master host runs the master daemon `sge_qmaster`. `sge_qmaster` controls all grid engine system components such as queues and jobs. It also maintains tables about the status of the components, about user access permissions and the like. The master host usually runs the scheduler `sge_schedd`. 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" in *N1 Grid Engine 6 Installation Guide*. For information about how to configure dynamic changes to the master host, see "Configuring Shadow Master Hosts" on page 21.

- **Execution hosts.** Execution hosts are nodes that have permission to run jobs. Therefore they host queue instances, and they run the execution daemon `sge_execd`. An execution host is initially set up by the installation procedure, as described in "How to Install Execution Hosts" in *N1 Grid Engine 6 Installation Guide*.

- **Administration hosts.** Permission can be given to hosts other than the master host to carry out any kind of administrative activity. Administrative hosts are set up with the following command:

  `qconf -ah` *hostname*

  See the `qconf`(1) man page for details.

- **Submit hosts.** Submit hosts allow for submitting and controlling batch jobs only. In particular, a user who is logged into a submit host can use `qsub` to submit jobs, can use `qstat` to control the job status, or can run the graphical user interface `QMON`. Submit hosts are set up using the following command:

  `qconf -as` *hostname*

  See the `qconf`(1) man page for details.

Note – A host can belong to more than one class. The master host is by default an administration host and a submit host.

Changing the Master Host

Because the spooling database cannot be located on an NFS-mounted file system, the following procedure requires that the Berkeley DB RPC server be used 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.

To change the master host, do the following:

1. On the current master host, stop the master daemon and the scheduler daemon by typing the following command:

   ```
   qconf -ks -km
   ```

2. Edit the *sge-root*/*cell*/`common/act_qmaster` file according to the following guidelines:

   a. In the `act_qmaster` file, replace the current host name with the new master host's name.

      This name should be the same as the name returned by the `gethostname` utility. To get that name, type the following command on the new master host:

      sge-root`/utilbin/$ARCH/gethostname`

   b. Replace the old name in the `act_qmaster` file with the name returned by the `gethostname` utility.

3. On the new master host, run the following script:

   sge-root/*cell*/`common/sge5`

   This starts up `sge_qmaster` and `sge_schedd` on the new master host.

Configuring Shadow Master Hosts

Shadow master hosts are machines in the cluster that can detect a failure of the master daemon and take over its role as master host. When the shadow master daemon detects that the master daemon `sge_qmaster` has failed abnormally, it starts up a new `sge_qmaster` on the host where the shadow master daemon is running.

Note – If the master daemon is shut down gracefully, the shadow master daemon does not start up. If you want the shadow master daemon to take over after you shut down the master daemon gracefully, remove the lock file that is located in the `sge_qmaster` spool directory. The default location of this spool directory is *sge-root*/*cell*/`spool/qmaster`.

The automatic failover start of a `sge_qmaster` on a shadow master host takes approximately one minute. Meanwhile, you get an error message whenever a grid engine system command is run.

Note – The file *sge-root*/*cell*/`common/act_qmaster` contains the name of the host actually running the `sge_qmaster` daemon.

Shadow Master Host Requirements

To prepare a host as a shadow master, the following requirements must be met:

- The shadow master host must run `sge_shadowd`.
- The shadow master host must share `sge_qmaster`'s status information, job configuration, and queue configuration logged to disk. In particular, a shadow master host needs read/write root access to the master host's spool directory and to the directory *sge-root*/*cell*/`common`.
- Either the Berkeley DB RPC server or classic grid engine system spooling must be used for `sge_qmaster` spooling. For more information, see "Database Server and Spooling Host" in *N1 Grid Engine 6 Installation Guide*.
- The shadow-master-hostname file must contain a line that defines the host as shadow master host.

As soon as these requirements are met, the shadow-master-host facility is activated for this host. No restart of grid engine system daemons is necessary to activate the feature.

Shadow Master Hosts File

The shadow master host name file, *sge-root*/*cell*/`common/shadow_masters`, contains the following:

- The name of the primary master host, which is the machine where the master daemon `sge_qmaster` initially runs
- The names of the shadow master hosts

The format of the shadow master hostname file is as follows:

- The first line of the file defines the primary master host
- The following lines define the shadow master hosts, one host per line

The order of the shadow master hosts is significant. The primary master host is the first line in the file. If the primary master host fails to proceed, the shadow master defined in the second line takes over. If this shadow master also fails, the shadow master defined in the third line takes over, and so forth.

Starting Shadow Master Hosts

In order to start a shadow `sge_qmaster`, the system must be sure either that the *old* `sge_qmaster` has terminated, or that it will terminate without performing actions that interfere with the newly-started shadow `sge_qmaster`.

In very rare circumstances it might be impossible to determine that the old `sge_qmaster` has terminated or that it will terminate. In such cases, an error message is logged to the messages log file of the `sge_shadowds` on the shadow master hosts. See Chapter 8. Also, any attempts to open a `tcp` connection to a `sge_qmaster` daemon permanently fail. If this occurs, make sure that no master daemon is running, and then restart `sge_qmaster` manually on any of the shadow master machines. See "Restarting Daemons From the Command Line" on page 39.

Configuring Shadow Master Hosts Environment Variables

There are three environment variables which affect the takeover time for a shadow master:

- SGE_DELAY_TIME - This variable controls the interval in which `sge_shadowd` pauses if a takeover bid fails. This value is used only when there are multiple `sge_shadowd` instances and they are contending to be the master. (the default is 600 seconds.)
- SGE_CHECK_INTERVAL - This variable controls the interval in which the `sge_shadowd` checks the heartbeat file (60 seconds by default.)
- SGE_GET_ACTIVE_INTERVAL - This variable controls the interval when a `sge_shadowd` instance tries to take over when the heartbeat file has not changed.

These variables interact in the following way.

1. The master host updates the heartbeat file every 30 seconds.
2. The `sge_shadowd` daemon checks for changes to heartbeat file every number of seconds defined by the SGE_CHECK_INTERVAL variable. So, this value must be greater than 30 seconds.

3. If the `sge_shadowd` daemon notices that the heartbeat file has been updated updated, it starts waiting again until it is once more time to check the heartbeat file.
4. If the `sge_shadowd` daemon notices that the heartbeat file has not been updated, it waits for number of seconds defined by the SGE_CHECK_INTERVAL variable to expire. This step lets you make sure that the `sge_shadowd` daemon is not too agressive in trying to takeover and allows the master host some leeway in updating the heartbeat file.
5. When the SGE_GET_ACTIVE_INTERVAL has expired, `sge_shadowd` daemon takes over if heartbeat file is still not updated.

A reasonable configuration might be to set the SGE_CHECK_INTERVAL to be 45 seconds and the SGE_GET_ACTIVE_INTERVAL to be 90 seconds. So, after about 2 minutes, the take over will occur. If you want to check the operation of the shadow host after you have configured these environment variables you will have to pull out the master host's network cable to simulate a failure.

Configuring Hosts

N1 Grid Engine 6 software (*grid engine software*) maintains object lists for all types of hosts except for the master host. The lists of administration host objects and submit host objects indicate whether a host has administrative or submit permission. The execution host objects include other parameters. Among these parameters are the load information that is reported by the `sge_execd` running on the host, and the load parameter scaling factors that are defined by the administrator.

You can configure host objects with `QMON` or from the command line.

`QMON` provides a set of host configuration dialog boxes that are invoked by clicking the Host Configuration button on the `QMON` Main Control window. The Host Configuration dialog box has four tabs:

- Administration Host tab. See Figure 1–3.
- Submit Host tab. See Figure 1–4.
- Host Groups tab. See Figure 1–5.
- Execution Host tab. See Figure 1–1.

The `qconf` command provides the command-line interface for managing host objects.

Configuring Execution Hosts With `QMON`

Before you configure an execution host, you must first install the software on the execution host as described in "How to Install Execution Hosts" in *N1 Grid Engine 6 Installation Guide*.

To configure execution hosts, on the QMON Main Control window click the Host Configuration button, and then click the Execution Host tab. The Execution Host tab looks like the following figure:

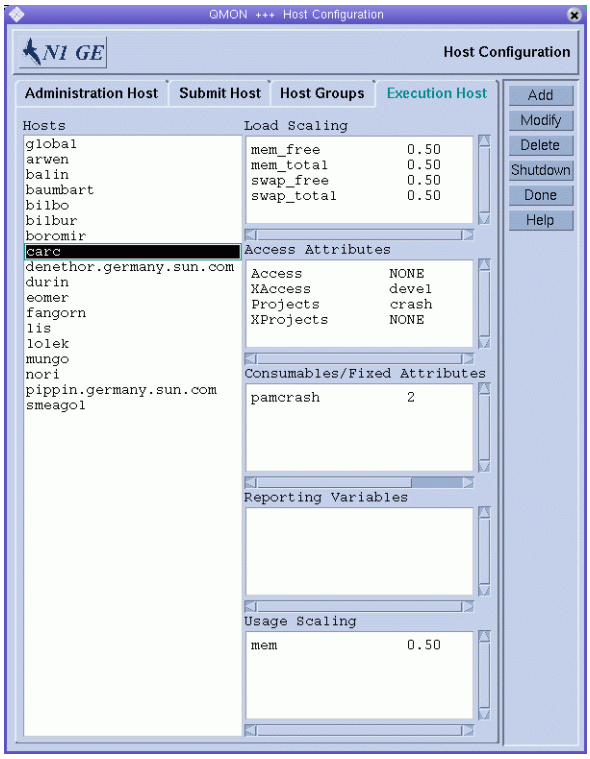


FIGURE 1–1 Execution Host Tab

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 "Configuring Administration Hosts With QMON" on page 31 and "Configuring Submit Hosts With QMON" on page 32.

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 "Load Parameters" on page 87 for information about load parameters.

The Access Attributes list displays access permissions. See Chapter 4 for information about access permissions.

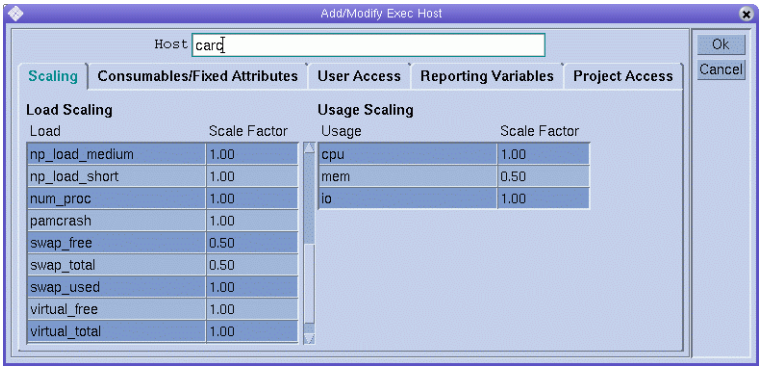
The Consumables/Fixed Attributes list displays resource availability for consumable and fixed resource attributes associated with the host. See "Complex Resource Attributes" on page 67 for information about resource attributes.

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" on page 29 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 window have a scaling factor of 1.

Adding or Modifying an Execution Host

To add or modify an execution host, click Add or Modify. The Add/Modify Exec Host dialog box appears.



The Add/Modify Exec Host dialog box enables you to modify all attributes associated with an execution host. The name of an existing execution host is displayed in the Host field.

If you are adding a new execution host, type its name in the Host field.

Defining Scaling Factors

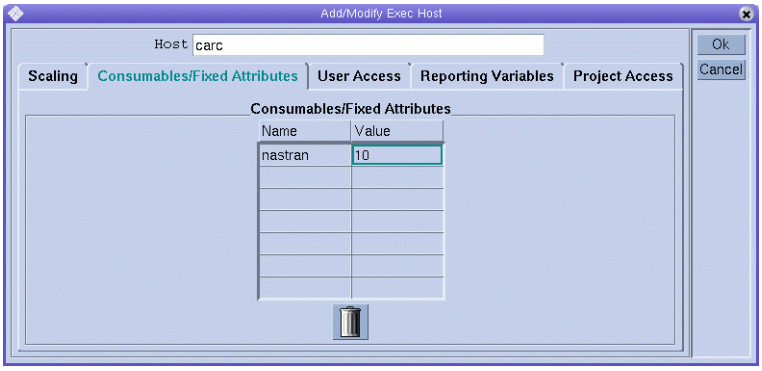
To define scaling factors, click the Scaling tab.

The Load column of the Load Scaling table lists all available load parameters, and 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.

Defining Resource Attributes

To define the resource attributes to associate with the host , click the Consumables/Fixed Attributes tab.



The resource attributes associated with the host are listed in the Consumables/Fixed Attributes table.

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 "Complex Resource Attributes" on page 67.

The Consumables/Fixed Attributes table lists all resource attributes for which a value is currently defined. 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.

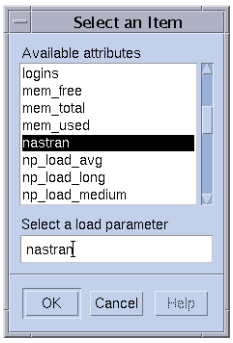


FIGURE 1–2 Attribute Selection Dialog Box

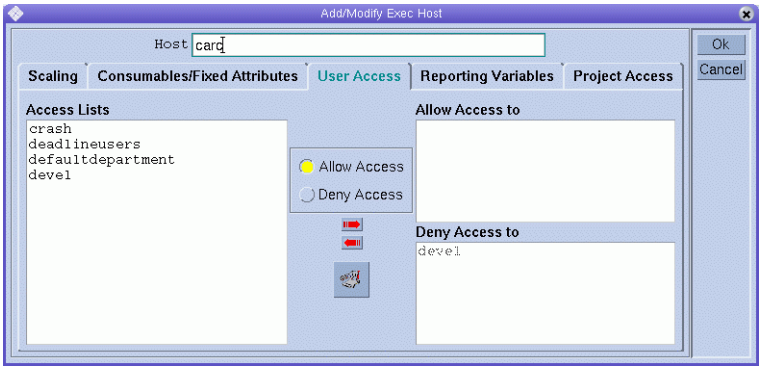
To add an attribute to the Consumables/Fixed Attributes table, select the attribute, and then click OK.

To modify an attribute value, double-click a Value field, and then type a value.

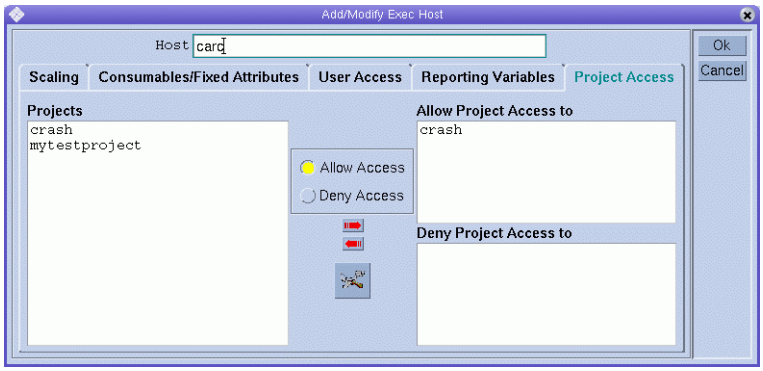
To delete an attribute, select the attribute, and then press Control-D or click mouse button 3. Click OK to confirm that you want to delete the attribute.

Defining Access Permissions

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

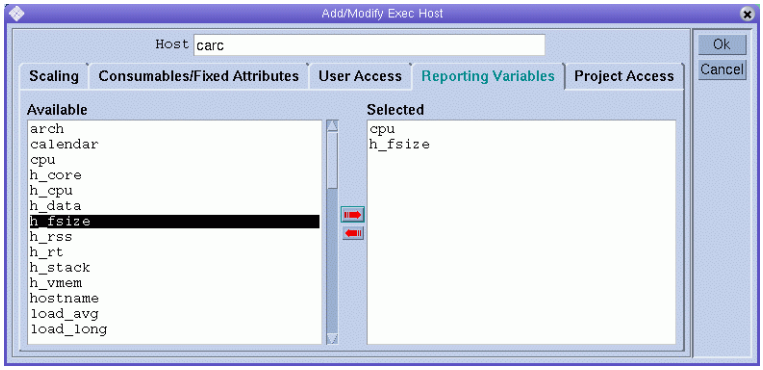


To define project access permissions to the execution host based on previously configured projects, click the Project Access tab.



Defining Reporting Variables

To define reporting variables, click the Reporting Variables tab.



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.

Select a reporting variable from the Available list, and then click the red right arrow to add the selected variable to the Selected list.

To remove a reporting variable from the Selected list, select the variable, and then click the left red arrow.

Deleting an Execution Host

To delete an execution host, on the QMON Main Control window click the Host Configuration button, and then click the Execution Host tab.

In the Execution Host dialog box, select the host that you want to delete, and then click Delete.

Shutting Down an Execution Host Daemon

To shut down an execution host daemon, on the QMON Main Control window click the Host Configuration button, and then click the Execution Host tab.

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

Configuring Execution Hosts From the Command Line

To configure execution hosts from the command line, type the following command with appropriate options:

```
% qconf options
```

The following options are available:

- `qconf -ae [exec-host]`

  The `-ae` option (add execution host) displays an editor containing an execution host configuration template. The editor is either the default `vi` editor or an editor corresponding 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. The execution host is configured by changing the template and saving to disk. See the `host_conf`(5) man page for a detailed description of the template entries to be changed.

- `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.

- `qconf -me hostname`

  The `-me` option (modify execution host) displays an editor containing the configuration of the specified execution host as template. The editor is either the default `vi` editor or an editor corresponding to the `EDITOR` environment variable. The execution host configuration is modified by changing the template and saving to disk. See the `host_conf`(5) man page for a detailed description of the template entries to be changed.

- `qconf -Me filename`

  The `-Me` option (modify execution host) uses the content of *filename* as 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.

- `qconf -se` *hostname*

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

- `qconf -sel`

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

Configuring Administration Hosts With `QMON`

On the `QMON` Main Control window, click the Host Configuration button. The Host Configuration dialog box appears, displaying the Administration Host tab. The Administration Host tab looks like the following figure:

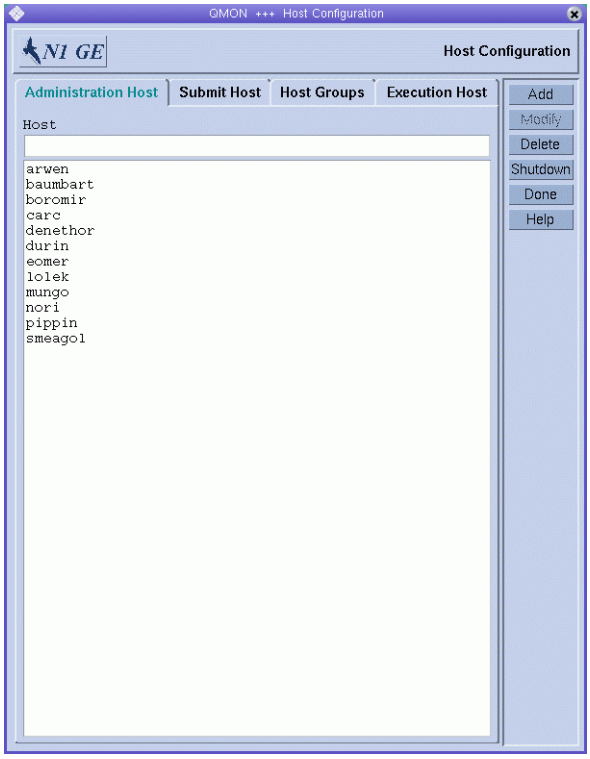


FIGURE 1–3 Administration Host Tab

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

Use the Administration Host tab to configure hosts on which administrative commands are allowed. The Host list displays the hosts that already have administrative permission.

Adding an Administration Host

To add a new administration host, type its name in the Host field, and then click Add, or press the Return key.

Deleting an Administration Host

To delete an administration host from the list, select the host, and then click Delete.

Configuring Administration Hosts From the Command Line

To configure administration hosts from the command line, type the following command with appropriate arguments:

```
% qconf arguments
```

Arguments to the `qconf` command and their consequences are as follows:

- `qconf -ah` *hostname*

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

- `qconf -dh` *hostname*

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

- `qconf -sh`

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

Configuring Submit Hosts With `QMON`

To configure submit hosts, on the `QMON` Main Control window click the Host Configuration button, and then click the Submit Host tab. The Submit Host tab is shown in the following figure.

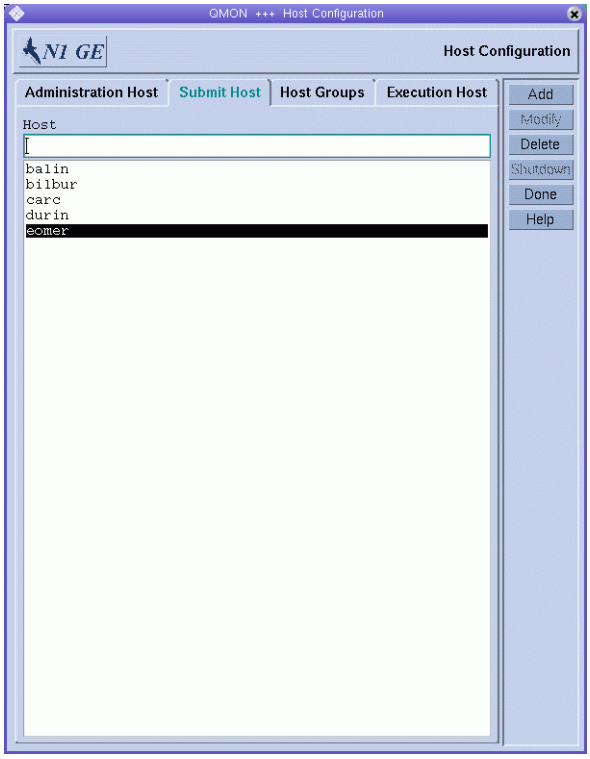


FIGURE 1–4 Submit Host Tab

Use the Submit Host tab to declare the hosts from which jobs can be submitted, monitored, and controlled. The Host list displays the hosts that already have submit permission.

No administrative commands are allowed from submit hosts unless the hosts are also declared to be administration hosts. See "Configuring Administration Hosts With QMON" on page 31 for more information.

Adding a Submit Host

To add a submit host, type its name in the Host field, and then click Add, or press the Return key.

Deleting a Submit Host

To delete a submit host, select it, and then click Delete.

Configuring Submit Hosts From the Command Line

To configure submit hosts from the command line, type the following command with appropriate arguments:

```
% qconf arguments
```

The following options are available:

- `qconf -as` *hostname*

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

- `qconf -ds` *hostname*

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

- `qconf -ss`

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

Configuring Host Groups With `QMON`

Host groups enable you to use a single name to refer to multiple hosts. You can group similar hosts together in a host group. 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`. This host group 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`.

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

Click the Host Groups tab. The Host Groups tab looks like the following figure.

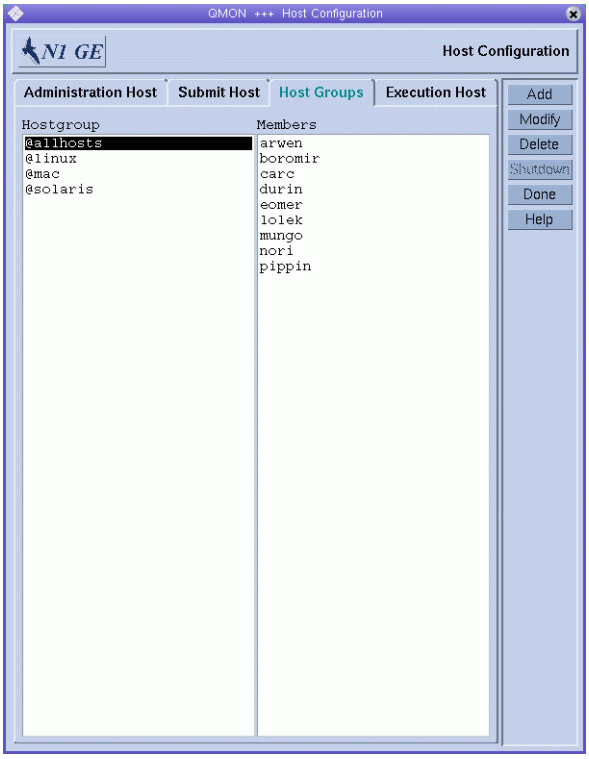
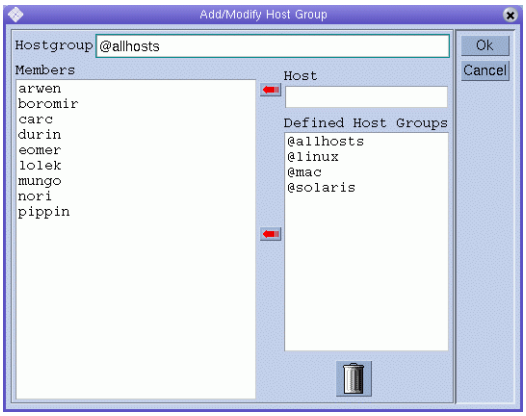


FIGURE 1–5 Host Groups Tab

Use the Host Groups tab to configure host groups. The Hostgroup list displays the currently configured host groups. The Members list displays all the hosts that are members of the selected host group.

Adding or Modifying a Host Group

To add a host group, click Add. To Modify a host group, click Modify. The Add/Modify Host Group dialog box appears.



If you are adding a new host group, type a host group name in the Hostgroup field. The host group name must begin with an @ sign.

If you are modifying an existing host group, the host group name is provided in the Hostgroup field.

To add a host to the host group that you are configuring, type the host name in the Host field, and then click the red arrow to add the name to the Members list. To add a host group as a subgroup, select a host group name from the Defined Host Groups list, and then click the red arrow to add the name to the Members list.

To remove a host or a host group from the Members list, select it, and then click the trash icon.

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

Deleting a Host Group

To delete a host group, select it from the Hostgroup list, and then click Delete.

Configuring Host Groups From the Command Line

To configure host groups from the command line, type the following command with appropriate options:

```
% qconf options
```

The following options are available:

- `qconf -ahgrp` [*host-group-name*]

  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.

- `qconf -Ahgrp` [*filename*]

  The `-Ahgrp` option (add host group from file) displays an editor containing a host group configuration defined in *filename*. The editor is either the default `vi` editor or an editor corresponding to the `EDITOR` environment variable. The host group is configured by changing the configuration and saving to disk.

- `qconf -dhgrp` *host-group-name*

  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.

- `qconf -mhgrp` *host-group-name*

  The `-mhgrp` option (modify host group) displays an editor containing the configuration of the specified host group as template. The editor is either the default `vi` editor or an editor corresponding to the `EDITOR` environment variable. The host group configuration is modified by changing the template and saving to disk.

- `qconf -Mhgrp` *filename*

  The `-Mhgrp` option (modify host group from file) uses the content of *filename* as host group configuration template. The configuration in the specified file must refer to an existing host group. The configuration of this host group is replaced by the file content.

- `qconf -shgrp` *host-group-name*

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

- `qconf -shgrp_tree` *host-group-name*

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

- `qconf -shgrp_resolved` *host-group-name*

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

- `qconf -shgrpl`

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

Monitoring Execution Hosts With `qhost`

Use the `qhost` command to retrieve a quick overview of the execution host status:

```
% qhost
```

This command produces output that is similar to the following example:

EXAMPLE 1–1 Sample `qhost` Output

```
HOSTNAME                ARCH         NCPU  LOAD  MEMTOT  MEMUSE  SWAPTO  SWAPUS
----------------------------------------------------------------------------------
global                  -               -     -       -       -       -       -
arwen                   aix43           1     -       -       -       -       -
baumbart                irix65          2  0.00    1.1G   91.5M  128.0M     0.0
boromir                 hp11            1     -  128.0M       -  256.0M       -
carc                    lx24-amd64      2  0.00    3.8G  989.8M    1.0G     0.0
denethor                aix51           1 4.54G       -       -       -       -
durin                   lx24-x86        1  0.37  123.1M   46.5M  213.6M   26.6M
eomer                   sol-sparc64     1  0.13  256.0M  248.0M  513.0M   93.0M
lolek                   tru64           1  0.02    1.0G  790.0M    1.0G    8.0K
mungo                   lx22-alpha      1  1.00  248.9M   78.8M  129.8M    2.5M
nori                    sol-x86         2  0.38 1023.0M  372.0M  512.0M   37.0M
pippin                  darwin          1  0.00  640.0M  264.0M     0.0     0.0
smeagol                 hp11            1  0.35  512.0M  425.0M    1.0G   95.0M
```

See the `qhost`(1) man page for a description of the output format and for more options.

Invalid Host Names

The following is a list of host names that are invalid, reserved, or otherwise not allowed to be used:

```
global
template
all
default
unknown
none
```

Killing Daemons From the Command Line

To kill grid engine system daemons from the command line, use one of the following commands:

```
% qconf -ke[j] {hostname,... | all}
% qconf -ks
% qconf -km
```

You must have manager or operator privileges to use these commands. See Chapter 4 for more information about manager and operator privileges.

- The `qconf –ke` command shuts down the execution daemons. However, it does not cancel 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. The job reports are not lost, however.

  The `qconf -kej` command kills all currently active jobs and brings down all execution daemons.

  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.
- The `qconf -ks` command shuts down the scheduler `sge_schedd`.
- The `qconf -km` command forces the `sge_qmaster` process to terminate.

If you want to wait for any active jobs to finish before you run the shutdown procedure, use the `qmod -dq` command for each cluster queue, queue instance, or queue domain before you run the `qconf` sequence described earlier. For information about cluster queues, queue instances, and queue domains, see "Configuring Queues" on page 45.

```
% qmod -dq {cluster-queue | queue-instance | queue-domain}
```

The `qmod -dq` command prevents new jobs from being scheduled to the disabled queue instances. You should then wait until no jobs are running in the queue instances before you kill the daemons.

Restarting Daemons From the Command Line

Log in as root on the machine on which you want to restart grid engine system daemons.

Type the following commands to run the startup scripts:

```
% sge-root/cell/common/sgemaster
% sge-root/cell/common/sgeexecd
```

These scripts looks for the daemons normally running on this host and then start the corresponding ones.

Basic Cluster Configuration

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 be kept up to date afterwards.

The `sge_conf`(5) man page contains a detailed description of the configuration entries.

Displaying a Cluster Configuration With `QMON`

On the `QMON` Main Control window, click the Cluster Configuration button. The Cluster Configuration dialog box appears.

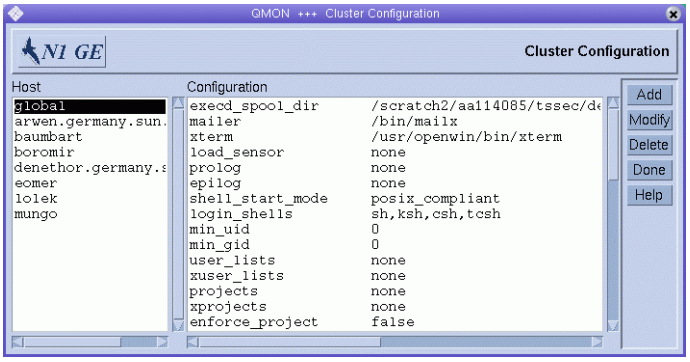


FIGURE 1–6 Cluster Configuration Dialog Box

In the Host list, select the name of a host. The current configuration for the selected host is displayed under Configuration.

Displaying the Global Cluster Configuration With QMON

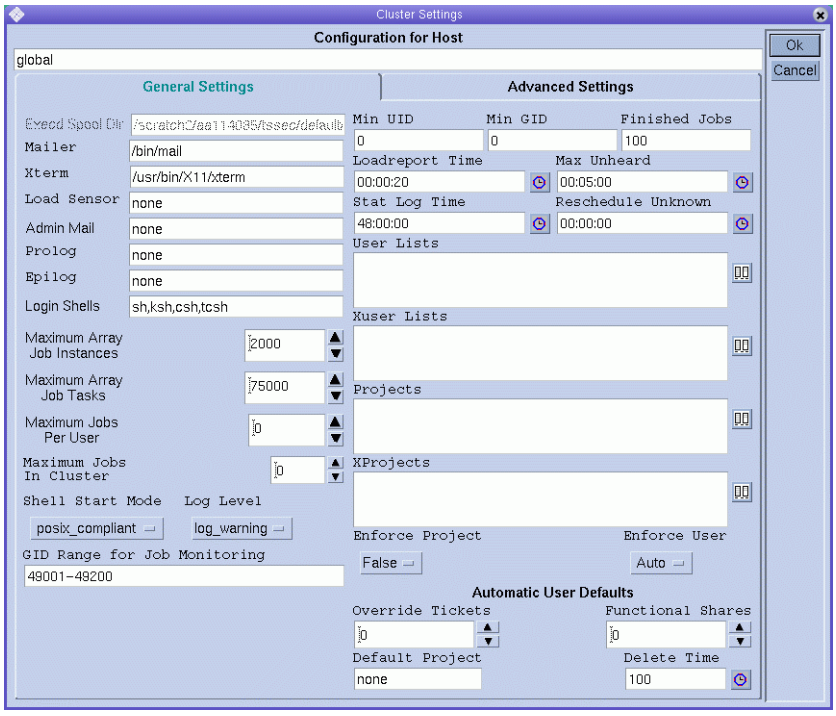
On the QMON Main Control window, click the Cluster Configuration button.

In the Host list, select `global`.

The configuration is displayed in the format that is described in the `sge_conf`(5) man page.

Adding and Modifying Global and Host Configurations With QMON

In the Cluster Configuration dialog box (Figure 1–6), select a host name or the name `global`, and then click Add or Modify. The Cluster Settings dialog box appears.

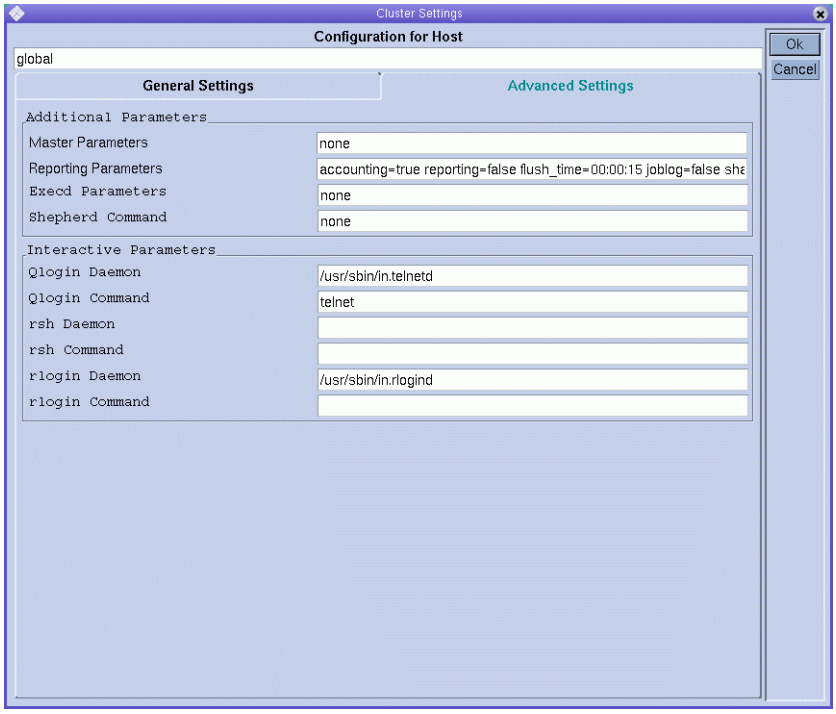


The Cluster Settings dialog box enables you to change all parameters of a global configuration or a local host configuration.

All fields of the dialog box are accessible only if you are modifying the global configuration. If you modify a local host, its configuration is reflected in the dialog box. You can modify only those parameters that are feasible for local host changes.

If you are adding a new local host configuration, the dialog box fields are empty.

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.



When you finish making changes, click OK to save your changes and close the dialog box. Click Cancel to close the dialog box without saving changes.

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

Deleting a Cluster Configuration With `QMON`

On the `QMON` Main Control window, click the Cluster Configuration button.

In the Host list, select the name of a host whose configuration you want to delete, and then click Delete.

Displaying the Basic Cluster Configurations From the Command Line

To display the current cluster configuration, use the `qconf -sconf` command. See the `qconf`(1) man page for a detailed description.

Type one of the following commands:

```
% qconf -sconf
% qconf -sconf global
% qconf -sconf host
```

- The `qconf –sconf` and `qconf –sconf global` commands are equivalent. They display the global configuration.
- The `qconf -sconf` *host* command displays the specified local host's configuration.

Modifying the Basic Cluster Configurations From the Command Line

---

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

Type one of the following commands:

```
% qconf -mconf global
% qconf -mconf host
```

- The `qconf -mconf global` command modifies the global configuration.
- The `qconf -mconf` *host* command modifies the local configuration of the specified execution host or master host.

The `qconf` commands that are described here are examples of the many available `qconf` commands. See the `qconf`(1) man page for others.

CHAPTER 2

Configuring Queues and Queue Calendars

This chapter provides background information about configuring queues and queue calendars. It also includes instructions for how to configure them.

The following is a list of specific tasks for which instructions are included in this chapter.

Configuring Queues

Queues are containers for different categories of jobs. Queues provide the corresponding resources for concurrent execution of multiple jobs that belong to the same category.

In N1 Grid Engine 6, a queue can be associated with one host or with multiple hosts. Because queues can extend across multiple hosts, they are called *cluster queues*. Cluster queues enable you to manage a cluster of execution hosts by means of a single cluster queue configuration.

Each host that is associated with a cluster queue receives an instance of that cluster queue, which resides on that host. This guide refers to these instances as *queue instances*. Within any cluster queue, you can configure each queue instance separately. 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.

Note – The distinction between cluster queues and queue instances is important. For example, jobs always run in queue instances, not in cluster queues.

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

A host group is a group of hosts that can be treated collectively as identical. Host groups enable you to manage multiple hosts by means of a single host group configuration. For more information about host groups, see "Configuring Host Groups With `QMON`" on page 34.

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`.

Note – Queue domain names always include two @ signs, because all host group names begin with an @ sign..

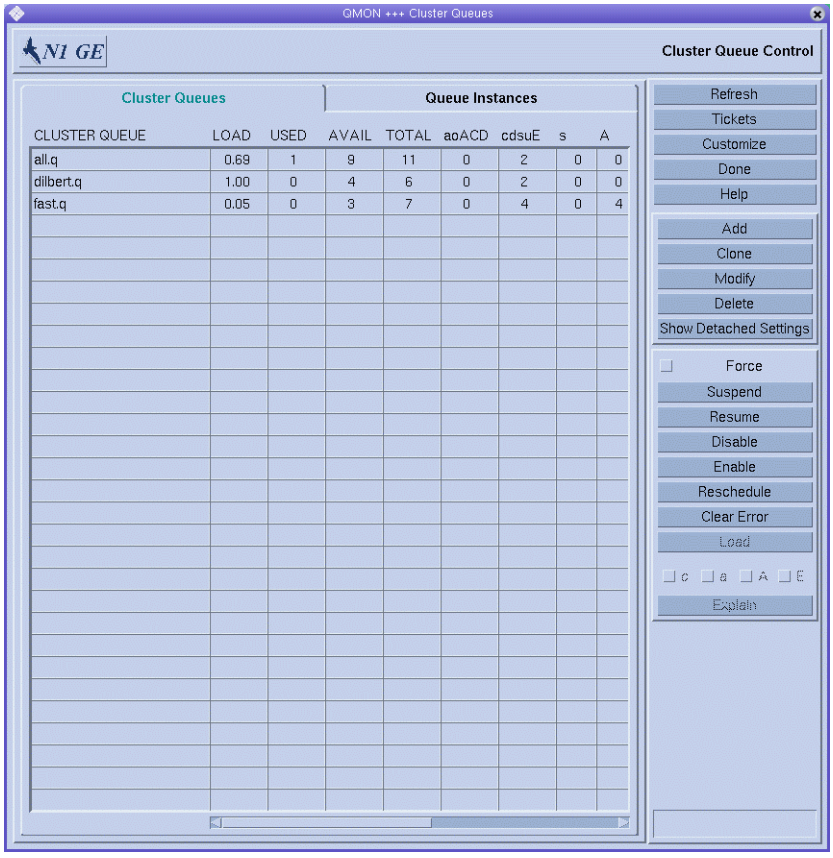
Jobs do not wait in queue instances. Jobs start running immediately as soon as they are dispatched. The scheduler's list of pending jobs is the only waiting area for jobs.

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.

For further details, see the `queue_conf`(5) man page.

Configuring Queues With QMON

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



The Cluster Queues dialog box and its facilities for monitoring and manipulating the status of cluster queues and queue instances are described in "Monitoring and Controlling Queues With QMON" in *N1 Grid Engine 6 User's Guide*.

To add a new cluster queue, click Add.

To modify an existing cluster queue, select it from the Cluster Queue list, and then click Modify.

The Clone button enables you to import all parameters of an existing cluster queue. You select the queue you want to clone from a list of existing queues.

When you click Add, the Queue Configuration – Add dialog box appears. When you click Modify, the Modify *queue-name* dialog box appears. When the Queue Configuration dialog box appears for the first time, it displays the General Configuration tab.

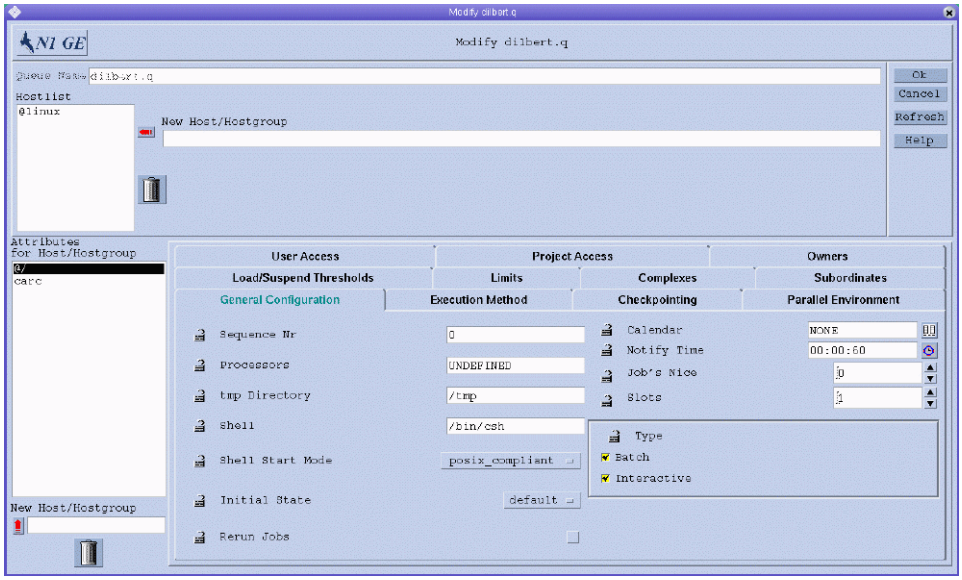


FIGURE 2–1 Queue Configuration– General Configuration Tab

If you are modifying an existing queue, the name of the queue is displayed in the Queue Name field. The hosts where the queue instances reside are displayed in the Hostlist field.

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.

In the Hostlist field, you can specify the names of individual hosts. You can also specify the names of previously defined host groups. Queue instances of this cluster queue will reside on all individual hosts and on all members of the host groups you specify, including all members of any host subgroups. For more information about host groups, see "Configuring Host Groups With QMON" on page 34.

The following 11 tabs for specifying parameter sets are available to define a queue:

- General Configuration – see "Configuring General Parameters" on page 49
- Execution Method – see "Configuring Execution Method Parameters" on page 50
- Checkpointing – see "Configuring the Checkpointing Parameters" on page 51
- Parallel Environment – see "Configuring Parallel Environments" on page 52
- Load/Suspend Thresholds – see "Configuring Load and Suspend Thresholds" on page 53
- Limits – see "Configuring Limits" on page 55

- Complex – see "Configuring Complex Resource Attributes" on page 56
- Subordinates – see "Configuring Subordinate Queues" on page 57
- User Access – see "Configuring User Access Parameters" on page 58
- Project Access – see "Configuring Project Access Parameters" on page 59
- Owners – see "Configuring Owners Parameters" on page 60

To set default parameters for the cluster queue, select `@/` in the Attributes for Host/Hostgroup list, and then click the tab containing the parameters that you want to set.

Default parameters are set for all queue instances on all hosts listed under Hostlist. You can override the default parameter values on a host or a host group that you specify. To set override parameters for a host or a host group, first select the name from the Attributes for Host/Hostgroup list. Then click the tab containing the parameters that you want to set. The values of the parameters that you set override the cluster queue's default parameters on the selected host or host group.

To set a host-specific parameter, you must first enable the parameter for configuration. Click the lock icon at the left of the parameter you want to set, and then change the parameter's value.

The Refresh button loads the settings of other objects that were modified while the Queue Configuration dialog box was open.

Click OK to register all queue configuration changes with `sge_qmaster` and close the dialog box. Click Cancel to close the dialog box without saving your changes.

Configuring General Parameters

To configure general parameters, click the General Configuration tab. The General Configuration tab is shown in Figure 2–1.

You can specify 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 N1 Grid Engine 6 software distribution for more information.

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

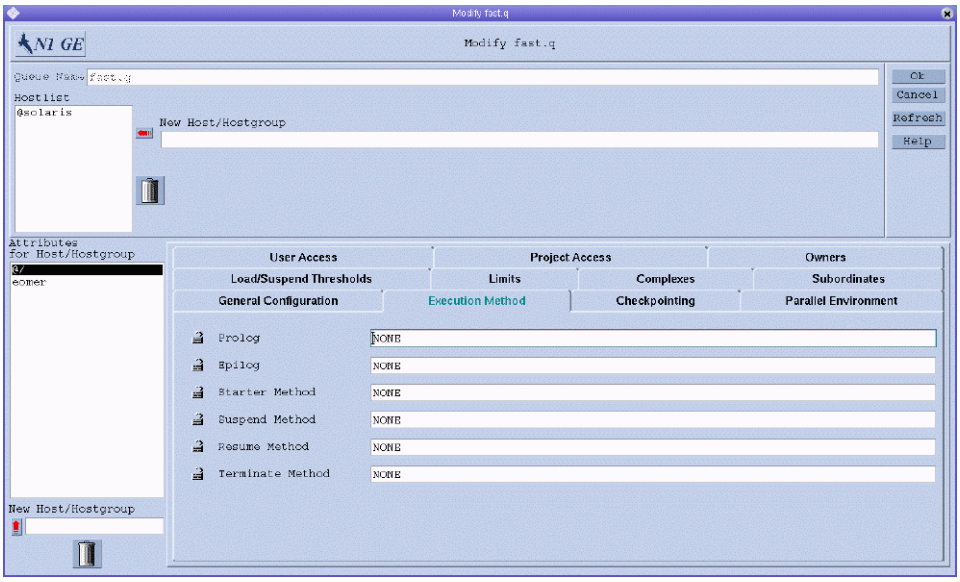
- `tmp` Directory. Temporary directory path.
- Shell. Default command interpreter to use for running the job scripts.

- 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 "Extended Job Example" in *N1 Grid Engine 6 User's Guide*.
- 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.

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

Configuring Execution Method Parameters

To configure execution method parameters, click the Execution Method tab. The Execution Method tab is shown in the following figure.



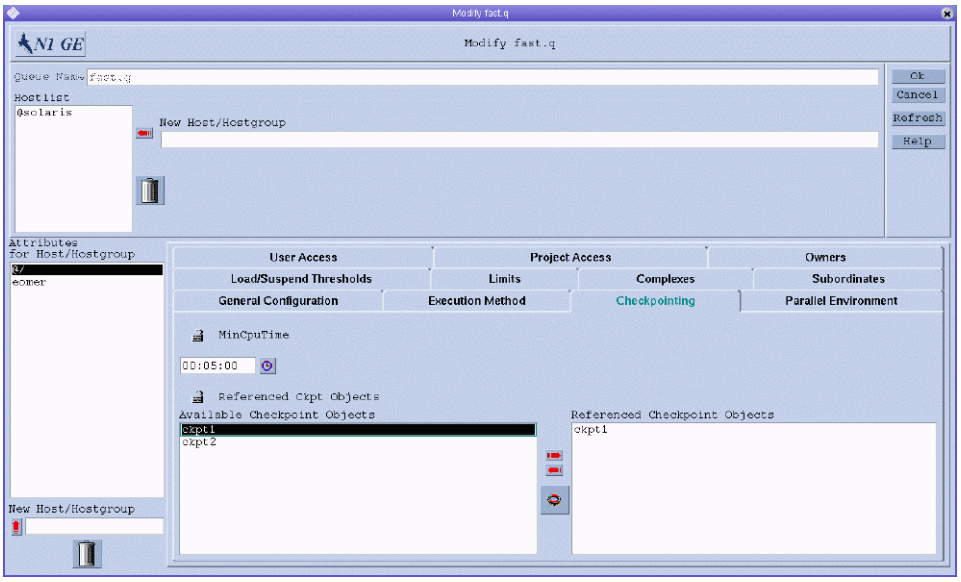
You can specify the following parameters:

- 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.

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

Configuring the Checkpointing Parameters

To configure the checkpointing parameters, click the Checkpointing tab. The Checkpointing tab is shown in the following figure.



You can specify the following parameters:

- MinCpuTime. The periodic checkpoint interval.
- Referenced Ckpt Objects. A list of checkpointing environments associated with the queue.

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.

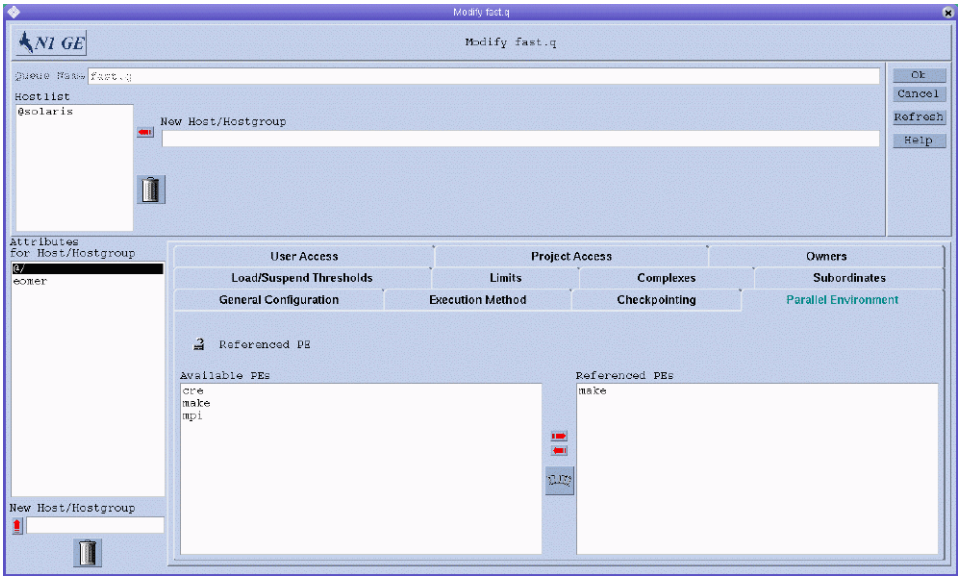
To remove a checkpointing environment from the Referenced list, select it, and then click the left arrow.

To add or modify checkpointing environments, click the button below the red arrows to open the Checkpointing Configuration dialog box. For more information, see "Configuring Checkpointing Environments With QMON" on page 166.

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

Configuring Parallel Environments

To configure parallel environments, click the Parallel Environment tab. The Parallel Environment tab is shown in the following figure.



You can specify the following parameter:

- Referenced PE. A list of parallel environments associated with the queue.

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.

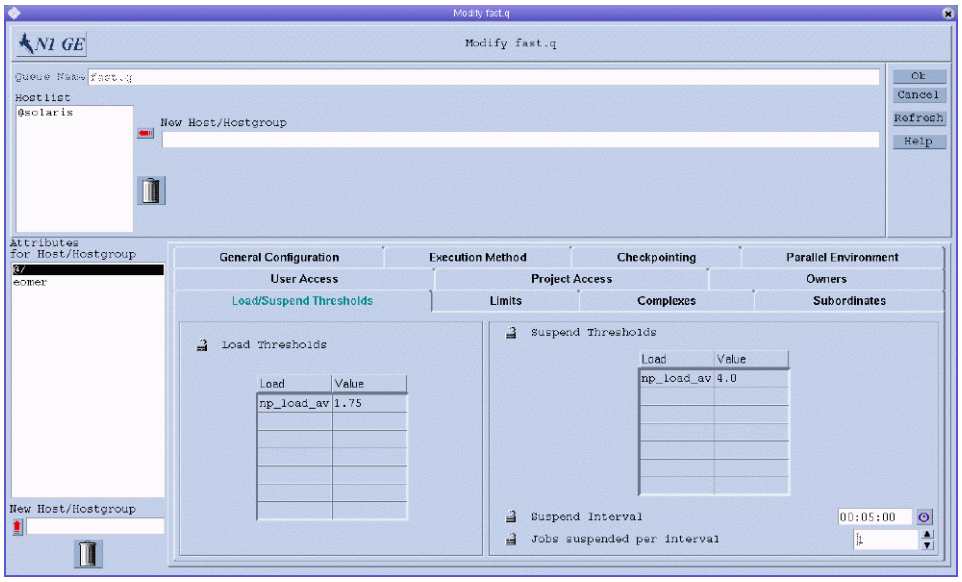
To remove a checkpointing environment from the Referenced PEs list, select it, and then click the left arrow.

To add or modify parallel environments, click the button below the red arrows to open the Parallel Environment Configuration dialog box. For more information, see "Configuring Parallel Environments With QMON" on page 156.

See the `queue_conf`(5) man page for detailed information about this parameter.

Configuring Load and Suspend Thresholds

To configure load and suspend thresholds, click the Load/Suspend Thresholds tab. The Load/Suspend Thresholds tab is shown in the following figure.



You can specify the following parameters:

- The Load Thresholds and the Suspend Thresholds tables, which define overload thresholds for load parameters and consumable resource attributes. See "Complex Resource Attributes" on page 67.

  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 in order to reduce the load.

  The tables display the currently configured thresholds.

  To change an existing threshold, select it, and then double-click the corresponding Value field.

  To add new thresholds, click Load or Value. A selection list appears with all valid attributes that are attached to the queue. The Attribute Selection dialog box is shown in Figure 1–2. To add an attribute to the Load column of the corresponding threshold table, select an attribute, and then click OK.

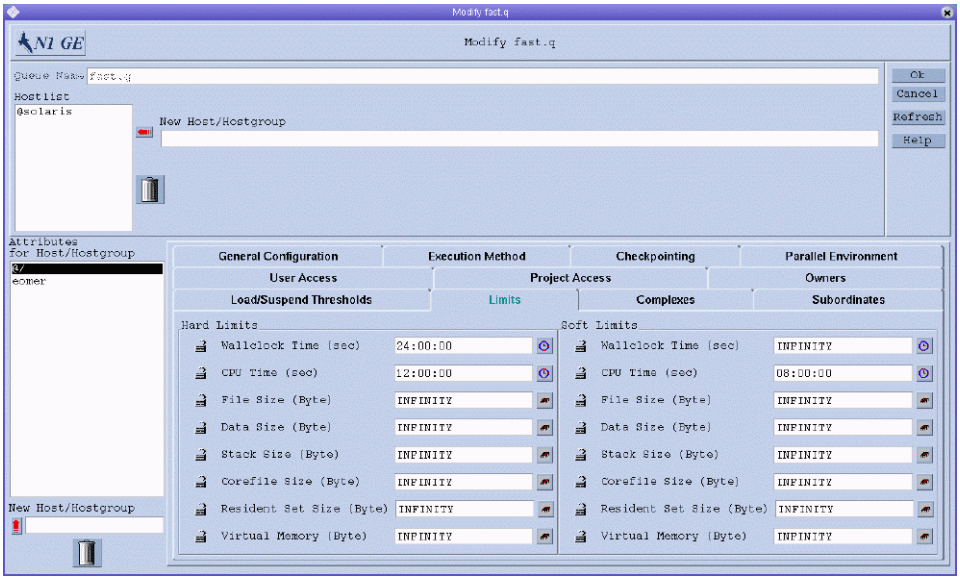
  To delete an existing threshold, select it, and then type Control-D or click mouse button 3. You are prompted to confirm that you want to delete the selection.

- 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.

Configuring Limits

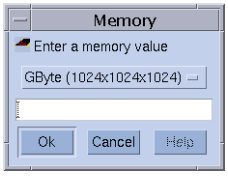
To configure limits parameters, click the Limits tab. The Limits tab is shown in the following figure.

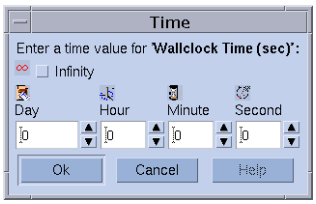


You can specify the following parameters:

- Hard Limit and Soft Limit. The hard limit and the soft limit to impose on the jobs that are running in the queue.

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.

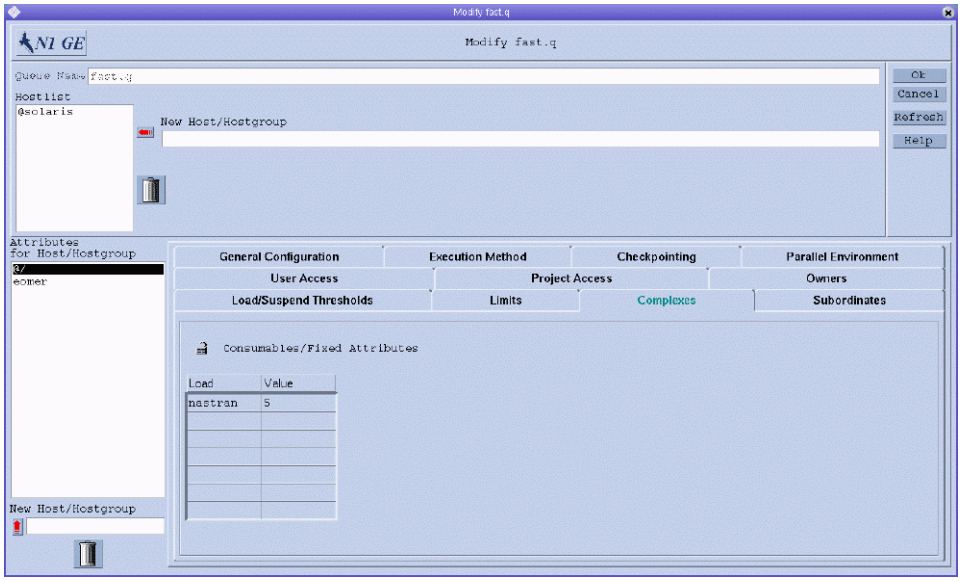




See the `queue_conf`(5) and the `setrlimit`(2) man pages for detailed information about limit parameters and their interpretation for different operating system architectures.

Configuring Complex Resource Attributes

To configure resource attributes, click the Complex tab. The Complex tab is shown in the following figure.



You can specify the following parameters:

- Consumables/Fixed Attributes. Value definitions for selected attributes from the set of resource attributes that are available for this queue.

  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 "Complex Resource Attributes" on page 67 for further details.

  The attributes for which values are explicitly defined are displayed in the Consumable/Fixed Attributes table. To change 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. The Attribute Selection dialog box is shown in Figure 1–2.

  To add an attribute to the Load column of the attribute table, select it, and then click OK.

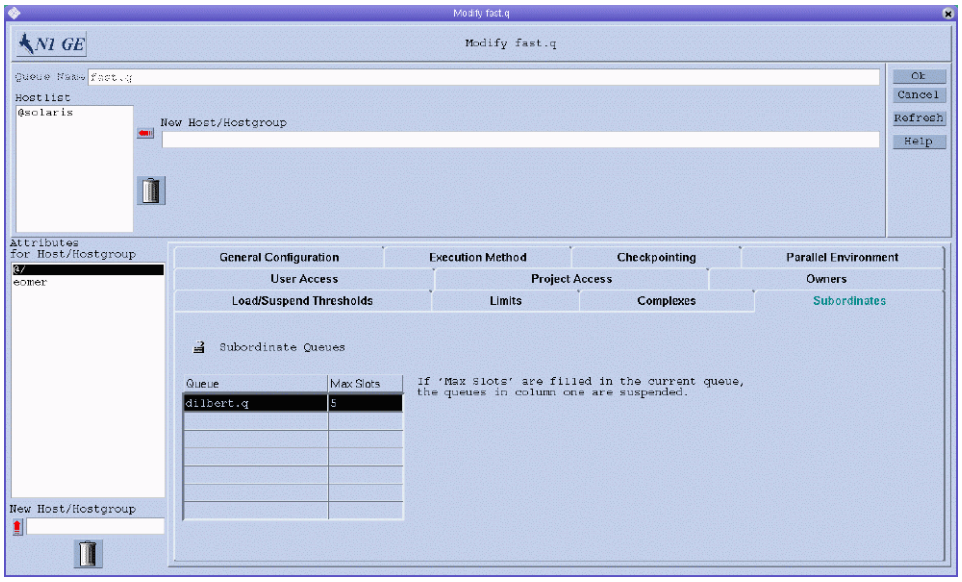
  To delete an attribute, select it, and then press Control-D or click mouse button 3. You are prompted to confirm that you want to delete the attribute.

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

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. See Figure 3–1 for an example.

Configuring Subordinate Queues

To configure subordinate queues, click the Subordinates tab. The Subordinates tab is shown in the following figure.



Use the subordinate queue facility to implement high priority and low priority queues as well as standalone queues.

You can specify the following parameters:

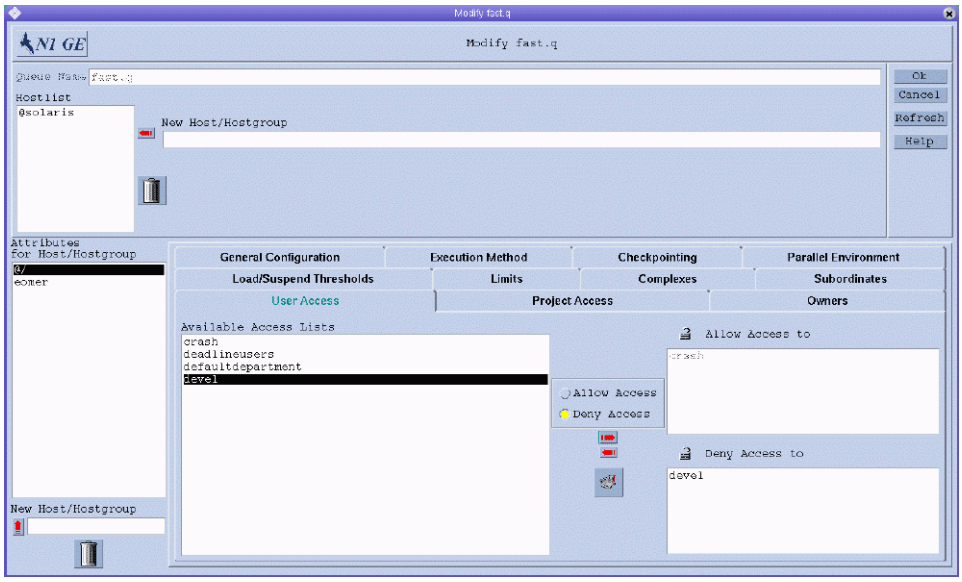
- Queue. A list of the queues that are *subordinated* to the configured queue.

  Subordinated queues are suspended if the configured queue becomes *busy*. Subordinated queues are resumed when the configured queue 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.

Configuring User Access Parameters

To configure user access parameters, click the User Access tab. The User Access tab is shown in the following figure.



You can specify the following parameters:

- Available Access Lists. The user access lists that can be included in the Allow Access list or the Deny Access list of the queue.

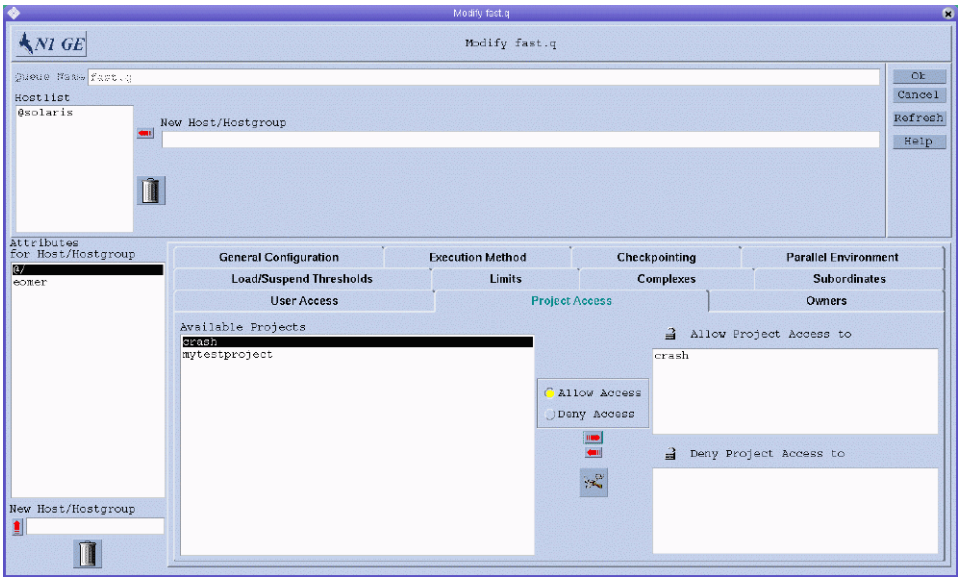
  Users or user groups belonging to access lists that are included in the Allow Access list have access to the queue. Users who are included in the Deny Access list cannot access the queue. If the Allow Access list is empty, access is unrestricted unless explicitly stated otherwise in the Deny Access list.

To add or modify user access lists, click the button between the Available Access Lists and the Allow Access and Deny Access lists to open the User Configuration dialog box. For more information, see "Configuring User Access Lists With QMON" on page 98.

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

Configuring Project Access Parameters

To configure project access parameters, click the Project Access tab. The Project Access tab is shown in the following figure.



You can specify the following parameters:

- Available Projects. The projects that are allowed access or denied access to the queue.

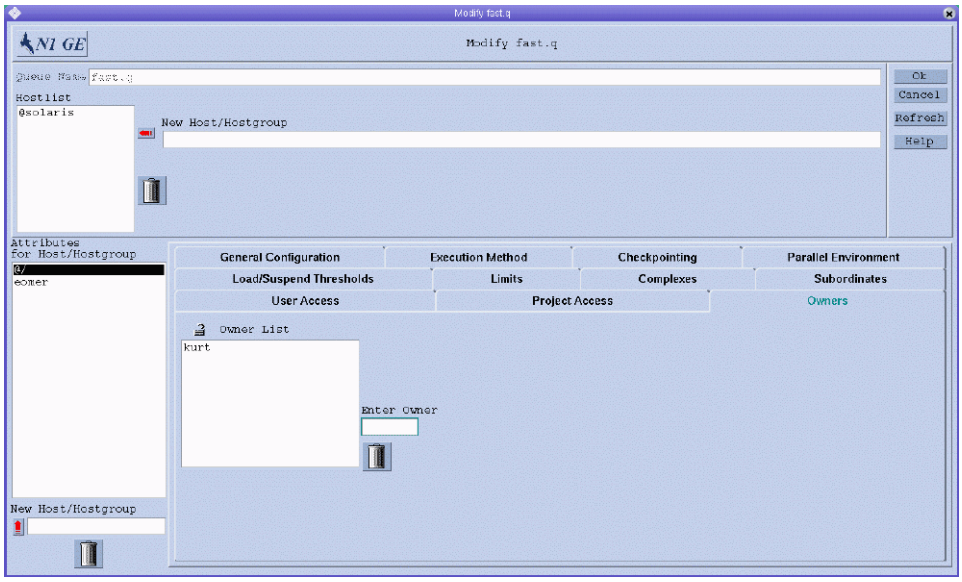
  Jobs submitted to a project belonging to the list of allowed projects have access to the queue. Jobs that are submitted to denied projects are not dispatched to the queue.

To add or modify project access, click the button between the Available Projects list and the Allow Project Access and Deny Project Access lists to open the Project Configuration dialog box. For more information, see "Defining Projects With QMON" on page 104.

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

Configuring Owners Parameters

To configure owners parameters, click the Owners tab. The Owners tab is shown in the following figure.



You can specify the following parameters:

- Owner List. The list of queue owners.

  Typically, users are set up to be owners of certain queue instances in order to allow them to suspend or disable jobs when they need to. 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.

  Queue owners can do the following:

  - **Suspend.** Stop execution of all jobs running in the queue and close the queue
  - **Resume.** Unsuspend the queue, and then open it
  - **Disable.** Close the queue, but do not affect running jobs
  - **Enable.** Open the queue

  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.

  All possible user accounts can be added to the owner list. To delete a user account from the queue owner list, select it, and then click the trash can icon.

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

Configuring Queues From the Command Line

To configure queues from the command line, type the following command with the appropriate options:

```
# qconf options
```

The `qconf` command has the following options:

- `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.

- `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.

- `qconf -cq` *queue*`[,...]`

  The `-cq` option (clean queue) cleans the status of the specified cluster queues, queue domains, or queue instances to be idle and free from running jobs. The status is reset without respect to the current status. This option is useful for eliminating error conditions, but you should not use it in normal operation mode.

- `qconf -dq` *cluster-queue*`[,...]`

  The `-dq` option (delete cluster queue) deletes the cluster queues specified in the argument list from the list of available queues.

- `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.

- `qconf -Mq` *filename*

  The `-Mq` option (modify cluster queue from file) uses the file *filename* to define the modified cluster queue configuration. The definition file might have been produced by the `qconf -sq` *queue* command and subsequent modification.

- `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.

- `qconf -sql`

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

The `qconf` command provides the following set of options that you can use to change specific queue attributes:

`-aattr` – Add attributes
`-Aattr` – Add attributes from a file
`-dattr` – Delete attributes
`-Dattr` – Delete attributes listed in a file
`-mattr` – Modify attributes
`-Mattr` – Modify attributes from a file
`-rattr` – Replace attributes
`-Rattr` – Replace attributes from a file
`-sobjl` – Show list of configuration objects

For a description of how to use these options and for some examples of their use, see "Using Files to Modify Queues, Hosts, and Environments" on page 180. For detailed information about these options, see the `qconf`(1) man page.

Configuring Queue Calendars

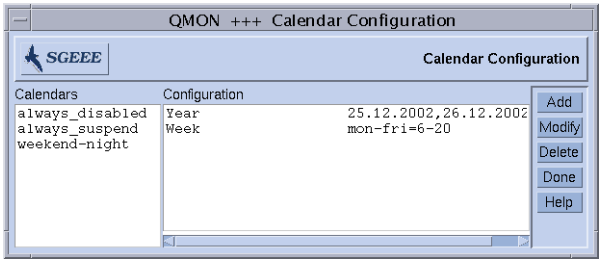
Queue calendars define the availability of queues according to the day of the year, the day of the week, or the time of day. You can configure queues to change their status at specified times. You can change the queue status to disabled, enabled, suspended, or resumed (unsuspended).

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.

Configuring Queue Calendars With `QMON`

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



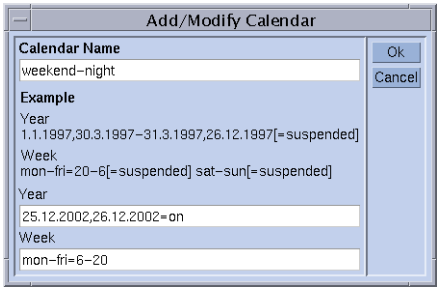
The Calendars list displays the available calendars.

In the Calendars list, click the calendar configuration that you want to modify or delete.

Do one of the following:

- To delete the selected calendar, click Delete.
- To modify the selected calendar, click Modify.
- To add access lists, click Add.

In all cases, the Add/Modify Calendar dialog box appears.



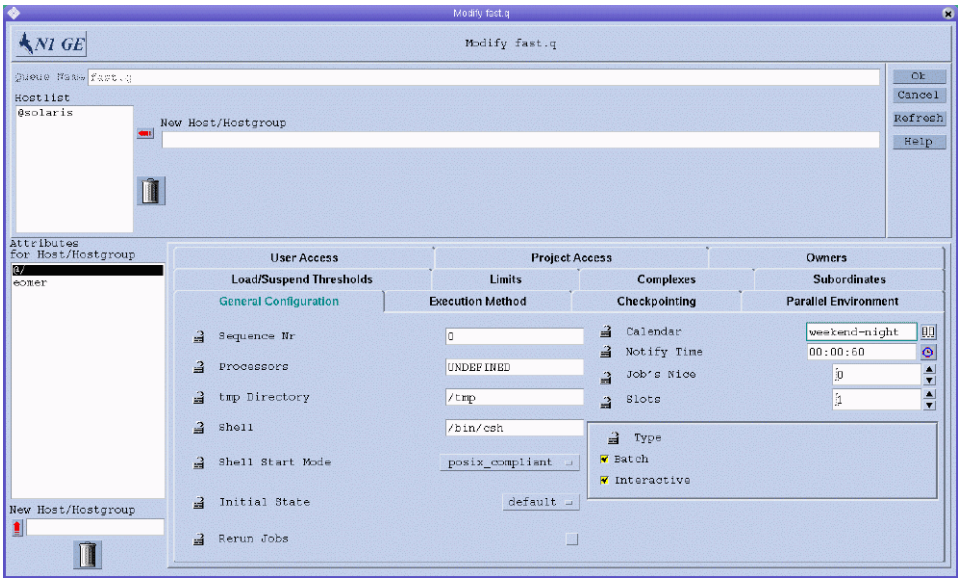
If you click Modify or Delete, the Calendar Name field displays the name of the selected calendar. If you click Add, type the name of the calendar you are defining.

The Year and Week fields enable you to define the calendar events, using the syntax described in the `calendar_conf`(5) man page.

The example of the calendar configuration shown in the previous figure is appropriate for queues that should be available outside office hours and on weekends. In addition, the Christmas holidays are defined to be handled like weekends.

See the `calendar_conf`(5) man page for a detailed description of the syntax and for more examples.

By attaching a calendar configuration to a queue, the availability profile defined by the calendar is set for the queue. Calendars are attached in the General Configuration tab of the Modify *queue-name* dialog box. The Calendar field contains the name of the calendar to attach. The button next to the Calendar field lists the currently configured calendars. See "Configuring Queues" on page 45 for more details about configuring queues.



Configuring Queue Calendars From the Command Line

To configure queue calendars from the command line, type the following command with appropriate options:

```
% qconf options
```

The following options are available:

- `qconf -acal` *calendar-name*

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

- `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.

- `qconf -dcal` *calendar-name* `[,...]`

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

- `qconf -mcal` *calendar-name*

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

- `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.

- `qconf -scal` *calendar-name*

  The `-scal` option (show calendar) displays the configuration for *calendar-name*.

- `qconf -scall`

  The`-scall` option (show calendar list) displays a list of all configured calendars.

CHAPTER 3

Configuring Complex Resource Attributes

This chapter describes how to configure resource attribute definitions. Resource attribute definitions are stored in an entity called the grid engine system *complex*. In addition to background information relating to the complex and its associated concepts, this chapter provides detailed instructions on how to accomplish the following tasks:

Complex Resource Attributes

The complex configuration provides all pertinent information about the *resource attributes* users can request for jobs with the `qsub -l` or `qalter -l` commands. The complex configuration also provides information about how the grid engine system should interpret these resource attributes.

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. 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 oversubscription of consumable resources.

Typical consumable resource attributes include:

- Available free memory
- Unoccupied licenses of a software package

- Free disk space
- Available bandwidth on a network connection

Attribute definitions in the grid engine complex define how resource attributes should be interpreted.

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` or `TIME`
- 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

Use the `QMON` Complex Configuration dialog box, which is shown in Figure 3–1, to define complex resource attributes.

Configuring Complex Resource Attributes With QMON

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

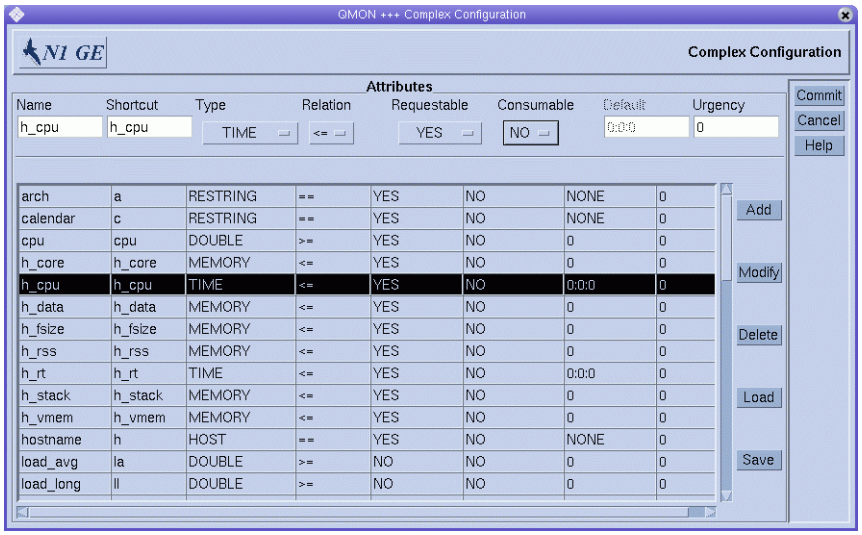


FIGURE 3–1 Complex Configuration Dialog Box

The Complex Configuration dialog box enables you to add, modify, or delete complex resource attributes.

To add a new attribute, first make sure that no line in the Attributes table is selected. In the fields above the Attributes table, type or select the values that you want, and then click Add.

Note – If you want to add a new attribute and an existing attribute is selected, you must clear the selection. To deselect a highlighted attribute, hold down the Control key and click mouse button 1.

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.

To modify an attribute listed in the Attributes table, select it. The values of the selected attribute are displayed above the Attributes table. Change the attribute values, and then click Modify.

To save configuration changes to a file, click Save. To load values from a file into the complex configuration, click Load, and then select the name of a file from the list that appears.

To delete an attribute in the Attribute table, select it, and then click Delete.

See the `complex`(5) man page for details about the meaning of the rows and columns in the table.

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

Assigning Resource Attributes to Queues, Hosts, and the Global Cluster

Resource attributes can be used in the following ways:

- As queue resource attributes
- As host resource attributes
- As global resource attributes

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.

User-defined resource attributes must first be defined in the complex before you can assign them to a queue instance, a host, or the global cluster. When you assign a resource attribute to one of these targets, you specify a value for the attribute.

The following sections describe each attribute type in detail.

Queue Resource Attributes

Default queue resource attributes are a set of parameters that are defined in the queue configuration. These parameters are described in the `queue_conf`(5) man page.

You can add new resource attributes to the default attributes. New attributes are attached only to the queue instances that you modify. When the configuration of a particular queue instance references a resource attribute that is defined in the complex, that queue configuration provides the values for the attribute definition. For details about queue configuration see "Configuring Queues" on page 45.

For example, the queue configuration value `h_vmem` is used for the virtual memory size limit. This value limits the amount of total memory that each job can consume. An entry in the `complex_values` list of the queue configuration defines the total available amount of virtual memory on a host or assigned to a queue. For detailed information about consumable resources, see "Consumable Resources" on page 74.

Host Resource Attributes

Host resource attributes are parameters that are intended to be managed on a host basis.

The default host-related attributes are load values. You can add new resource attributes to the default attributes, as described earlier in "Queue Resource Attributes" on page 70.

Every `sge_execd` periodically reports load to `sge_qmaster`. 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 "Load Parameters" on page 87.

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" on page 24.

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 oversubscription, which often results in reduced system performance that is caused by *swapping*. For more information about consumable resources, see "Consumable Resources" on page 74.

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

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 "Load Parameters" on page 87. 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" on page 24.
- 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.

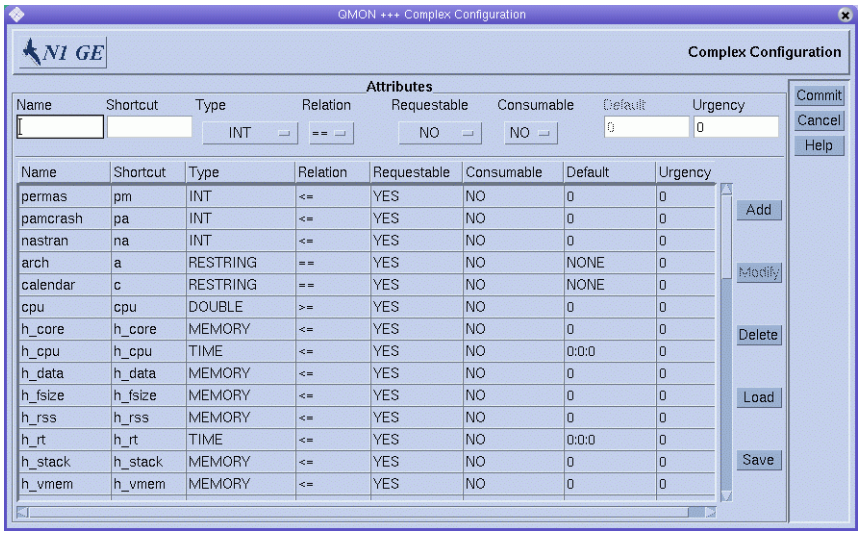
Adding Resource Attributes to the Complex

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

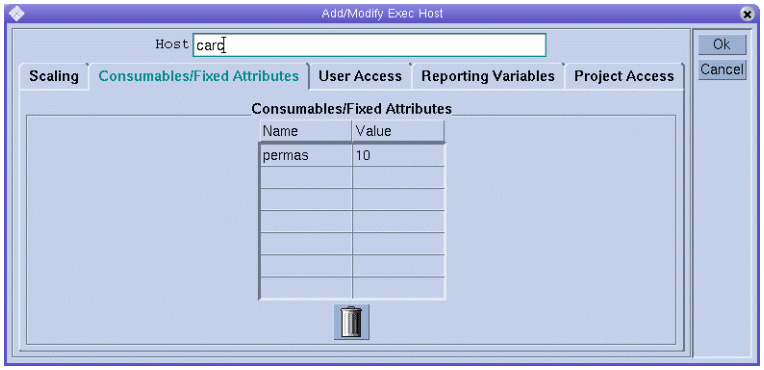
User-defined attributes are a named collection of attributes with the corresponding definitions as to how the grid engine software is to handle these attributes. You can attach one or more user-defined attributes to a queue, to a host, or globally to all hosts in the cluster. Use the `complex_values` parameter for the queue configuration and the host configuration. For more information, see "Configuring Queues" on page 45 and "Configuring Hosts" on page 24. 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.

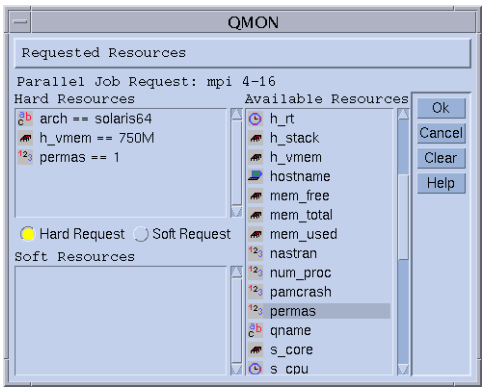
For example, say the user-defined resource attributes `permas`, `pamcrash`, and `nastran`, shown in the following figure, are defined.



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" on page 45 and its related sections.



Then the displayed queue is configured to manage up to 10 licenses of the software package `permas`. Furthermore, the attribute `permas` becomes requestable for jobs, as expressed in the Available Resources list in the Requested Resources dialog box.



For details about how to submit jobs, see Chapter 3, "Submitting Jobs," in *N1 Grid Engine 6 User's Guide*.

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

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

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

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.

Consumable Resources

Consumable resources provide an efficient way to manage limited resources such as available memory, free space on a file system, network bandwidth, or floating software licenses. Consumable resources are also called *consumables*. The total available capacity of a consumable is defined by the administrator. The consumption of the corresponding resource is monitored by grid engine software internal bookkeeping. The grid engine system accounts for the consumption of this resource for all running jobs. Jobs are dispatched only if the internal bookkeeping indicates that sufficient consumable resources are available.

Consumables can be combined with default load parameters or user-defined load parameters. Load values can be reported for consumable attributes. Conversely, the Consumable flag can be set for load attributes. Load measures the availability of the resource. Consumable resource management takes both the load and the internal bookkeeping into account, ensuring that neither exceeds a given limit. For more information about load parameters, see "Load Parameters" on page 87.

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.

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" on page 45 and "Configuring Hosts" on page 24.

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.

A parallel job consumes as many consumable resources as it consumes job slots. For example, the following command consumes a total of 800 Mbytes of memory:

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

Memory 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.

Setting Up 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, as shown in Figure 3–1.

To enable the consumable management for an attribute, set the Consumable flag for the attribute in the complex configuration. For example, the following figure shows that the Consumable flag is set for the `virtual_free` memory resource.

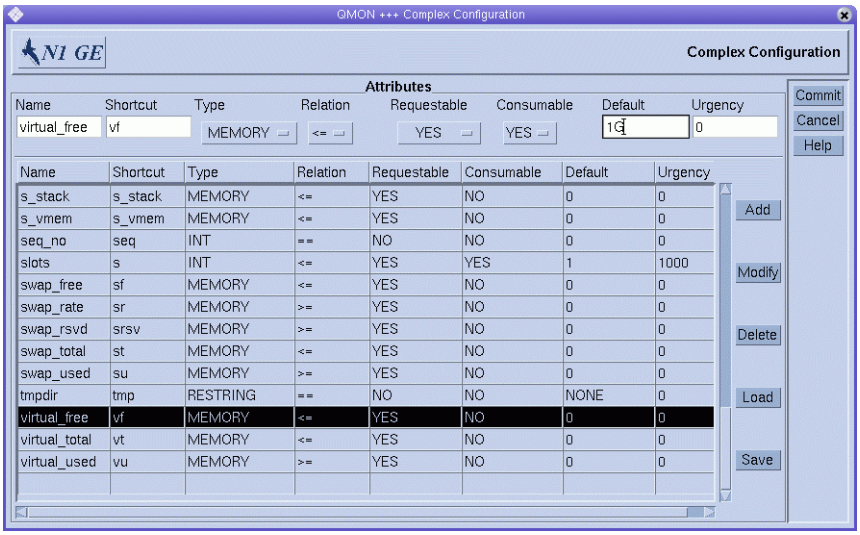


FIGURE 3–2 Complex Configuration Dialog Box: `virtual_free`

Set up other consumable resources, guided by the examples detailed in the following sections:

- "Example 1: Floating Software License Management" on page 77
- "Example 2: Space Sharing for Virtual Memory" on page 81
- "Example 3: Managing Available Disk Space" on page 84

Then, for each queue or host for which you want the grid engine software to do the required capacity planning, you must define the capacity in a `complex_values` list. An example is shown in the following figure, where 1 Gbyte of virtual memory is defined as the capacity value of the current host.

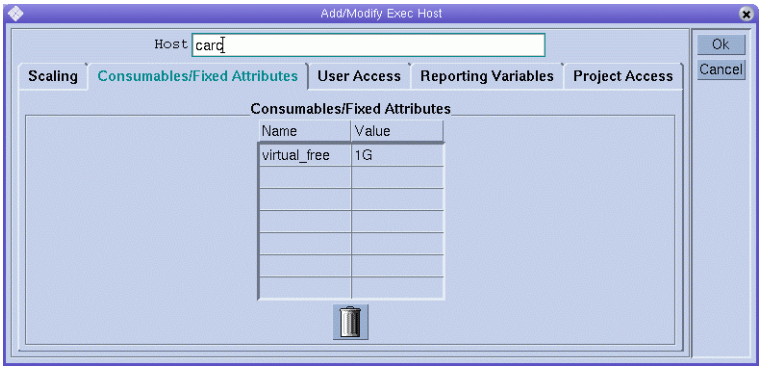


FIGURE 3–3 Add/Modify Exec Host: `virtual_free`

The virtual memory requirements of all jobs running concurrently in any queue on that host are accumulated. The requirements are then subtracted from the capacity of 1 Gbyte to determine available virtual memory. If a job request for `virtual_free` exceeds the available amount, the job is not dispatched to a queue on that host.

Note – Jobs can be forced to request a resource and thus to specify their assumed consumption through the `FORCED` value of the Requestable parameter.

For consumable attributes that are not explicitly requested by the job, the administrator can predefine a default value for resource consumption. Doing so is meaningful only if requesting the attribute is not forced, as explained in the previous note. 200 Mbytes is set as the default value.

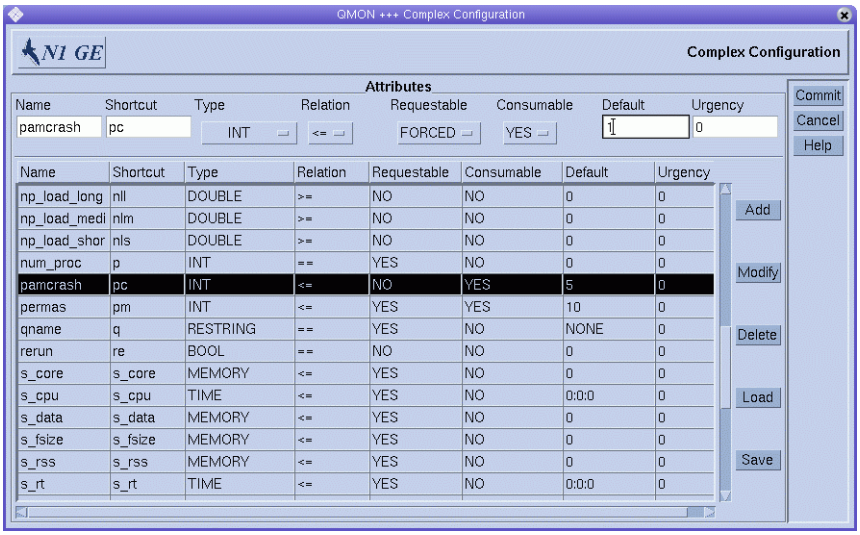
Examples of Setting Up Consumable Resources

Use the following examples to guide you in setting up consumable resources for your site.

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 in a way that prevents 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.



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.

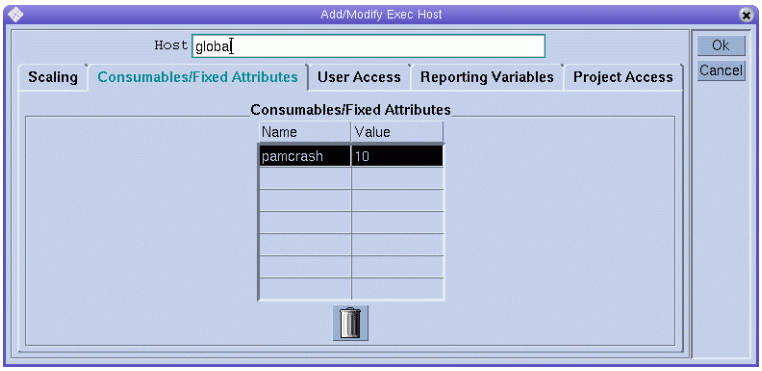
The Requestable flag is set to `FORCED`. This setting specifies that users must request how many `pam-crash` licenses that their job will occupy when the job is submitted.

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" on page 45 and "Configuring Hosts" on page 24.

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.



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

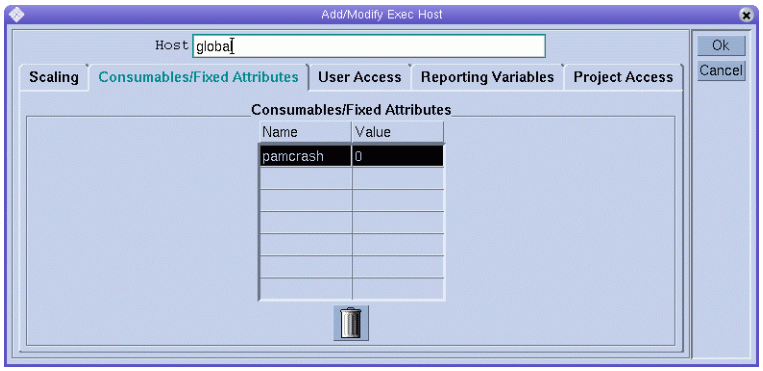
Note – The table `Consumables/Fixed Attributes` corresponds to the `complex_values` entry that is described in the host configuration file format `host_conf`(5).

Assume that a user submits the following job:

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

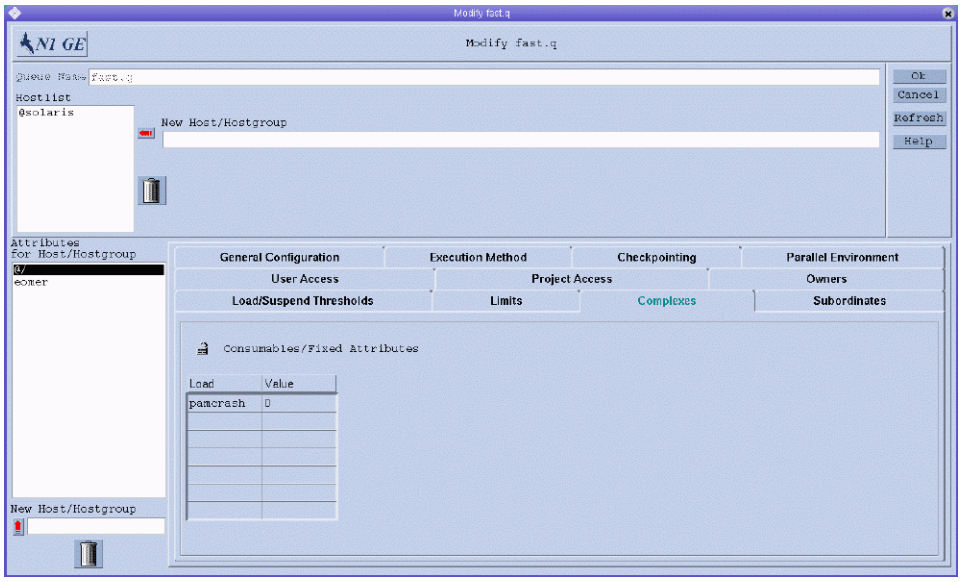
The job starts only if fewer than 10 `pam-crash` licenses are currently occupied. The job can run anywhere in the cluster, however, and the job occupies one `pam-crash` license throughout its run time.

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`. Use the Execution Host tab of the Host Configuration dialog box.



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.



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 oversubscription, 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.

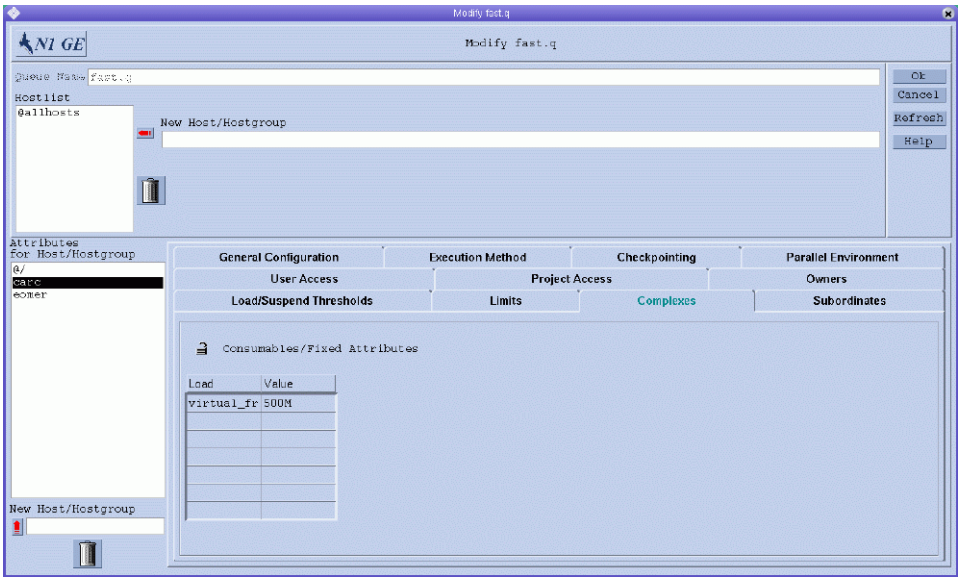
An example of a possible `virtual_free` resource definition is shown in Figure 3–2. A corresponding execution host configuration for a host with 1 Gbyte of main memory is shown in Figure 3–3.

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 oversubscription. 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 different job classes with different memory requirements on one machine, you might want to partition the memory that these job classes use. This functionality is called *space sharing*. You can accomplish this functionality by configuring a queue for each job class. 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.



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
```

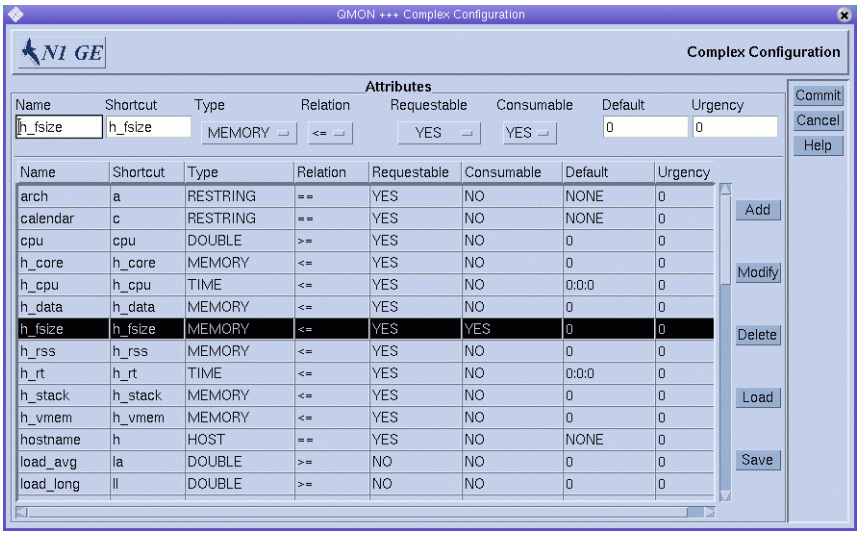
The job submitted by the first command can be started as soon as at least 100 Mbytes of memory are available. This amount is taken into account in the capacity planning for the `virtual_free` consumable resource. The second job runs only if no other job is on the system, as the second job implicitly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

Example 3: Managing Available Disk Space

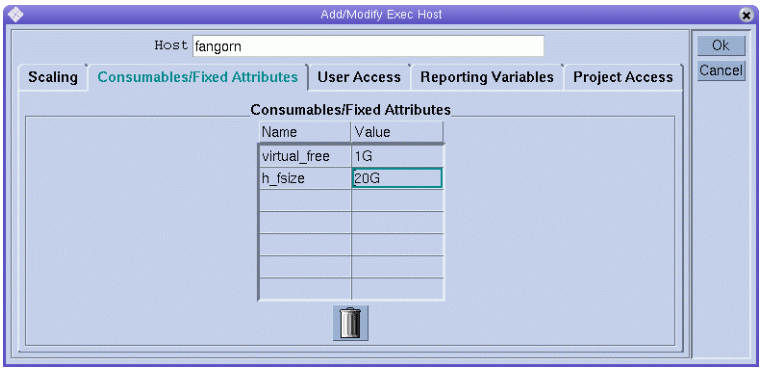
Some applications need to manipulate huge data sets stored in files. Such applications therefore depend on the availability of sufficient disk space throughout their run time. This requirement is similar to the space sharing of available memory, as discussed in the preceding example. The main difference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



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.



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 reason the `h_fsize` attribute is recommended here is that `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, which indicates total disk space consumption, including the one occurring outside 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" on page 87 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.

Configuring Complex Resource Attributes From the Command Line

To configure the complex from the command line, type the following command with appropriate options:

```
% qconf options
```

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

The following options enable you to modify the grid engine system complex:

- `-mc` – The `-mc` option opens an editor filled in with a template complex configuration or with an existing complex configuration for modification.
- `-Mc` – The `qconf -Mc` option takes a complex configuration file as an argument.

The following command prints the current complex configuration to the standard output stream in the file format defined in the `complex`(5) man page:

```
% qconf -sc
```

A sample output is shown in the following example.

EXAMPLE 3–1 `qconf -sc` Sample Output

```
#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 -----------
```

Load Parameters

This section explains the grid engine system's load parameters. Instructions are included for writing your own load sensors.

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" on page 20. 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" on page 70 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 N1 Grid Engine 6 software may 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. Therefore 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" on page 87 for more information.

Writing Your Own Load Sensors

To feed `sge_execd` with additional load information, you must supply a *load sensor*. 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`.

Example of a Load Sensor Script

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

EXAMPLE 3–2 Load Sensor – Bourne Shell Script

```
#!/bin/sh
```

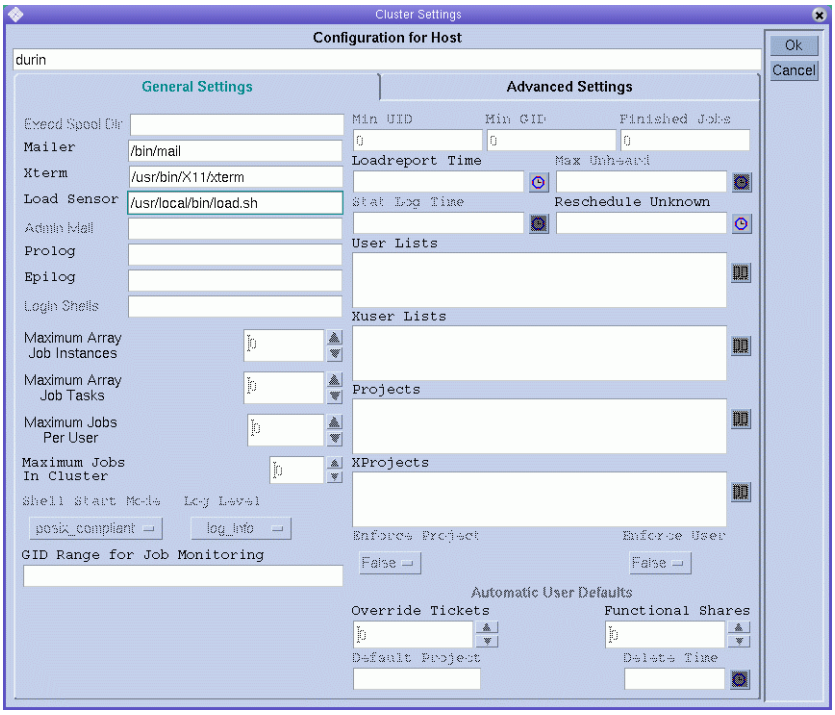
EXAMPLE 3–2 Load Sensor – Bourne Shell Script *(Continued)*

```
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
```

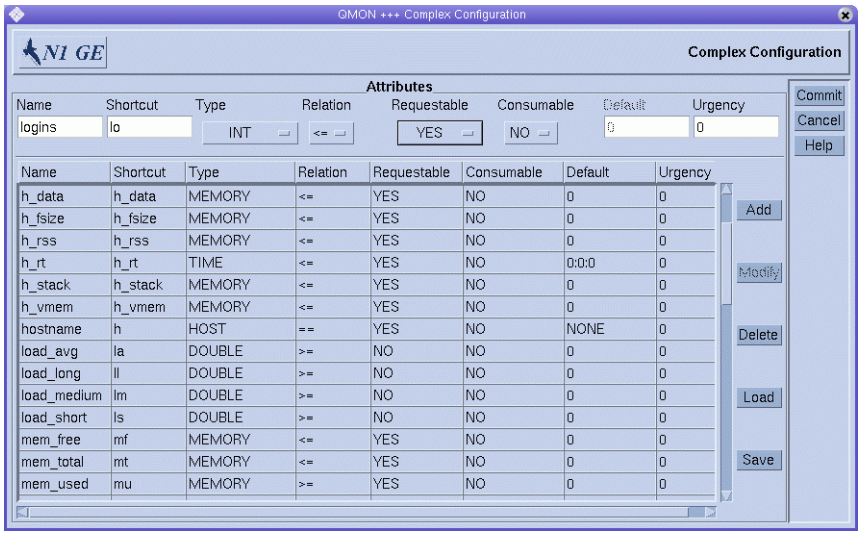
Save this script to the file `load.sh`. Assign executable permission to the file with the `chmod` command. To test the script interactively from the command line, type `load.sh` and repeatedly press the Return key.

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 "Basic Cluster Configuration" on page 40 or the `sge_conf`(5) man page for more information.

The corresponding `QMON` window might look like the following figure:



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.


QMON +++ Complex Configuration
N1 GE
Complex Configuration
Attributes
Name
Shortcut
Type
Relation
Requestable
Consumable
Default
Urgency
logins
lo
INT
<=
YES
NO
0
Commit
Cancel
Help
Name Shortcut Type Relation Requestable Consumable Default Urgency
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
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
mem_used mu MEMORY >= YES NO 0 0
Add
Modify
Delete
Load
Save

CHAPTER 4

Managing User Access

This chapter contains information about managing user accounts and other related accounts. Topics in this chapter include the following:

- User access
- Projects and project access
- Path-aliasing
- Default requests

In addition to the background information, this chapter includes detailed instructions on how to accomplish the following tasks:

Setting Up a User

You need to perform the following tasks to set up a user for the grid engine system:

- **Assign required logins.**

  To submit jobs from host A for execution on host B, users must have identical accounts on both hosts. The accounts must have identical user names. No login is required on the machine where `sge_qmaster` runs.

- **Set access permissions.**

  The grid engine software enables you to restrict user access to the entire cluster, to queues, and to parallel environments. See "Configuring Users" on page 101 for a detailed description.

  In addition, you can grant users permission to suspend or enable certain queues. See "Configuring Owners Parameters" on page 60 for more information.

- **Declare a Grid Engine System user.**

  In order 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 "Configuring Policy-Based Resource Management With `QMON`" on page 127 and "Configuring User Objects With `QMON`" on page 101.

- **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 "Configuring Policy-Based Resource Management With `QMON`" on page 127 for more information.

- **Set file access restrictions.**

  Users of the grid engine system must have read access to the directory *sge-root*/*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 Chapter 6, "Verifying the Installation," in *N1 Grid Engine 6 Installation Guide*.

Because batch jobs are usually run off line, only two ways exist to notify a job owner about error events and the like. One way is to log the error messages to a file, the other way is to send email.

Under some rare circumstances, for example, if the error log file can't be opened, email is the only way to directly notify the user. Error messages are logged to the grid engine system log file anyway, but usually the user would not look at the system log file. 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 "Customizing QMON" in *N1 Grid Engine 6 User's Guide*.
  - `sge_aliases` – Aliases for the path to the current working directory. See "Using Path Aliasing" on page 106.
  - `sge_request` – Default request definition file. See "Configuring Default Requests" on page 108.

Configuring User Access

The grid engine system has the following four categories of users:

- **Managers.** Managers have full capabilities to manipulate the grid engine system. By default, the superusers of the master host and of any machine that hosts a queue instance have manager privileges.
- **Operators.** Operators can perform many of the same commands as managers, except that operators cannot add, delete, or modify queues.
- **Owners.** Queue owners are restricted to suspending and resuming, or disabling and enabling, the queues that they own. 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.
- **Users.** Users have certain access permissions, as described in "Configuring Users" on page 101, but users have no cluster or queue management capabilities.

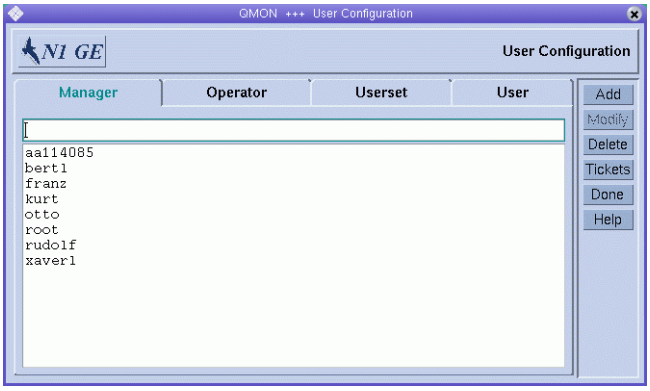
The following sections describe each category in more detail.

Configuring Manager Accounts

You can configure Manager accounts with `QMON` or from the command line.

Configuring Manager Accounts With QMON

On the QMON Main Control window, click the User Configuration button. The Manager tab appears, which enables you to declare which accounts are allowed to run any administrative command.



This tab lists all accounts that are already declared to have administrative permission.

To add a new manager account, type its name in the field above the manager account list, and then click Add or press the Return key.

To delete a manager account, select it, and then click Delete.

Configuring Manager Accounts From the Command Line

To configure a manager account from the command line, type the following command with appropriate options:

```
# qconf options
```

The following options are available:

- `qconf -am` *user-name* [,...]

  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" on page 20 for more information.

- `qconf -dm` *user-name* [,...]

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

- `qconf -sm`

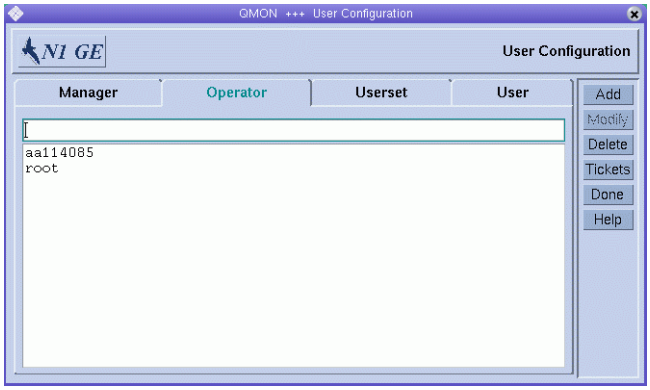
The `-sm` option (show managers) displays a list of all grid engine system managers.

Configuring Operator Accounts

You can configure operator accounts with `QMON` or from the command line.

Configuring Operator Accounts With `QMON`

On the `QMON` Main Control window, click the User Configuration button, and then click the Operator tab.



The Operator tab enables you to declare which accounts are allowed to have *restricted* administrative permission, unless the accounts are also declared to be manager accounts. See "Configuring Manager Accounts With `QMON`" on page 96.

This tab lists all accounts that are already declared to have operator permission.

To add a new operator account, type its name in the field above the operator account list, and then click Add or press the Return key.

To delete an operator account, select it, and then click Delete.

Configuring Operator Accounts From the Command Line

To configure an operator account from the command line, type the following command with appropriate options:

```
# qconf options
```

The following options are available:

- `qconf -ao` *user-name*`[,...]`

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

- `qconf -do` *user-name*`[,...]`

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

- `qconf -so`

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

Configuring User Access Lists

Any user with a valid login ID on at least one submit host and one execution host can use the grid engine system. However, grid engine system managers can prohibit access for certain users to certain queues or to all queues. Furthermore, managers can restrict the use of facilities such as specific parallel environments. See "Configuring Parallel Environments" on page 155 for more information.

In order to define access permissions, you must define *user access lists*, which are made up of named sets of users. You use user names and UNIX group names to define user access lists. The user access lists are then used either to deny or to allow access to a specific resource in any of the following configurations:

- Cluster configuration – see "Basic Cluster Configuration" on page 40
- Queue configuration – see "Configuring Subordinate Queues" on page 57
- Configuring of parallel environment interfaces – see "Configuring Parallel Environments With `QMON`" on page 156.

Configuring User Access Lists With `QMON`

On the `QMON` Main Control window, click the User Configuration button, and then click the Userset tab. The Userset tab appears.

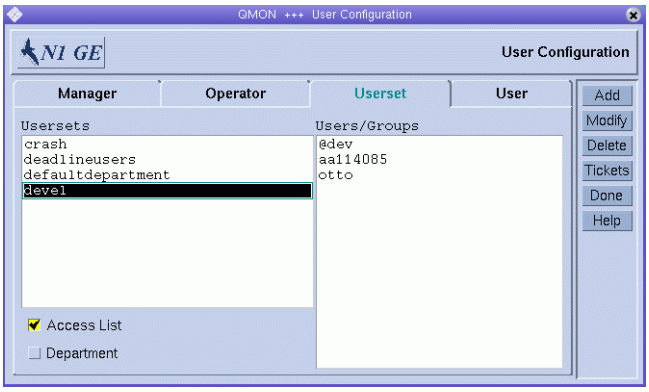


FIGURE 4–1 Userset Tab

In the grid engine system, a userset can be either an Access List or a Department, or both. The two check boxes below the Usersets list indicate the type of the selected userset. This section describes access lists. Departments are explained in "Defining Usersets As Projects and Departments" on page 101.

The Usersets lists displays all available access lists. To display the contents of an access list, select it. The contents are displayed in the Users/Groups list.

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

To add a new userset, click Add.

To modify an existing userset, select it, and then click Modify.

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

When you click Add or Modify, an Access List Definition dialog box appears.

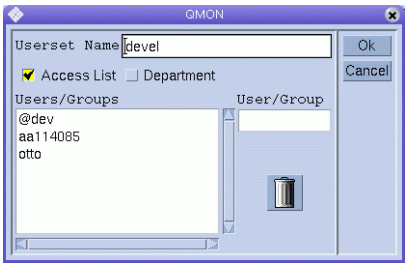


FIGURE 4–2 Access List Definition Dialog Box

To add a new access list definition, type the name of the access list in the Userset Name field. If you are modifying an existing access list, its name is displayed in the Userset Name field.

To add a new user or group to the access list, type a user or group name in the User/Group field. Be sure to prefix group names with an @ sign.

The Users/Groups list displays all currently defined users and groups.

To delete a user or group from the Users/Groups list, select it, and then click the trash icon.

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

Configuring User Access Lists From the Command Line

To configure user access lists from the command line, type the following command with appropriate options.

```
# qconf options
```

The following options are available:

- `qconf -au` *user-name*`[,...]` *access-list-name*`[,...]`

  The `-au` option (add user) adds one or more users to the specified access lists.

- `qconf -Au` *filename*

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

- `qconf -du` *user-name*`[,...]` *access-list-name* `[,...]`

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

- `qconf -dul` *access-list-name*`[,...]`

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

- `qconf -mu` *access-list-name*

  The `-mu` option (modify user access list) modifies the specified access lists.

- `qconf -Mu` *filename*

  The `-Mu` option (modify user access list from file) uses a configuration file, *filename*, to modify the specified access lists.

- `qconf -su` *access-list-name*`[,...]`

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

- `qconf -sul`

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

Defining Usersets As Projects and Departments

Usersets are also used to define grid engine system projects and departments. For details about projects, see "Defining Projects" on page 103.

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" on page 147 and "Configuring the Override Policy" on page 151.

A Userset is identified as a department by the Department flag, which is shown in Figure 4–1 and Figure 4–2. 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.

Configuring Users

You must declare user names before you define the share-based, functional, or override policies for users. See "Configuring Policy-Based Resource Management With QMON" on page 127.

If you do not want to explicitly declare user names before you define policies, the grid engine system can automatically create users for you, based on predefined default values. The automatic creation of users can significantly reduce the administrative burden for sites with many users.

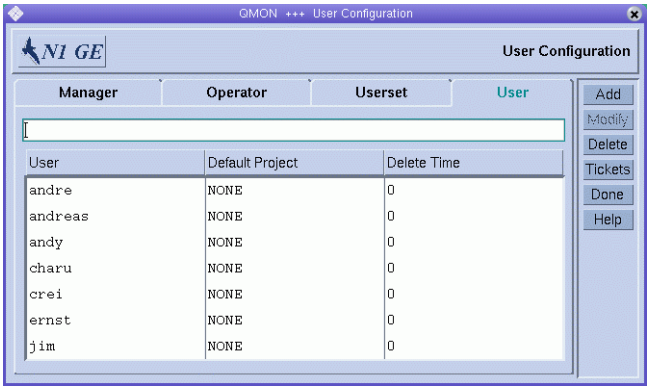
To have the system create users automatically, set the Enforce User parameter on the Cluster Settings dialog box to `Auto`. To set default values for automatically created users, specify values for the following Automatic User Defaults on the Cluster Settings dialog box:

- Override Tickets
- Functional Shares
- Default Project
- Delete Time

For more information about the cluster configuration, see "Basic Cluster Configuration" on page 40.

Configuring User Objects With QMON

On the QMON Main Control window, click the User Configuration button, and then click the User tab. The User tab looks like the following figure:



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.

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. For details about projects, see "Defining Projects" on page 103.

To assign a default project, select a user, and then click the Default Project column heading. A Project Selection dialog box appears.

Select a project for the highlighted user entry.

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 User Objects From the Command Line

To configure user objects from the command line, type the following command with appropriate options:

```
# qconf options
```

The following options are available:

- `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`.

- `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.

- `qconf -duser` *user-name*[,...]

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

- `qconf -muser` *user-name*

  The `-muser` option (modify user) enables you to modify an existing user entry. The option loads the user 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`.

- `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.

- `qconf -suser` *user-name*

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

- `qconf -suserl`

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

Defining 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 are used in three scheduling policy areas:

- Share-based, when shares are assigned to projects – see "Configuring the Share-Based Policy" on page 135
- Functional, when projects receive a percentage of the functional tickets – see "Configuring the Functional Policy" on page 147
- Override, when an administrator grants override tickets to a project – see "Configuring the Override Policy" on page 151

Projects must be declared before they can be used in any of the three policies.

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.

Defining Projects With QMON

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

To define a project, on the QMON Main Control window, click the Project Configuration button. The Project Configuration dialog box appears.

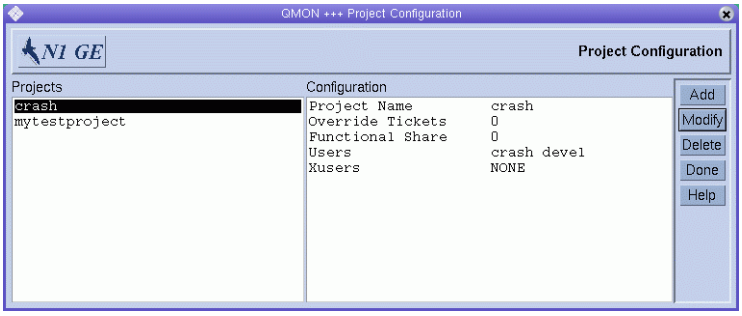


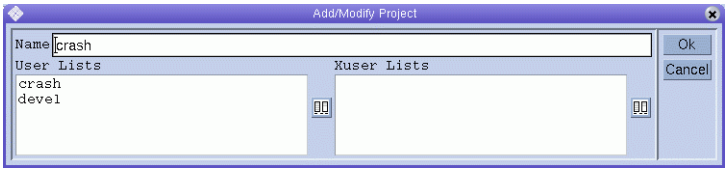
FIGURE 4–3 Project Configuration Dialog Box

The currently defined projects are displayed in the Projects list.

The project definition of a selected project is displayed under Configuration.

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

To add a new project, click Add. To modify a project, select it, and then click Modify. Clicking Add or Modify opens the Add/Modify Project dialog box.



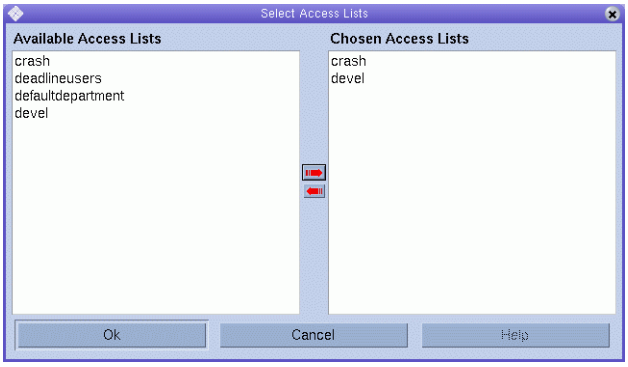
The name of the selected project is displayed in the Name field. The project defines the access lists of users who are permitted access or who are denied access to the project.

Users who are included in any of the access lists under User Lists have permission to access the project. Users who are included in any of the access lists under Xuser Lists are denied access to the project. See "Configuring Users" on page 101 for more information.

If both lists are empty, all users can access the project. Users who are included in different access lists that are attached to both the User Lists and the Xuser Lists are denied access to the project.

You can add access lists to User Lists or Xuser Lists, and you can remove access lists from either list. To do so, click the button at the right of the User Lists or the Xuser Lists.

The Select Access Lists dialog box appears.



The Select Access Lists dialog box displays all currently defined access lists under Available Access Lists. The dialog box displays the attached lists 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.

Click OK to save your changes and close the dialog box. Click Cancel to close the dialog box without saving your changes.

Defining Projects From the Command Line

To define projects from the command line, type the following command with appropriate options:

```
# qconf options
```

The following options are available:

- `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`.

- `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.

- `qconf -dprj` *project-name*[,...]

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

- `qconf -mprj` *project-name*

  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`.

- `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.

- `qconf -sprj` *project-name*

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

- `qconf -sprjl`

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

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, the directory might be made accessible across NFS. However, sometimes the home directory path is not exactly the same on all 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, 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*`/`*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 first string specifies a source path, the second a submit host, the third an execution host, and the fourth 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 as follows:

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.

Example 4–1 is an example how the NFS *automounter* problem described earlier can be resolved with an aliases file entry.

EXAMPLE 4–1 Example of Path-Aliasing File

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

Configuring Default Requests

Batch jobs are normally assigned to queues with respect to a request profile. The user defines a request profile for a particular job. The user assembles a set of requests that must be met to successfully run the job. The scheduler considers only those queues that satisfy the set of requests for this 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. However, 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/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 Chapter 3, "Submitting Jobs," in *N1 Grid Engine 6 User's Guide* 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 Example 4–2.

EXAMPLE 4–2 Example of 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.

CHAPTER 5

Managing Policies and the Scheduler

This chapter contains information about grid engine system policies. Topics in this chapter include the following:

- Scheduling
- Policies

In addition to the background information, this chapter includes detailed instructions on how to accomplish the following tasks:

Administering the Scheduler

This section describes how the grid engine system schedules jobs for execution. The section describes different types of scheduling strategies and explains how to configure the scheduler.

About Scheduling

The grid engine system includes the following job-scheduling activities:

- **Predispatching decisions.** Activities such as eliminating queues because they are full or overloaded, spooling jobs that are currently not eligible for execution, and reserving resources for higher-priority jobs
- **Dispatching.** Deciding a job's importance with respect to other pending jobs and running jobs, sensing the load on all machines in the cluster, and sending the job to a queue on a machine selected according to configured selection criteria
- **Postdispatch monitoring.** Adjusting a job's relative importance as it gets resources and as other jobs with their own relative importance enter or leave the system

The grid engine software schedules jobs across a heterogeneous cluster of computers, based on the following criteria:

- The cluster's current load
- The jobs' relative importance
- The hosts' relative performance
- The jobs' resource requirements, for example, CPU, memory, and I/O bandwidth

Decisions about scheduling are based on the strategy for the site and on the instantaneous load characteristics of each computer in the cluster. A site's scheduling strategy is expressed through the grid engine system's configuration parameters. Load characteristics are ascertained by collecting performance data as the system runs.

Scheduling Strategies

The administrator can set up strategies with respect to the following scheduling tasks:

- **Dynamic resource management.** The grid engine system dynamically controls and adjusts the resource entitlements that are allocated to running jobs. In other words, the system modifies their CPU share.
- **Queue sorting.** The software ranks the queues in the cluster according to the order in which the queues should be filled up.
- **Job sorting.** Job sorting determines the order in which the grid engine system attempts to schedule jobs.
- **Resource reservation and backfilling.** Resource reservation reserves resources for jobs, blocking their use by jobs of lower priority. Backfilling enables lower-priority jobs to use blocked resources when using those resources does not interfere with the reservation.

Dynamic Resource Management

The grid engine software uses a weighted combination of the following three ticket-based policies to implement automated job scheduling strategies:

- Share-based
- Functional (sometimes called Priority)
- Override

You can set up the grid engine system to routinely use either a share-based policy, a functional policy, or both. You can combine these policies in any combination. For example, you could give zero weight to one policy and use only the second policy. Or you could give both policies equal weight.

Along with routine policies, administrators can also override share-based and functional scheduling temporarily or, for certain purposes such as express queues, permanently. You can apply an override to one job or to all jobs associated with a user, a department, a project, or a job class (that is, a queue).

In addition to the three policies for mediating among all jobs, the grid engine system sometimes lets users set priorities among the jobs they own. For example, a user might say that jobs one and two are equally important, but that job three is more important than either job one or job two. Users can set their own job priorities if the combination of policies includes the share-based policy, the functional policy, or both. Also, functional tickets must be granted to jobs.

Tickets

The share-based, functional, and override scheduling policies are implemented with *tickets*. Each policy has a pool of tickets. A policy allocates tickets to jobs as the jobs enter the multimachine grid engine system. Each routine policy that is in force allocates some tickets to each new job. The policy might also reallocate tickets to running jobs at each scheduling interval.

Tickets weight the three policies. For example, if no tickets are allocated to the functional policy, that policy is not used. If the functional ticket pool and the share-based ticket pool have an equal number of tickets, both policies have equal weight in determining a job's importance.

Tickets are allocated to the routine policies at system configuration by grid engine system managers. Managers and operators can change ticket allocations at any time with immediate effect. Additional tickets are injected into the system temporarily to indicate an override. Policies are combined by assignment of tickets. When tickets are allocated to multiple policies, a job gets a portion of each policy's tickets, which indicates the job's importance in each policy in force.

The grid engine system grants tickets to jobs that are entering the system to indicate their importance under each policy in force. At each scheduling interval, each running job can gain tickets, lose tickets, or keep the same number of tickets. For example, a job

might gain tickets from an override. A job might lose tickets because it is getting more than its fair share of resources. The number of tickets that a job holds represent the resource share that the grid engine system tries to grant that job during each scheduling interval.

You configure a site's dynamic resource management strategy during installation. First, you allocate tickets to the share-based policy and to the functional policy. You then define the share tree and the functional shares. The share-based ticket allocation and the functional ticket allocation can change automatically at any time. The administrator manually assigns or removes tickets.

Queue Sorting

The following means are provided to determine the order in which the grid engine system attempts to fill up queues:

- **Load reporting.** Administrators can select which load parameters are used to compare the load status of hosts and their queue instances. The wide range of standard load parameters that are available, and an interface for extending this set with site-specific load sensors, are described in "Load Parameters" on page 87.
- **Load scaling.** Load reports from different hosts can be normalized to reflect a comparable situation. See "Configuring Execution Hosts With QMON" on page 24.
- **Load adjustment.** The grid engine software can be configured to automatically correct the last reported load as jobs are dispatched to hosts. The corrected load represents the expected increase in the load situation caused by recently started jobs. This artificial increase of load can be automatically reduced as the load impact of these jobs takes effect.
- **Sequence number.** Queues can be sorted following a strict sequence.

Job Sorting

Before the grid engine system starts to dispatch jobs, the jobs are brought into priority order, highest priority first. The system then attempts to find suitable resources for the jobs in priority sequence.

Without any administrator influence, the order is first-in-first-out (FIFO). The administrator has the following means to control the job order:

- **Ticket-based job priority.** 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. Any change that the administrator applies to the ticket policy also changes the sorting order.
- **Urgency-based job priority.** Jobs can have an urgency value that determines their relative importance. Pending jobs are sorted according to their urgency value. Any change applied to the urgency policy also changes the sorting order.

- **POSIX priority.** You can use the `-p` option to the `qsub` command to implement site-specific priority policies. The `-p` option specifies a range of priorities from –1023 to 1024. The higher the number, the higher the priority. The default priority for jobs is zero.
- **Maximum number of user or user group jobs.** You can restrict the maximum number of jobs that a user or a UNIX user group can run concurrently. This restriction influences the sorting order of the pending job list, because the jobs of users who have not exceeded their limit are given preference.

For each priority type, a weighting factor can be specified. 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 = weight_urgency * normalized_urgency_value +
weight_ticket * normalized_ticket_value +
weight_POSIX_priority * normalized_POSIX_priority_value
```

You can use the `qstat` command to monitor job priorities:

- Use `qstat -prio` to monitor job priorities overall, including POSIX priority.
- Use `qstat -ext` to monitor job priorities based on the ticket policy.
- Use `qstat -urg` to monitor job priorities based on the urgency policy.
- Use `qstat -pri`to diagnose job priority issues when urgency policy, ticket based policies and *-p <priority>* are used concurrently
- Use `qstat -explain`to diagnose various queue instance based error conditions.

About the Urgency Policy

The urgency policy defines an urgency value for each job. The urgency value is derived from the sum of three contributions:

- Resource requirement contribution
- Waiting time contribution
- Deadline contribution

The resource requirement contribution is derived from the sum of all hard resource requests, one addend for each request.

If the resource request is of the type *numeric*, the resource request addend is the product of the following three elements:

- The resource's urgency value as defined in the complex. For more information, see "Configuring Complex Resource Attributes With `QMON`" on page 68.
- The assumed slot allocation of the job.
- The per slot request specified by the `qsub -l` command.

If the resource request is of the type *string*, the resource request addend is the resource's urgency value as defined in the complex.

The waiting time contribution is the product of the job's waiting time, in seconds, and the waiting-weight value specified in the Policy Configuration dialog box.

The deadline contribution is zero for jobs without a deadline. For jobs with a deadline, the deadline contribution is the weight-deadline value, which is defined in the Policy Configuration dialog box, divided by the free time, in seconds, until the deadline initiation time.

For information about configuring the urgency policy, see "Configuring the Urgency Policy" on page 129.

Resource Reservation and Backfilling

Resource reservation enables you to reserve system resources for specified pending jobs. When you reserve resources for a job, those resources are blocked from being used by jobs with lower priority.

Jobs can reserve resources depending on criteria such as resource requirements, job priority, waiting time, resource sharing entitlements, and so forth. The scheduler enforces reservations in such a way that jobs with the highest priority get the earliest possible resource assignment. This avoids such well-known problems as "job starvation".

You can use resource reservation to guarantee that resources are dedicated to jobs in job-priority order.

Consider the following example. Job A is a large pending job, possibly parallel, that requires a large amount of a particular resource. A stream of smaller jobs B(i) require a smaller amount of the same resource. Without resource reservation, a resource assignment for job A cannot be guaranteed, assuming that the stream of B(i) jobs does not stop. The resource cannot be guaranteed even though job A has a higher priority than the B(i) jobs.

With resource reservation, job A gets a reservation that blocks the lower priority jobs B(i). Resources are guaranteed to be available for job A as soon as possible.

Backfilling enables a lower-priority job to use resources that are blocked due to a resource reservation. Backfilling work only if there is a runnable job whose prospective run time is small enough to allow the blocked resource to be used without interfering with the original reservation.

In the example described earlier, a job C, of very short duration, could use backfilling to start before job A.

Because resource reservation causes the scheduler to look ahead, using resource reservation affects system performance. In a small cluster, the effect on performance is negligible when there are only a few pending jobs. In larger clusters, however, and in clusters with many pending jobs, the effect on performance might be significant.

To offset this potential performance degradation, you can limit the overall number of resource reservations that can be made during a scheduling interval. You can limit resource reservation in two ways:

- To limit the absolute number of reservations that can be made during a scheduling interval, set the Maximum Reservation parameter on the Scheduler Configuration dialog box. For example, if you set Maximum Reservation to 20, no more than 20 reservations can be made within an interval.
- To limit reservation scheduling to only those jobs that are important, use the `-R y` option of the `qsub` command. In the example described earlier, there is no need to schedule B(i) job reservations just for the sake of guaranteeing the resource reservation for job A. Job A is the only job that you need to submit with the `-R y` option.

You can configure the scheduler to monitor how it is influenced by resource reservation. When you monitor the scheduler, information about each scheduling run is recorded in the file *sge-root*/*cell*/`common/schedule`.

The following example shows what schedule monitoring does. Assume that the following sequence of jobs is submitted to a cluster where the global `license` consumable resource is limited to 5 licenses:

```
qsub -N L4_RR -R y -l h_rt=30,license=4 -p 100 $SGE_ROOT/examples/jobs/sleeper.sh 20
qsub -N L5_RR -R y -l h_rt-30,license=5        $SGE_ROOT/examples/jobs/sleeper.sh 20
qsub -N L1_RR -R y -l h_rt=31,license=1        $SGE_ROOT/examples/jobs/sleeper.sh 20
```

Assume that the default priority settings in the scheduler configuration are being used:

```
weight_priority          1.000000
weight_urgency           0.100000
weight_ticket            0.010000
```

The `-p 100` priority of job `L4_RR` supersedes the license-based urgency, which results in the following prioritization:

```
job-ID  prior    name
--------------------
   3127 1.08000 L4_RR
   3128 0.10500 L5_RR
   3129 0.00500 L1_RR
```

In this case, traces of these jobs can be found in the `schedule` file for 6 schedule intervals:

```
::::::::
      3127:1:STARTING:1077903416:30:G:global:license:4.000000
      3127:1:STARTING:1077903416:30:Q:all.q@carc:slots:1.000000
```

```
3128:1:RESERVING:1077903446:30:G:global:license:5.000000
3128:1:RESERVING:1077903446:30:Q:all.q@bilbur:slots:1.000000
3129:1:RESERVING:1077903476:31:G:global:license:1.000000
3129:1:RESERVING:1077903476:31:Q:all.q@es-ergb01-01:slots:1.000000
::::::::
3127:1:RUNNING:1077903416:30:G:global:license:4.000000
3127:1:RUNNING:1077903416:30:Q:all.q@carc:slots:1.000000
3128:1:RESERVING:1077903446:30:G:global:license:5.000000
3128:1:RESERVING:1077903446:30:Q:all.q@es-ergb01-01:slots:1.000000
3129:1:RESERVING:1077903476:31:G:global:license:1.000000
3129:1:RESERVING:1077903476:31:Q:all.q@es-ergb01-01:slots:1.000000
::::::::
3128:1:STARTING:1077903448:30:G:global:license:5.000000
3128:1:STARTING:1077903448:30:Q:all.q@carc:slots:1.000000
3129:1:RESERVING:1077903478:31:G:global:license:1.000000
3129:1:RESERVING:1077903478:31:Q:all.q@bilbur:slots:1.000000
::::::::
3128:1:RUNNING:1077903448:30:G:global:license:5.000000
3128:1:RUNNING:1077903448:30:Q:all.q@carc:slots:1.000000
3129:1:RESERVING:1077903478:31:G:global:license:1.000000
3129:1:RESERVING:1077903478:31:Q:all.q@es-ergb01-01:slots:1.000000
::::::::
3129:1:STARTING:1077903480:31:G:global:license:1.000000
3129:1:STARTING:1077903480:31:Q:all.q@carc:slots:1.000000
::::::::
3129:1:RUNNING:1077903480:31:G:global:license:1.000000
3129:1:RUNNING:1077903480:31:Q:all.q@carc:slots:1.000000
```

Each section shows, for a schedule interval, all resource usage that was taken into account. `RUNNING` entries show usage of jobs that were already running at the start of the interval. `STARTING` entries show the immediate uses that were decided within the interval. `RESERVING` entries show uses that are planned for the future, that is, reservations.

The format of the schedule file is as follows:

jobID	The job ID
taskID	The array task ID, or 1 in the case of nonarray jobs
state	Can be `RUNNING`, `SUSPENDED`, `MIGRATING`, `STARTING`, `RESERVING`
start-time	Start time in seconds after 1.1.1070
duration	Assumed job duration in seconds
level-char	Can be P (for parallel environment), G (for global), H (for host), or Q (for queue)
object-name	The name of the parallel environment, host, or queue
resource-name	The name of the consumable resource
usage	The resource usage incurred by the job

The line `::::::::` marks the beginning of a new schedule interval.

Note – The `schedule` file is not truncated. Be sure to turn monitoring off if you do not have an automated procedure that is set up to truncate the file.

What Happens in a Scheduler Interval

The Scheduler schedules work in intervals. Between scheduling actions, the grid engine system keeps information about significant events such as the following:

- Job submission
- Job completion
- Job cancellation
- An update of the cluster configuration
- Registration of a new machine in the cluster

When scheduling occurs, the scheduler first does the following:

- Takes into account all significant events
- Sorts jobs and queues according to the administrator's specifications
- Takes into account all the jobs' resource requirements
- Reserves resources for jobs in a forward-looking schedule

Then the grid engine system does the following tasks, as needed:

- Dispatches new jobs
- Suspends running jobs
- Increases or decreases the resources allocated to running jobs
- Maintains the status quo

If share-based scheduling is used, the calculation takes into account the usage that has already occurred for that user or project.

If scheduling is not at least in part share-based, the calculation ranks all the jobs running and waiting to run. The calculation then takes the most important job until the resources in the cluster (CPU, memory, and I/O bandwidth) are used as fully as possible.

Scheduler Monitoring

If the reasons why a job does not get started are unclear to you, run the `qalter -w v` command for the job. The grid engine software assumes an empty cluster and checks whether any queue that is suitable for the job is available.

Further information can be obtained by running the `qstat -j` *job-id* command. This command prints a summary of the job's request profile. The summary also includes the reasons why the job was not scheduled in the last scheduling run. Running the `qstat -j` command without a job ID summarizes the reasons for all jobs not being scheduled in the last scheduling interval.

Note – Collection of job scheduling information must be switched on in the scheduler configuration `sched_conf`(5). Refer to the `schedd_job_info` parameter description in the `sched_conf`(5) man page, or to "Changing the Scheduler Configuration With QMON" on page 123.

To retrieve even more detail about the decisions of the scheduler `sge_schedd`, use the `-tsm` option of the `qconf` command. This command forces `sge_schedd` to write trace output to the file.

Configuring the Scheduler

Refer to "Configuring Policy-Based Resource Management With QMON" on page 127 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` 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 in order 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" on page 87 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" on page 24, 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" on page 87) and "Consumable Resources" on page 74.

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 oversubscription 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 "Changing the Scheduler Configuration With `QMON`" on page 123.

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" on page 24 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. In order to enforce this feature, do one of the following:

- Set `maxujobs` or `maxgjobs`, or both, 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.

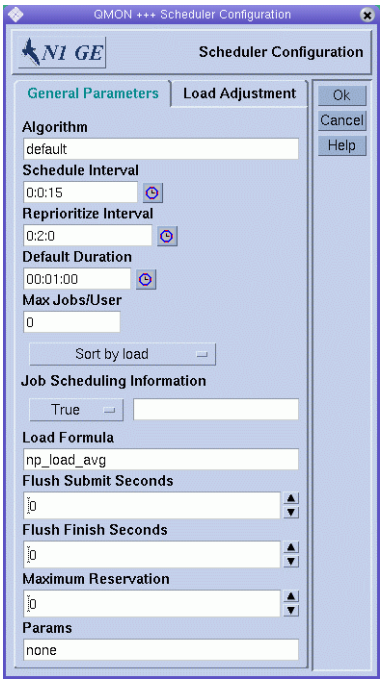
Changing the Scheduler Configuration With QMON

On the QMON Main Control window, click the Scheduler Configuration button.

The Scheduler Configuration dialog box appears. The dialog box has two tabs:

- General Parameters tab
- Load Adjustment tab

To change general scheduling parameters, click the General Parameters tab. The General Parameters tab looks like the following figure.



Use the General Parameters tab to set the following parameters:

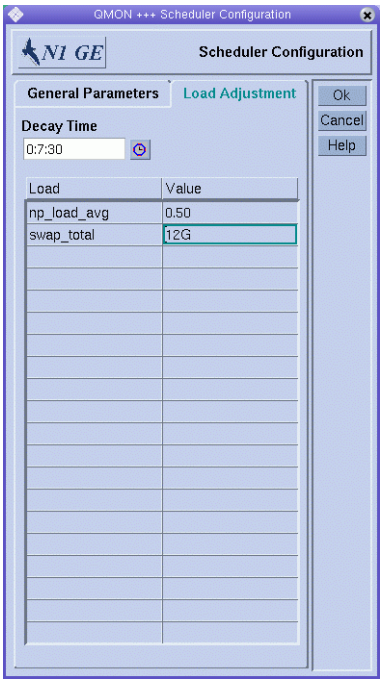
- **Algorithm.** The scheduling algorithm. See "Changing the Scheduling Algorithm" on page 121.
- **Schedule Interval.** The regular time interval between scheduler runs.
- **Reprioritize Interval.** The regular time interval to reprioritize jobs on the execution hosts, based on the current ticket amount 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. See "Restricting the Number of Jobs per User or Group" on page 123.
- **Sort by.** The queue sorting scheme, either sorting by load or sorting by sequence number. See "Selecting Queue by Sequence Number" on page 122.
- **Job Scheduling Information.** Whether job scheduling information is accessible through `qstat -j`, or whether this information should be collected only for a range of job IDs. You should turn on general collection of job scheduling information only temporarily, in case an extremely high number of jobs are pending.

  Scheduler monitoring can help you find out the reason why certain jobs are not dispatched. However, providing this information for all jobs at all times can consume resources. Such information is usually not needed.

- **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. See "Resource Reservation and Backfilling" on page 116.
- **Params.** Use this setting to specify additional parameters to pass to the scheduler. Params can be `PROFILE` or `MONITOR`. If you specify `PROFILE`, the scheduler logs profiling information that summarizes each scheduling run. If you specify `MONITOR`, the scheduler records information for each scheduling run in the file *sge-root*`/`*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. For more information, see "Immediate Scheduling" on page 189.

To change load adjustment parameters, click the Load Adjustment tab. The Load Adjustment tab looks like the following figure:



Use the Load Adjustment tab to set the 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.

  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.

  The Attribute Selection dialog box is shown in Figure 1–2. To add a resource attribute to the Load column of the Consumable/Fixed Attributes table, 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 "Scaling System Load" on page 121 for background information. See the `sched_conf`(5) man page for more details about the scheduler configuration.

Administering Policies

This section describes how to configure policies to manage cluster resources.

The grid engine software orchestrates the delivery of computational power, based on enterprise resource policies that the administrator manages. The system uses these policies to examine available computer resources in the grid. The system 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

As administrator, you can define high-level usage policies that are customized for your site. Four such policies are available:

- Urgency policy – See "Configuring the Urgency Policy" on page 129
- Share-based policy – See "Configuring the Share-Based Policy" on page 135
- Functional policy – See "Configuring the Functional Policy" on page 147
- Override policy – See "Configuring the Override Policy" on page 151

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.

Configuring Policy-Based Resource Management With `QMON`

On the `QMON` Main Control window, click the Policy Configuration button. The Policy Configuration dialog box appears.

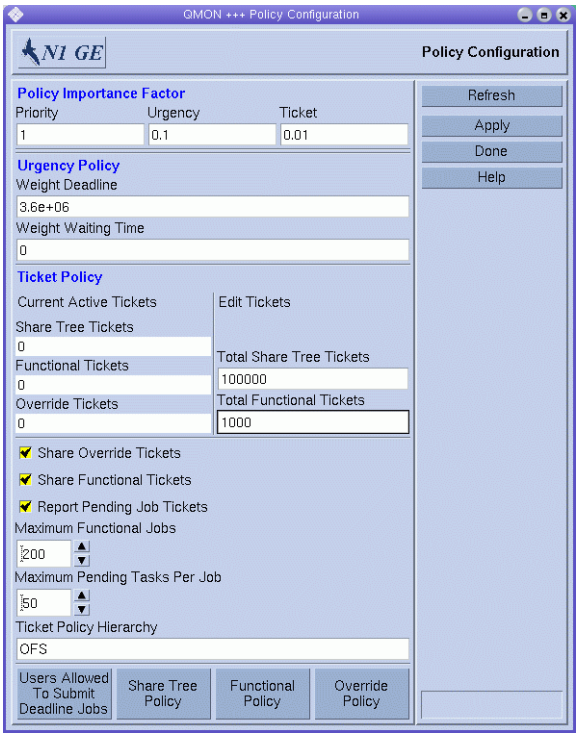


FIGURE 5–1 Policy Configuration Dialog Box

The Policy Configuration dialog box shows the following information:

- Policy Importance Factor
- Urgency Policy
- Ticket Policy. You can readjust the policy-related tickets.

From this dialog box you can access specific configuration dialog boxes for the three ticket-based policies.

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:

- **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" on page 114.

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
```

Urgency, Ticket, and Priority are the three weighting factors you specify under Policy Importance Factor. 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.

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" on page 68. 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.

- **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" on page 115.

Configuring Ticket-Based Policies

The tickets that are currently assigned to individual policies are listed under Current Active Tickets. 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" on page 131.

Select the Share Override Tickets check box to control the total ticket amount distributed by the override policy. Clear the check box to control the importance of individual jobs relative to the ticket pools that are available for the other policies and override categories. For detailed information, see "Sharing Override Tickets" on page 131.

Select the Share Functional Tickets check box to give a category member a constant entitlement level for the sum of all its jobs. Clear 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" on page 132.

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" on page 134.

To refresh the information displayed, click Refresh.

To save any changes that you make to the Policy Configuration, click Apply. To close the dialog box without saving changes, click Done.

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 clear 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.

Clear 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, a department, and a job class.

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. Or 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.
- Clearing 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.

Clear 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 5–1 Functional Policy Example

The following example describes a common scenario where a user wishes to translate the SGE-5.3 Scheduler Option `-user_sort true` to an N1GE 6.1 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" on page 114. 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:

- 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.

```
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.

`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 in order 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 underusage or overusage 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 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

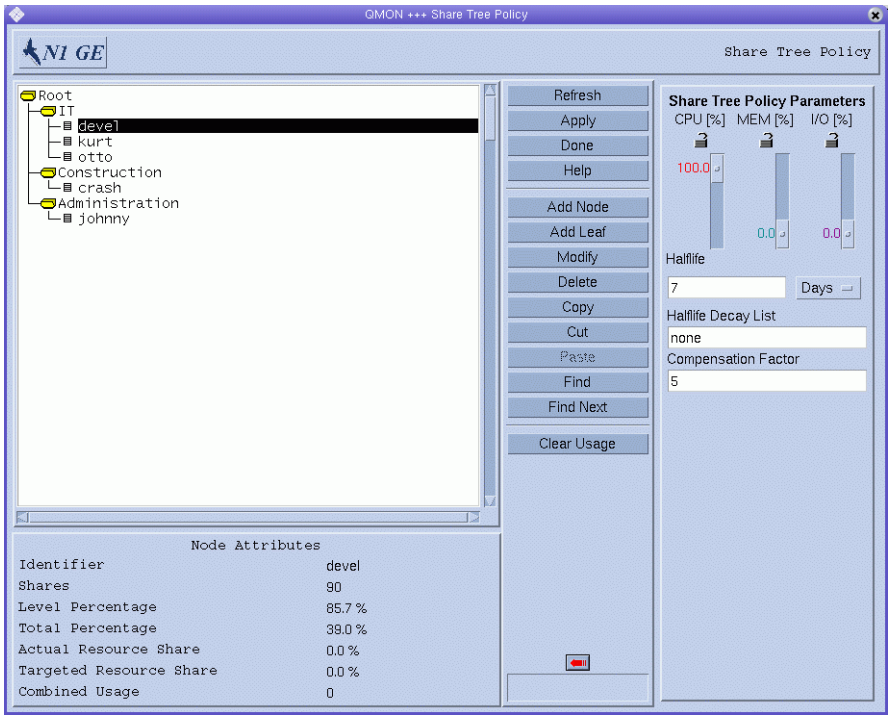
The share-based policy is implemented through a *hierarchical share tree*. The share tree specifies, for a moving accumulation period, how system resources are to be shared among all users and projects. The length of the accumulation period is determined by a configurable decay constant. The grid engine 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 (Figure 5–1), click Share Tree Policy. The Share Tree Policy dialog box appears.



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`" on page 101) or project ("Defining Projects" on page 103)

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 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`" on page 142.
- Project subtrees do not have subprojects.

- 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 descendents, 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 N1 Grid Engine 6 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 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 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 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 add 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 are in the 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" on page 137 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 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.

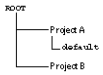
You can also use the `default` user for "hybrid" share trees, where users are subordinated under projects in the share tree. The `default` user can be a leaf node under a project node.

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.

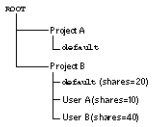
In Example A, all users submitting to Project A get equal long-term entitlements. The users submitting to Project B only contribute to the accumulated resource consumption of Project B. Entitlements of Project B users are not managed.

EXAMPLE 5–2 Example A



Compare Example A with Example B:

EXAMPLE 5–3 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.

Configuring 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 appropriate options.

- The `qconf` options `-astree`, `-mstree`, `-dstree`, and `-sstree`, enable you to do the following:
  - Add a new share tree
  - Modify an existing share tree
  - Delete a share tree
  - Display the share tree configuration

  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 Create Project-Based Share-Tree Scheduling

The objective of this setup is to guarantee a certain share assignment of all the cluster resources to different projects over time.

Steps

1. **Specify the number of share-tree tickets (for example, 1000000) in the scheduler configuration.**

See "Configuring Policy-Based Resource Management With QMON" on page 127, and the `sched_conf`(5) man page.

2. **(Optional) Add one user for each scheduling-relevant user.**

   See "Configuring User Objects With QMON" on page 101, and the `user`(5) man page.

3. **Add one project for each scheduling-relevant project.**

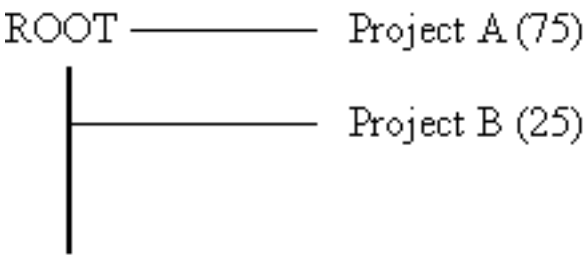
   See "Defining Projects With QMON" on page 104, and the `project`(5) man page.

4. **Use `QMON` to set up a share tree that reflects the structure of all scheduling-relevant projects as nodes.**

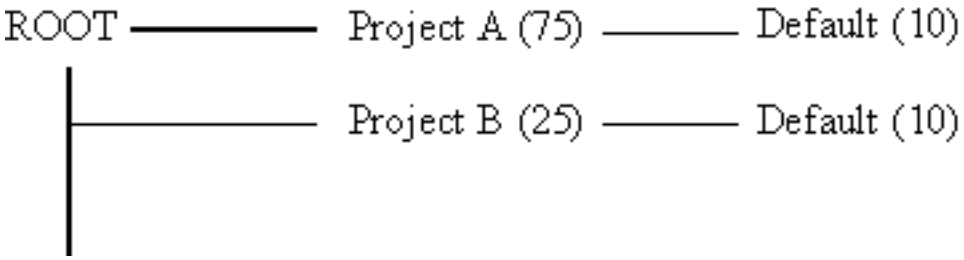
   See "Configuring the Share-Tree Policy With QMON" on page 138.

5. **Assign share tree shares to the projects.**

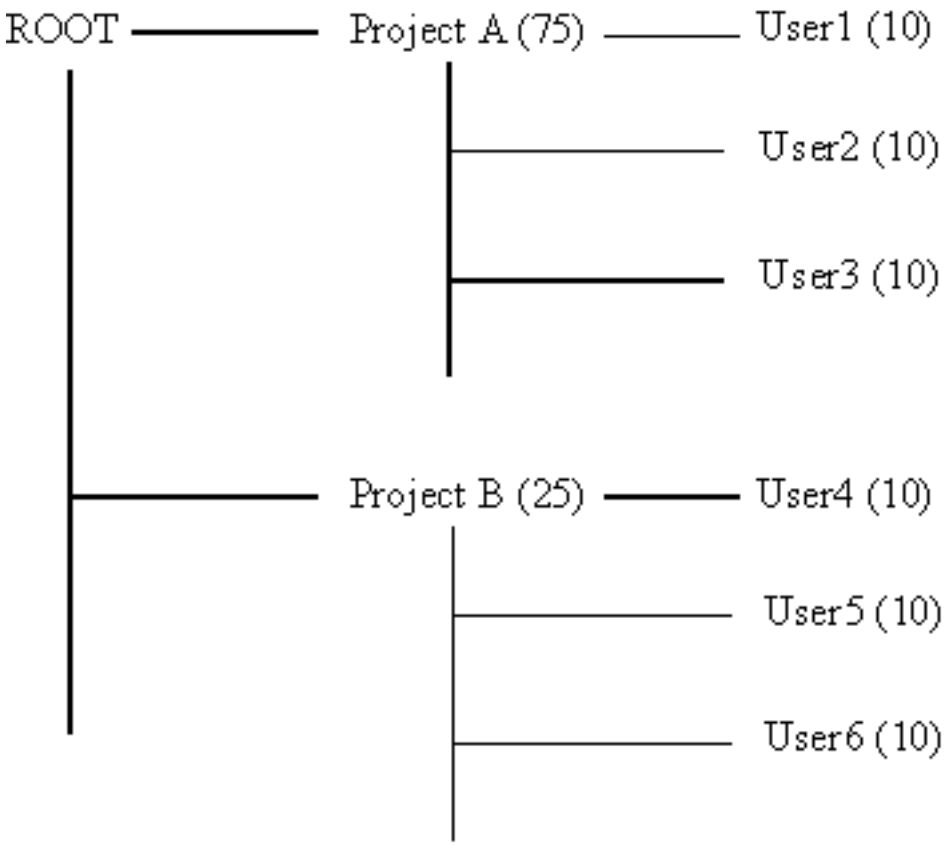
   For example, if you are creating project-based share-tree scheduling with first-come, first-served scheduling among jobs of the same project, a simple structure might look like the following:



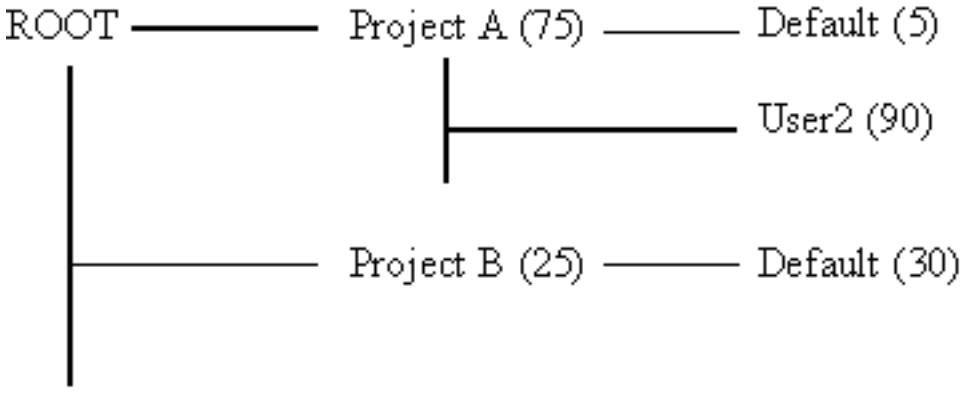
If you are creating project-based share-tree scheduling with equal shares for each user, a simple structure might look like the following:



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, department, and job class. Functional scheduling is sometimes called priority scheduling. The functional policy setup ensures that a defined share is guaranteed to each user, project, 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, and departments 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 share-based policy entitlement.

The total number of tickets that are allotted to the functional policy determines the weight of functional scheduling among the three scheduling policies. During installation, the administrator divides the total number of functional tickets among the functional categories of user, department, project, job, and job class.

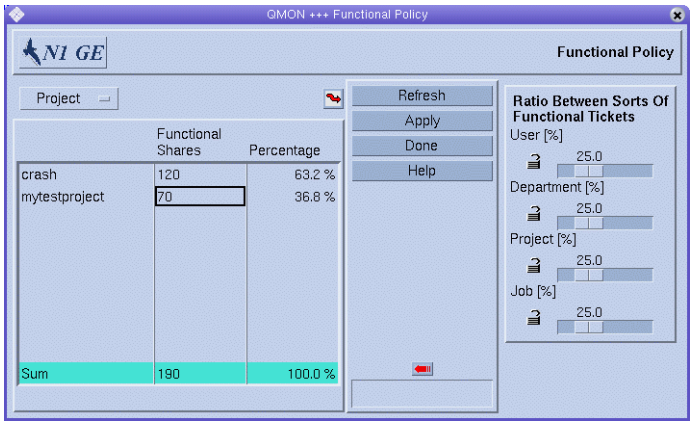
Functional Shares

Functional shares are assigned to every member of each functional category: user, department, project, job, and job class. These shares indicate what proportion of the tickets for a category each job associated with a member of the category is entitled to. 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 `donlee`'s job, no matter how many tickets there are.

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 "Configuring User Objects With QMON" on page 101.
- 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. See "Defining Usersets As Projects and Departments" on page 101.
- For Project functional shares, the Project Configuration dialog box appears. See "Defining Projects With QMON" on page 104.
- For Job functional shares, the Job Control dialog box appears. See "Monitoring and Controlling Jobs With QMON" in *N1 Grid Engine 6 User's Guide*.

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 [%], Job [%] and Job Class [%] 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
- 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

Configuring the Functional Share Policy From the Command Line

Note – You can assign functional shares to jobs *only* using `QMON`. No command-line interface is available for this function.

To configure the functional share policy from the command line, use the `qconf` command with the appropriate options.

- Use the `qconf -muser` command to configure the user category. 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.
- Use the `qconf -mu` command to configure the department category. 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.
- Use the `qconf -mprj` command to configure the project category. 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.
- Use the `qconf -mq` command to configure the job class category. The `-mq` option modifies the `fshare` parameter of the queue configuration file. See the `queue_conf`(5) man page for information about the queue configuration file, which is used to represent job classes.
- 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`, `weight_job`, and `weight_jobclass`. The parameter values range between 0 and 1, and the total sum of parameters must add up to 1.

▼ How to Create User-Based, Project-Based, and Department-Based Functional Scheduling

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.

Steps

1. **In the Scheduler Configuration dialog box, select the Share Functional Tickets check box.**

   See "Sharing Functional Ticket Shares" on page 132, and the `sched_conf`(5) man page.

2. **Specify the number of functional tickets (for example, 1000000) in the scheduler configuration.**

   See "Configuring Policy-Based Resource Management With QMON" on page 127, and the `sched_conf`(5) man page.

3. **Add scheduling-relevant items:**

   - **Add one user for each scheduling-relevant user.**

     See "Configuring User Objects With QMON" on page 101, and the `user`(5) man page.

   - **Add one project for each scheduling-relevant project.**

     See "Defining Projects With QMON" on page 104, and the `project`(5) man page.

   - **Add each scheduling-relevant department.**

4. **Assign functional shares to each user, project, or department.**

   See "Configuring User Access Lists With QMON" on page 98, 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)

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, a project, or a job class. This adjustment adds tickets to the specified job, user, department, project, or job class. 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 job classes, or high, medium, or low priority classes.

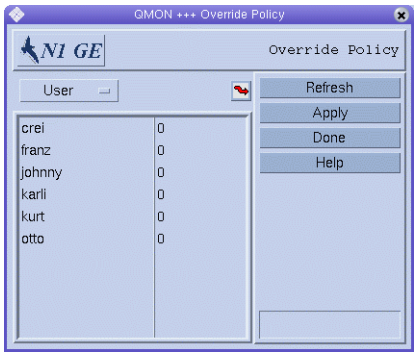
Override tickets that are assigned directly to a job go away when the job finishes. All other tickets are inflated back to their original value. Override tickets that are assigned to users, departments, projects, and job classes remain in effect until the administrator explicitly removes the tickets.

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 `QMON` 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, job, and job class.
- 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 "Configuring User Objects With `QMON`" on page 101.
- For Department override tickets, 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. See "Defining Usersets As Projects and Departments" on page 101.
- For Project override tickets, the Project Configuration dialog box appears. See "Defining Projects With `QMON`" on page 104.
- For Job override tickets, the Job Control dialog box appears. See "Monitoring and Controlling Jobs With QMON" in *N1 Grid Engine 6 User's Guide*.

Configuring the Override Policy From the Command Line

Note – You can assign override tickets to jobs *only* using `QMON`. No command line interface is available for this function.

To configure the override policy from the command line, use the `qconf` command with the appropriate options.

- Use the `qconf -muser` command to configure the user category. 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.
- Use the `qconf -mu` command to configure the department category. 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.
- Use the `qconf -mprj` command to configure the project category. 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.
- Use the `qconf -mq` command to configure the job class category. The `-mq` option modifies the `oticket` parameter of the queue configuration file. See the `queue_conf`(5) man page for information about the queue configuration file, which is used to represent job classes.

CHAPTER 6

Managing Special Environments

This chapter describes how to manage and administer the following special environments:

- Parallel environments
- Checkpointing environments

In addition to background information about these environments, this chapter includes detailed instructions for accomplishing the following tasks:

Configuring 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:

- PVM – Parallel Virtual Machine, Oak Ridge National Laboratories
- MPI – Message Passing Interface, 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 segregative requirements. In order 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 provides means to run parallel jobs by means of the following programs:

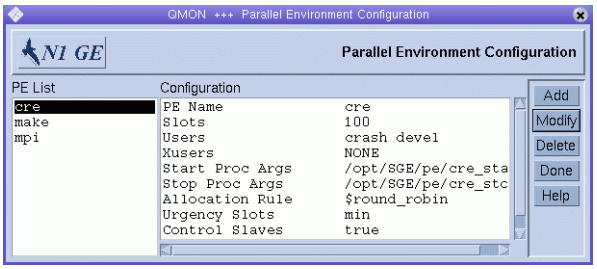
- Arbitrary message-passing environments such as PVM or MPI. See the *PVM User's Guide* and the *MPI User's Guide* for details.
- Shared memory parallel programs on multiple slots, either in single queues or distributed across multiple queues and across machines for distributed memory parallel jobs.

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" on page 162 and in "Termination of the Parallel Environment" on page 163, respectively.

Configuring Parallel Environments With QMON

On the QMON Main Control window, click the Parallel Environment Configuration button. The Parallel Environment Configuration dialog box appears.



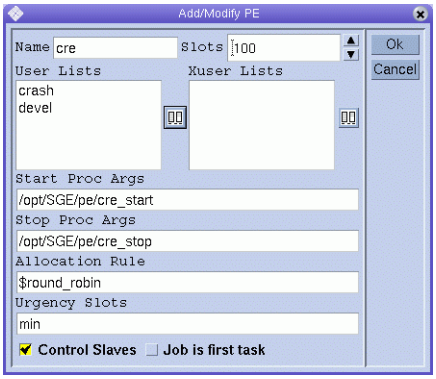
Currently configured parallel environments are displayed under PE List.

To display the contents of a parallel environment, select it. The selected parallel environment configuration is displayed under Configuration.

To delete a parallel environment, select it, and then click Delete.

To add a new parallel environment, click Add. To modify a parallel environment, select it, and then click Modify.

When you click Add or Modify, the Add/Modify PE dialog box appears.

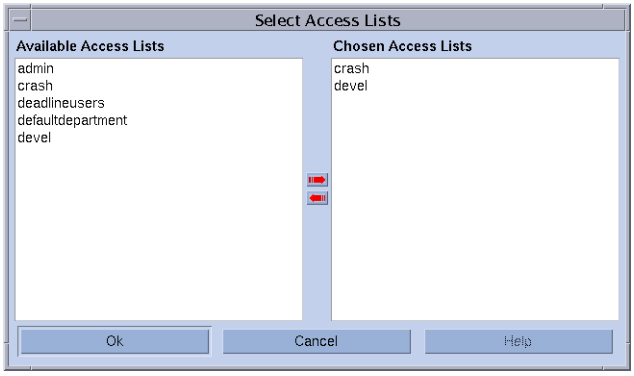


If you are adding a new parallel environment, type its name in the Name field. If you are modifying a parallel environment, its name is displayed in the Name field.

In the Slots box, enter the total number of job slots that can be occupied by all parallel environment jobs running concurrently.

User Lists displays the user access lists that are allowed to access the parallel environment. Xuser Lists displays the user access lists that are not allowed to access the parallel environment. See "Configuring User Access Lists" on page 98 for more information about user access lists.

Click the icons at the right of each list to modify the content of the lists. The Select Access Lists dialog box appears.



The Start Proc Args and Stop Proc Args fields are optional. Use these fields to enter the precise invocation sequence of the parallel environment startup and stop procedures. See the sections "Parallel Environment Startup Procedure" on page 162 and "Termination of the Parallel Environment" on page 163, respectively. If no such procedures are required for a certain parallel environment, you can leave the fields empty.

The first argument is usually the name of the start or stop procedure itself. The remaining parameters are command-line arguments to the procedures.

A variety of special identifiers, which begin with a `$` prefix, are available to pass internal runtime information to the procedures. The `sge_pe`(5) man page contains a list of all available parameters.

The Allocation Rule field defines the number of parallel processes to allocate on each machine that is used by a parallel environment. A positive integer fixes the number of processes for each suitable host. Use the special denominator `$pe_slots` to cause the full range of processes of a job to be allocated on a single host (SMP). Use the denominators `$fill_up` and `$round_robin` to cause unbalanced distributions of processes at each host. For more details about these allocation rules, see the `sge_pe`(5) man page.

The Urgency Slots field specifies the method the grid engine system uses to assess the number of slots that pending jobs with a slot range get. The assumed slot allocation is meaningful when determining the resource-request-based priority contribution for numeric resources. You can specify an integer value for the number of slots. Specify `min` to use the slot range minimum. Specify `max` to use the slot range maximum. Specify `avg` to use the average of all numbers occurring within the job's parallel environment range request.

The Control Slaves check box specifies whether the grid engine system generates parallel tasks or whether the corresponding parallel environment creates its own process. The grid engine system uses `sge_execd` and `sge_shepherd` to generate parallel tasks. Full control over slave tasks by the grid engine system is preferable, because the system provides the correct accounting and resource control. However, this functionality is available only for parallel environment interfaces especially customized for the grid engine system. See "Tight Integration of Parallel Environments and Grid Engine Software" on page 164 for more details.

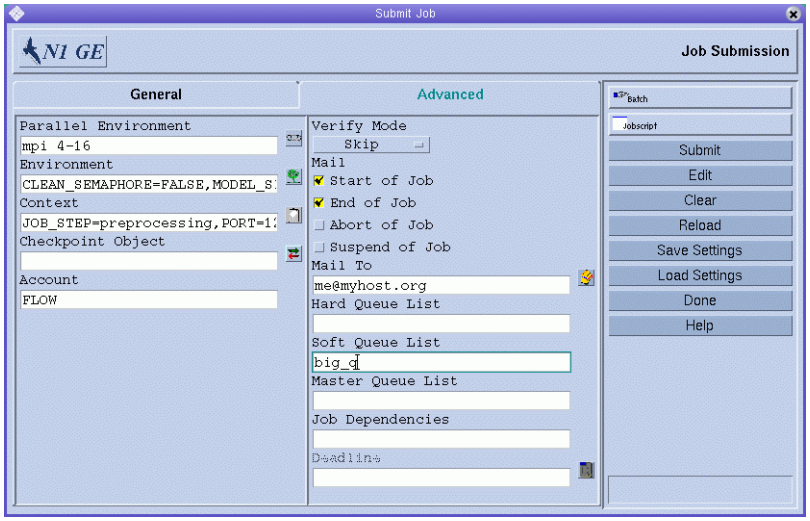
The Job Is First Task check box is meaningful only if Control Slaves is selected. If you select Job Is First Task, the job script or one of its child processes acts as one of the parallel tasks of the parallel application. For PVM, you usually want the job script to be part of the parallel application, for example. If you clear the Job Is First Task check box, the job script initiates the parallel application but does not participate. For MPI, you usually do not want the job script to be part of the parallel application, for example, when you use `mpirun`.

Click OK to save your changes and close the dialog box. Click Cancel to close the dialog box without saving changes.

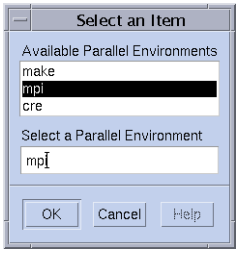
Displaying Configured Parallel Environment Interfaces With QMON

On the QMON Main Control window, click the Parallel Environment Configuration button. The Parallel Environment Configuration dialog box appears. See "Configuring Parallel Environments With QMON" on page 156 for more information.

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. 16 is preferable.



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.



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.

The `qsub` command corresponding to the parallel job specification described previously 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 the grid engine system administration 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 "Defining Resource Requirements" in *N1 Grid Engine 6 User's Guide*.

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.

Configuring Parallel Environments From the Command Line

Type the `qconf` command with appropriate options:

`qconf` *options*

The following options are available:

- `qconf -ap` *pe-name*

  The `-ap` option (add parallel environment) displays an editor containing a parallel environment configuration template. The editor is either the default `vi` editor or an 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. Configure the parallel environment by changing the template and saving to disk. See the `sge_pe`(5) man page for a detailed description of the template entries to change.

- `qconf -Ap` *filename*

  The `-Ap` option (add parallel environment from file) parses the specified file *filename* and adds the new parallel environment configuration.

  The file must have the format of the parallel environment configuration template.

- `qconf -dp` *pe-name*

  The `-dp` option (delete parallel environment) deletes the specified parallel environment.

- `qconf -mp` *pe-name*

  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 an editor defined by the `EDITOR` environment variable. Modify the parallel environment by changing the template and saving to disk. See the `sge_pe`(5) man page for a detailed description of the template entries to change.

- `qconf -Mp` *filename*

  The `-Mp` option (modify parallel environment from file) parses the specified file *filename* and modifies the existing parallel environment configuration.

  The file must have the format of the parallel environment configuration template.

- `qconf -sp` *pe-name*

  The `-sp` option (show parallel environment) prints the configuration of the specified parallel environment to standard output.

- `qconf -spl`

  The `-spl` option (show parallel environment list) lists the names of all currently configured parallel environments.

Parallel Environment Startup Procedure

The grid engine system 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 system.

An example for such a startup procedure for the PVM environment is contained in the distribution tree of the grid engine system. 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 "Configuring Parallel Environments With `QMON`" on page 156. The parameters are among the parameters provided to parallel environment startup and stop scripts by the grid engine system during runtime. The required host file, as an example, is generated by the grid engine system. 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 system. 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 system 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 system framework.

Termination of the Parallel Environment

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, "Configuring Parallel Environments With `QMON`" on page 156.

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 system 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 system. Doing so avoids all errors that can be hard to trace if the procedure is integrated into the grid engine system framework.

Tight Integration of Parallel Environments and Grid Engine Software

"Configuring Parallel Environments With QMON" on page 156 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 contain 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 system parallel environment interface. You might want to contact your Sun support representative distributor for assistance.

Configuring Checkpointing Environments

Checkpointing is a facility that does the following tasks:

1. Freezes the status of an running job or application
2. Saves this status (the checkpoint) to disk
3. 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 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.

About Checkpointing Environments

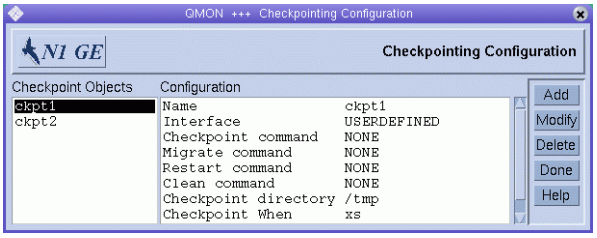
The grid engine system provides a configurable attribute description for each checkpointing method used. Different attribute descriptions reflect the different checkpointing methods and the potential variety of derivatives from these methods on different operating system architectures.

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.

Configuring Checkpointing Environments With QMON

On the QMON Main Control window, click the Checkpoint Configuration button. The Checkpointing Configuration dialog box appears.

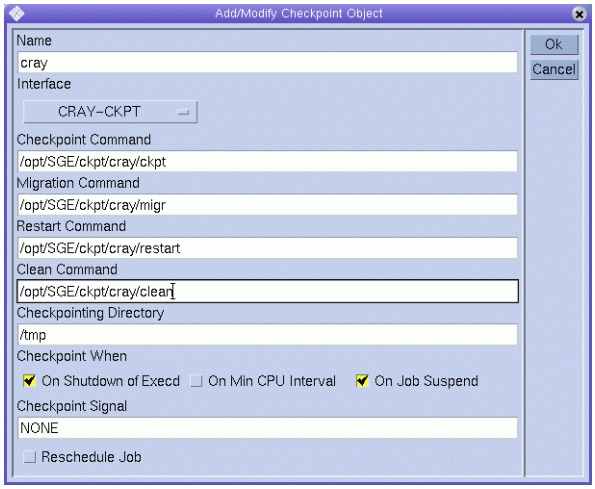


Viewing Configured Checkpointing Environments

To view previously configured checkpointing environments, select one of the checkpointing environment names listed under Checkpoint Objects. The corresponding configuration is displayed under Configuration.

Adding a Checkpointing Environment

In the Checkpointing Configuration dialog box, click Add. The Add/Modify Checkpoint Object dialog box appears, along with a template configuration that you can edit.



Fill out the template with the requested information.

Click OK to register your changes with `sge_qmaster`. Click Cancel to close the dialog box without saving changes.

Modifying Checkpointing Environments

In the Checkpoint Objects list, 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
- 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.

In addition, you must define the Interface to use. The Interface is also called *checkpointing method*. From the Interface list under Name, select an Interface. See the `checkpoint`(5) man page for details about the meaning of the different interfaces.

Note – For the checkpointing environments provided with the distribution of the grid engine system, change only the Name parameter and the Checkpointing Directory parameter.

Click OK to register your changes with `sge_qmaster`. Click Cancel to close the dialog box without saving changes.

Deleting Checkpointing Environments

To delete a configured checkpointing environment, select it, and then click Delete.

Configuring Checkpointing Environments From the Command Line

To configure the checkpointing environment from the command line, type the `qconf` command with the appropriate options.

The following options are available:

- `qconf -ackpt` *ckpt-name*

  The `-ackpt` option (add checkpointing environment) displays an editor containing a checkpointing environment configuration template. The editor is either the default `vi` editor or an editor corresponding to the `EDITOR` environment variable. The parameter *ckpt-name* specifies the name of the checkpointing environment. The parameter is already provided in the corresponding field of the template. Configure the checkpointing environment by changing the template and saving to disk. See the `checkpoint`(5) man page for a detailed description of the template entries to be changed.

- `qconf -Ackpt` *filename*

  The `-Ackpt` option (add checkpointing environment from file) parses the specified file and adds the new checkpointing environment configuration.

  The file must have the format of the checkpointing environment template.

- `qconf -dckpt` *ckpt-name*

  The `-dckpt` option (delete checkpointing environment) deletes the specified checkpointing environment.

- `qconf -mckpt` *ckpt-name*

The `-mckpt` option (modify checkpointing environment) displays an editor containing the specified checkpointing environment as a configuration template. The editor is either the default `vi` editor or an editor corresponding to the `EDITOR` environment variable. Modify the checkpointing environment by changing the template and saving to disk. See the `checkpoint`(5) man page for a detailed description of the template entries to be changed.

- `qconf -Mckpt` *filename*

  The `-Mckpt` option (modify checkpointing environment from file) parses the specified file and modifies the existing checkpointing configuration.

  The file must have the format of the checkpointing environment template.

- `qconf -sckpt` *ckpt-name*

  The `-sckpt` option (show checkpointing environment) prints the configuration of the specified checkpointing environment to standard output.

- `qconf -sckptl`

  The `-sckptl` option (show checkpointing environment list) displays a list of the names of all checkpointing environments currently configured.

CHAPTER 7

Other Administrative Tasks

This chapter describes how to use files and scripts to add or modify grid engine system objects such as queues, hosts, and environments.

This chapter includes the following sections:

Gathering Accounting and Reporting Statistics

The grid engine system provides two kinds of reporting and accounting facilities:

- Accounting and Reporting Console (ARCo)
- `qacct` command and the `accounting` file

Report Statistics (ARCo)

You can use the optional Accounting and Reporting Console (ARCo) to generate live accounting and reporting data from the grid engine system and store the data in the *reporting database*, which is a standard SQL database. ARCo supports the following SQL database systems:

- PostgreSQL
- Oracle
- MySQL

ARCo also provides a web-based tool for generating information queries on the reporting database and for retrieving the results in tabular or graphical form. ARCo enables you to store queries for later use, to run predefined queries, and to run queries in batch mode. For more information about how to use ARCo, see Chapter 5, "Accounting and Reporting," in *N1 Grid Engine 6 User's Guide*. For information about how to install ARCo, see Chapter 8, "Installing the Accounting and Reporting Console," in *N1 Grid Engine 6 Installation Guide*.

Raw reporting data is generated by `sge_qmaster`. This raw data is stored in a *reporting file*. The `dbwriter` program reads the raw data in the reporting file and writes it to the SQL reporting database, where it can be accessed by ARCo.

About the `dbwriter` Program

The `dbwriter` program performs the following tasks:

- Reads raw data from the reporting file and writes this raw data to the reporting database.
- Calculates derived values. You can configure which values to calculate, as well as the rules that govern the calculations.
- Deletes outdated data. You can configure how long to keep data.

When `dbwriter` starts up, it calculates derived values. `dbwriter` also deletes outdated records at startup. If `dbwriter` runs in continuous mode, `dbwriter` continues to calculate derived values and to delete outdated records at hourly intervals, or at whatever interval you specify.

You can specify in an XML file the values that you want to calculate and the records that you want to delete. Use the `-calculation` option of the `dbwriter` command to specify the path to this XML file.

For detailed information about calculating derived values, see "Calculating Derived Values With `dbwriter`" on page 173.

For detailed information about deleting outdated records, see "Deleting Outdated Records With `dbwriter`" on page 176.

Enabling the Reporting File

The reporting file contains the following types of data:

- Host load values and consumable resources
- Queue consumable resources
- Job logging
- Job accounting
- Share-tree usage

When the grid engine system is first installed, the reporting file is disabled. To use ARCo, you must enable the reporting file for the cluster. Once enabled, the reporting file will be generated by `sge_qmaster`. By default, the reporting file is located in *sge-root*/*cell*/`common`. You can change the default with the `-reporting` option of the `dbwriter` command.

For information about configuring the generation of the reporting file, see the `reporting_params` parameter of the `sge_conf`(5) man page, and the `report_variables` parameter of the `sge_host`(5) man page.

To enable the reporting file with `QMON`, on the Main Control window click the Cluster Configuration button, select the `global` host, and then click Modify.

On the Cluster Settings dialog box, click the Advanced Settings tab.

In the Reporting Parameters field, set the following parameters:

- Set `accounting` to `true`. `true` is the default value.
- Set `reporting` to `true`.
- Set `flush_time` to `00:00:15`. `00:00:15` is the default value.
- Set `joblog` to `true`.
- Set `sharelog` to `00:00:00`. `00:00:00` is the default value.

To enable the reporting file from the command line, use the `qconf –mconf` command to set the `reporting_params` attributes, as described in the preceding paragraph.

Once the reporting file is enabled, the `dbwriter` can read raw data from the reporting file and write it to the reporting database.

For more information about configuring the reporting file, see the `reporting`(5) man page. For complete details about installing and setting up ARCo, see Chapter 8, "Installing the Accounting and Reporting Console," in *N1 Grid Engine 6 Installation Guide*.

Calculating Derived Values With `dbwriter`

The rules for calculating derived values are specified in a derived tag, which is a sub tag of the `DbWriterConfig` tag. The following table lists the attributes of the derived tag:

Attribute	Description
`object`	The object for which data is aggregated. The object is one of the following: ■ `host` ■ `queue` ■ `project` ■ `department` ■ `user`

- group

`interval` The time range specifying how often to calculate the derived values. The time range is one of the following:

- hour
- day
- month
- year

`variable` The name of the variable to hold the calculated data.

The following table lists the subelements of the derived tag:

`sql` The SQL statement that calculates the derived values. The statement must produce the following columns:

- `time_start` – Together with `time_end`, specifies the time period for the calculated value
- `time_end`
- `value` – The calculated derived value

The SQL statement can contain the following placeholders. `dbwriter` replaces the placeholders for each query, based on a rule:

- `__time_start__` – Start time for the query. `dbwriter` searches for the last previously calculated derived value from this rule, and uses this timestamp as the start time for the next query.
- `__time_end__` – End time for the query. This timestamp specifies the end of the last passed time range. For example, if the time range is `day`, and if derived values are calculated at `00:30`, `00:00` is taken as `time_end`.
- `__key_0__`, `__key_1__`, . . . , `__key_n__` – Components of the primary key for the specified object type. For example, the `sge_hosts` table has the primary `h_hostname`. If a rule is processed for the host object type, one query is executed per entry in the `sge_hosts` table, the `__key_0__` placeholder in the SQL statement is replaced by the hostname.

  The `sge_queue` table has a composed primary key that is made up of `q_qname` and `q_hostname`.

`auto` `dbwriter` generates the SQL statement for the calculation of derived values.

The autogenerated SQL statement looks like the following template:

```
SELECT time_start, time_end, <function>(<value_field>) as value
FROM ( SELECT TRUNC( <timestart_field>, <interval> ) as time_start
              TRUNC( <timestart_field>, <interval> ) +
                    INTERVAL '1' <interval> as time_end,
              <value_field>
       FROM <object value table>
       WHERE <primary key field 0> = __key_0__
```

```
        AND <primary key field 1> = __key_1__
        AND . . .
        AND <parent key field> =
            (SELECT <parent key field> FROM <parent table>
             WHERE <parent filter> )
        AND <timestart_field> <= {ts __time_start__ }
        AND <timeend_field>   >  {ts __time_end__ }
GROUP BY time_start, time_end
```

The SQL template parameters are as follows:

Parameter	Description
<function>	Aggregate function for calculating the derived value. Comes from the function attribute of the auto tag in the XML file.
<value_field>	Depends on the object of the derived value.
<timestart_field>	Depends on the object of the derived value.
<timeend_field>	Depends on the object of the derived value.
<interval>	Comes from the interval attribute of the derived tag
<object value table>	Name of the database table where the values are stored. Depends on the object (host => host_values, user => user_values, . . .)
<primary key field n>	Primary key that is necessary to join the value table to the parent table. Depends on the object.
<parent key field>	Name of the field that holds the ID of the parent. Depends on the object.
<parent table>	Name of the parent database table. Depends on the object (host => host, user => user)
<parent filter>	Filter for the parent table. A derived value for each entry of the parent table is calculated, for example, u_user = 'user1').

Here is an example of an autogenerated SQL statement:

```
<derive object="host" interval="day" variable="d_load">
   <auto function="AVG" variable="h_load" />
</derive>

SELECT time_start, time_end, AGE(hv_dvalue)
FROM ( SELECT TRUNC( hv_time_start, 'day' ) as time_start,
              TRUNC( hv_time_start, 'day' ) +
                           INTERVAL '1' day as time_end,
              hv_dvalue
       FROM sge_host_values
       WHERE hv_variable = 'h_load' AND
             hv_parent =
               (SELECT h_id FROM sge_host
                WHERE h_hostname = 'foo.bar') AND
```

```
            hv_time_start <= {ts '2004-05-21 00:00:00.0'} AND
            hv_time_end    > {ts '2004-05-17 00:00:00.0'} )
GROUP BY time_start, time_end
```

Deleting Outdated Records With `dbwriter`

To delete outdated records in the reporting database, you must specify a deletion rule in the delete tag. The following table lists the attributes of the delete tag:

Attribute	Description
`scope`	The type of data to delete. Valid entries are the following: ■ `job` ■ `job_log` ■ `share_log` ■ `host_values` ■ `queue_values` ■ `project_values` ■ `department_values` ■ `user_values` ■ `group_values`
`time_range`	The unit of time_amount:
`time_amount`	Number of units (`time_range`) during which a record is to be kept.

The following table lists a subelement of the delete tag:

`sub_scope`	For certain scopes, a subscope can be configured. The subscope specifies an additional condition for deletion. A subscope can be configured for all `*_values` scopes and for the `share_log` scope. If a subscope is configured for a `*_values` rule, it contains a list of variables to delete, separated by spaces. If a subscope is specified for the `share_log`, it contains a list of share-tree nodes to delete, separated by spaces. If subscope are used, you should always have a fallback rule without subscope, which will delete all objects that are not explicitly named by the subscope.

Here is an example of a delete tag:

```
<?xml version="1.0" encoding="UTF-8"?>
<DbWriterConfig>
  <!-- keep host values for 2 years -->
  <delete scope="host_values" time_range="year" time_amount="2"/>
```

```
  <!-- keep queue values one month -->
  <delete scope="queue_values" time_range="month" time_amount="1">
    <sub_scope>slots</sub_scope>
    <sub_scope>state</sub_scope>
  </delete>
</DbWriterConfig>
```

Accounting and Usage Statistics (`qacct`)

You can use the `qacct` command to generate alphanumeric accounting statistics. If you specify no options, `qacct` displays the aggregate usage on all machines of the cluster, as generated by all jobs that have finished and that are contained in the cluster accounting file *sge-root*`/`*cell*`/common/accounting`. In this case, `qacct` reports three times, in seconds:

- Real time – Wall clock time, which is the time between when the job starts and when it finishes
- User time – CPU time spent in user processes
- System time – CPU time spent in system calls

Several options are available for reporting accounting information about queues, users, and the like. In particular, you can use the `qacct -l` command to request information about all jobs that have finished and that match a resource requirement specification.

Use the `qacct -j [`*job-id* | *job-name*`]` command to get direct access to the complete resource usage information stored by the grid engine system. This information includes the information that is provided by the `getrusage` system call.

The `-j` option reports the resource usage entry for the jobs with *job-id* or with *job-name*. 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.
- A checkpointing job that migrated is displayed.

See the `qacct`(1) man page for more information.

Backing Up the Grid Engine 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.
- The grid engine system cell directory.
- 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.

Using Files and Scripts for Administration Tasks

This section describes how to use files and scripts to add or modify grid engine system objects such as queues, hosts, and environments.

You can use the `QMON` graphical user interface to perform all administrative tasks in the grid engine system. You can also administer a grid engine system through commands you type at a shell prompt and call from within shell scripts. Many experienced administrators find that using files and scripts is a more flexible, quicker, and more powerful way to change settings.

Using Files to Add or Modify Objects

Use the `qconf` command with the following options to add objects according to specifications you create in a file:

```
qconf -Ae
qconf -Aq
qconf -Au
qconf -Ackpt
qconf -Ap
```

Use the `qconf` command with the following options to modify objects according to specifications you create in a file:

```
qconf -Me
qconf -Mq
qconf -Mu
qconf -Mckpt
qconf -Mp
```

The `-Ae` and `-Me` options add or modify execution hosts.

The `-Aq` and `-Mq` options add or modify queues.

The `-Au` and `-Mu` options add or modify usersets.

The `-Ackpt` and `-Mckpt` options add or modify checkpointing environments.

The `-Ap` and `-Mp` options add or modify parallel environments.

Use these options in combination with the `qconf -s` command to take an existing object and modify it. You can then update the existing object or create a new object.

EXAMPLE 7–1 Modifying the Migration Command of a Checkpoint Environment

```
#!/bin/sh
# ckptmod.sh: modify the migration command
# of a checkpointing environment
# Usage: ckptmod.sh <checkpoint-env-name> <full-path-to-command>
TMPFILE=tmp/ckptmod.$$

CKPT=$1
MIGMETHOD=$2

qconf -sckpt $CKPT | grep -v '^migr_command' > $TMPFILE
echo "migr_command $MIGMETHOD" >> $TMPFILE
qconf -Mckpt $TMPFILE
rm $TMPFILE
```

Using Files to Modify Queues, Hosts, and Environments

You can modify individual queues, hosts, parallel environments, and checkpointing environments from the command line. Use the `qconf` command in combination with other commands.

- If you have already prepared a file, type the `qconf` command with appropriate options:

  ```
  qconf -Me
  qconf -Mq
  qconf -Mckpt
  qconf -Mp
  ```

- If you have *not* prepared a file, type the `qconf` command with appropriate options:

  ```
  qconf -me
  qconf -mq
  qconf -mckpt
  qconf -mp
  ```

The -Me and -me options modify execution hosts.

The -Mq and -mq options modify queues.

The -Mckpt and -mckpt options modify checkpointing environments.

The -Mp and -mp options modify parallel environments.

The difference between the uppercase -M options and the lowercase -m options controls the qconf command's result. Both -M and -m mean *modify*, but the uppercase -M denotes modification from an existing file, whereas the lowercase -m does not. Instead, the lowercase -m opens a temporary file in an editor. When you save any changes you make to this file and exit the editor, the system immediately reflects those changes.

However, when you want to change many objects at once, or you want to change object configuration noninteractively, use the qconf command with the options that modify object attributes (such as -Aattr, -Mattr, and so forth).

The following commands make modifications according to specifications in a *file*:

```
qconf -Aattr {queue | exechost | pe | ckpt} filename
qconf -Mattr {queue | exechost | pe | ckpt} filename
qconf -Rattr {queue | exechost | pe | ckpt} filename
qconf -Dattr {queue | exechost | pe | ckpt} filename
```

The following commands make modifications according to specifications on the *command line*:

```
qconf -aattr {queue | exechost | pe | ckpt} attribute value {queue-list | host-list}
qconf -mattr {queue | exechost | pe | ckpt} attribute value {queue-list | host-list}
qconf -rattr {queue | exechost | pe | ckpt} attribute value {queue-list | host-list}
qconf -dattr {queue | exechost | pe | ckpt} attribute value {queue-list | host-list}
```

The -Aattr and -aattr options add attributes.

The -Mattr and -mattr options modify attributes.

The -Rattr and -rattr options replace attributes.

The -Dattr and -dattr options delete attributes.

filename is the name of a file that contains attribute-value pairs.

attribute is the queue or host attribute that you want to change.

value is the value of the attribute you want to change.

The -aattr, -mattr, and -dattr options enable you to operate on individual values in a list of values. The -rattr option replaces the entire list of values with the new one that you specify, either on the command line or in the file.

EXAMPLE 7–2 Changing the Queue Type

The following command changes the queue type of `tcf27-e019.q` to batch only:

```
% qconf -rattr queue qtype batch tcf27-e019.q
```

EXAMPLE 7–3 Modifying the Queue Type and the Shell Start Behavior

The following command uses the file `new.cfg` to modify the queue type and the shell start behavior of `tcf27-e019.q`:

```
% cat new.cfg
qtype batch interactive checkpointing
shell_start_mode unix_behavior
% qconf -Rattr queue new.cfg tcf27-e019.q
```

EXAMPLE 7–4 Adding Resource Attributes

The following command adds the resource attribute `scratch1` with a value of 1000M and the resource attribute `long` with a value of 2:

```
% qconf -rattr exechost complex_values scratch1=1000M,long=2 tcf27-e019
```

EXAMPLE 7–5 Attaching a Resource Attribute to a Host

The following command attaches the resource attribute `short` to the host with a value of 4:

```
% qconf -aattr exechost complex_values short=4 tcf27-e019
```

EXAMPLE 7–6 Changing a Resource Value

The following command changes the value of `scratch1` to 500M, leaving other values unchanged:

```
% qconf -mattr exechost complex_values scratch-=500M tcf27-e019
```

EXAMPLE 7–7 Deleting a Resource Attribute

The following command deletes the resource attribute `long`:

```
% qconf -dattr exechost complex_values long tcf27-e019
```

EXAMPLE 7–8 Adding a Queue to the List of Queues for a Checkpointing Environment

The following command adds `tcf27-b011.q` to the list of queues for the checkpointing environment `sph`:

```
% qconf -aattr ckpt queue_list tcf27-b011.q sph
```

EXAMPLE 7–9 Changing the Number of Slots in a Parallel Environment

The following command changes the number of slots in the parallel environment `make` to 50:

```
% qconf -mattr pe slots 50 make
```

Targeting Queue Instances with the `qselect` Command

The `qselect` command outputs a list of queue instances. If you specify options, `qselect` lists only the queue instances that match the criteria you specify. You can use `qselect` in combination with the `qconf` command to target specific queue instances that you want to modify.

EXAMPLE 7–10 Listing Queues

The following command lists all queue instances on Linux machines:

```
% qselect -l arch=glinux
```

The following command lists all queue instances on machines with two CPUs:

```
% qselect -l num_proc=2
```

The following command lists all queue instances on all four-CPU 64–bit Solaris machines:

```
% qselect -l arch=solaris64,num_proc=4
```

The following command lists queue instances that provide an application license. The queue instances were previously configured.

```
% qselect -l app_lic=TRUE
```

You can combine `qselect` with `qconf` to do wide-reaching changes with a single command line. To do this, put the entire `qselect` command inside backward quotation marks (`` ` ` ``) and use it in place of the queue-list variable on the `qconf` command line.

EXAMPLE 7–11 Using `qselect` in `qconf` Commands

The following command sets the `prolog` script to `sol_prolog.sh` on all queue instances on Solaris machines:

```
% qconf -mattr queue prolog /usr/local/scripts/sol_prolog.sh `qselect -l arch=solaris`
```

The following command sets the attribute `fluent_license` to two on all queue instances on two-processor systems:

```
% qconf -mattr queue complex_values fluent_license=2 `qselect -l num_proc=2`
```

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.

Using Files to Modify a Global Configuration or the Scheduler

To change a global configuration, use the `qconf -mconf` command. To change the scheduler, use the `qconf -msconf` command.

Both of these commands open a temporary file in an editor. When you exit the editor, any changes that you save to this temporary file are processed by the system and take effect immediately. The editor used to open the temporary file is the editor specified by the `EDITOR` environment variable. If this variable is undefined, the `vi` editor is used by default.

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.

EXAMPLE 7–12 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
```

EXAMPLE 7–12 Modifying the Schedule Interval *(Continued)*

```
     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.

CHAPTER 8

Fine Tuning, Error Messages, and Troubleshooting

This chapter describes some ways to fine-tune your grid engine system environment. The chapter also describes the error messaging procedures and offers tips on how to resolve various common problems.

This chapter includes the following sections:

Fine-Tuning Your Grid 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 to find out why certain jobs are not dispatched. However, providing this information for all jobs at all times can consume resources. You usually do not need this much information.

To disable scheduler monitoring, set `schedd_job_info` to false in the scheduler configuration. See "Changing the Scheduler Configuration With QMON" on page 123, 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 "Adding and Modifying Global and Host Configurations With QMON" on page 41, and the `sge_conf`(5) man page.

Job Validation

Forced validation at job submission time can be a valuable procedure to prevent nondispatchable 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. See "Submitting Advanced Jobs With QMON" in *N1 Grid Engine 6 User's Guide*, 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" on page 53, 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 oversubscription 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 "Changing the Scheduler Configuration With QMON" on page 123, 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 "Changing the Scheduler Configuration With QMON" on page 123, 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.

How the Grid Engine Software Retrieves Error Reports

The grid engine software reports errors and warnings by logging messages into certain files or by sending email, or both. The log files include message files and job `STDERR` output.

As soon as a job is started, the standard error (`STDERR`) output of the job script is redirected to a file. The default file name and location are used, or you can specify the filename and the location with certain options of the `qsub` command. See the grid engine system man pages for detailed information.

Separate messages files exist for the `sge_qmaster`, the `sge_schedd`, and the `sge_execd`s. The files have the same file name: `messages`. The `sge_qmaster` log file resides in the master spool directory. The `sge_schedd` message file resides in the scheduler spool directory. The execution daemons' log files reside in the spool directories of the execution daemons. See "Spool Directories Under the Root Directory" in *N1 Grid Engine 6 Installation Guide* for more information about the spool directories.

Each message takes up a single line in the files. Each message is subdivided into five components separated by the vertical bar sign (`|`).

The components of a message are as follows:

1. The first component is a time stamp for the message.
2. The second component specifies the daemon that generates the message.
3. The third component is the name of the host where the daemon runs.
4. The fourth is a message type. The message type is one of the following:
   - `N` for notice – for informational purposes
   - `I` for info – for informational purposes
   - `W` for warning
   - `E` for error – an error condition has been detected
   - `C` for critical – can lead to a program abort

   Use the `loglevel` parameter in the cluster configuration to specify on a global basis or a local basis what message types you want to log.
5. The fifth component is the message text.

Note – If an error log file is not accessible for some reason, the grid engine system tries to log the error message to the files `/tmp/sge_qmaster_messages`, `/tmp/sge_schedd_messages`, or `/tmp/sge_execd_messages` on the corresponding host.

In some circumstances, the grid engine system notifies users, administrators, or both, about error events by email. The email messages sent by the grid engine system do not contain a message body. The message text is fully contained in the mail subject field.

Consequences of Different Error or Exit Codes

The following table lists the consequences of different job-related error codes or exit codes. These codes are valid for every type of job.

TABLE 8–1 Job-Related Error or Exit Codes

Script/Method	Exit or Error Code	Consequence
Job script	0	Success
	99	Requeue
	Rest	Success: exit code in accounting file
prolog/epilog	0	Success
	99	Requeue
	Rest	Queue error state, job requeued

The following table lists the consequences of error codes or exit codes of jobs related to parallel environment (PE) configuration.

TABLE 8–2 Parallel-Environment-Related Error or Exit Codes

Script/Method	Exit or Error Code	Consequence
pe_start	0	Success
	Rest	Queue set to error state, job requeued
pe_stop	0	Success

TABLE 8–2 Parallel-Environment-Related Error or Exit Codes *(Continued)*

Script/Method	Exit or Error Code	Consequence
	Rest	Queue set to error state, job not requeued

The following table lists the consequences of error codes or exit codes of jobs related to queue configuration. These codes are valid only if corresponding methods were overwritten.

TABLE 8–3 Queue-Related Error or Exit Codes

Script/Method	Exit or Error Code	Consequence
Job starter	0	Success
	Rest	Success, no other special meaning
Suspend	0	Success
	Rest	Success, no other special meaning
Resume	0	Success
	Rest	Success, no other special meaning
Terminate	0	Success
	Rest	Success, no other special meaning

The following table lists the consequences of error or exit codes of jobs related to checkpointing.

TABLE 8–4 Checkpointing-Related Error or Exit Codes

Script/Method	Exit or Error Code	Consequence
Checkpoint	0	Success
	Rest	Success. For kernel checkpoint, however, this means that the checkpoint was not successful.
Migrate	0	Success
	Rest	Success. For kernel checkpoint, however, this means that the checkpoint was not successful. Migration will occur.

TABLE 8–4 Checkpointing-Related Error or Exit Codes *(Continued)*

Script/Method	Exit or Error Code	Consequence
Restart	0	Success
	Rest	Success, no other special meaning
Clean	0	Success
	Rest	Success, no other special meaning

Running Grid Engine System Programs in Debug Mode

For some severe error conditions, the error-logging mechanism might not yield sufficient information to identify the problems. Therefore, the grid engine system offers the ability to run almost all ancillary programs and the daemons in *debug* mode. Different debug levels vary in the extent and depth of information that is provided. The debug levels range from zero through 10, with 10 being the level delivering the most detailed information and zero turning off debugging.

To set a debug level, an extension to your `.cshrc` or `.profile` resource files is provided with the distribution of the grid engine system. For `csh` or `tcsh` users, the file *sge-root*`/util/dl.csh` is included. For `sh` or `ksh` users, the corresponding file is named *sge-root*`/util/dl.sh`. The files must be sourced into your standard resource file. As `csh` or `tcsh` user, include the following line in your `.cshrc` file:

```
source sge-root/util/dl.csh
```

As `sh` or `ksh` user, include the following line in your `.profile` file:

```
. sge-root/util/dl.sh
```

As soon as you log out and log in again, you can use the following command to set a debug level:

```
% dl level
```

If *level* is greater than 0, starting a grid engine system command forces the command to write trace output to `STDOUT`. The trace output can contain warning messages, status messages, and error messages, as well as the names of the program modules that are called internally. The messages also include line number information, which is helpful for error reporting, depending on the debug level you specify.

Note – To watch a debug trace, you should use a window with a large scroll-line buffer. For example, you might use a scroll-line buffer of 1000 lines.

Note – If your window is an `xterm`, you might want to use the `xterm` logging mechanism to examine the trace output later on.

If you run one of the grid engine system daemons in debug mode, the daemons keep their terminal connection to write the trace output. You can abort the terminal connections by typing the interrupt character of the terminal emulation you use. For example, you might use `Control-C`.

To switch off debug mode, set the debug level back to 0.

Setting the `dbwriter` Debug Level

The `sgedbwriter` script starts the `dbwriter` program. The script is located in *sge_root*`/dbwriter/bin/sgedbwriter`. The `sgedbwriter` script reads the `dbwriter` configuration file, `dbwriter.conf`. This configuration file is located in *sge_root*/*cell*`/common/dbwriter.conf`. This configuration file sets the debug level of `dbwriter`. For example:

```
#
# Debug level
# Valid values: WARNING, INFO, CONFIG, FINE, FINER, FINEST, ALL
#
DBWRITER_DEBUG=INFO
```

You can use the `-debug` option of the `dbwriter` command to change the number of messages that the `dbwriter` produces. In general, you should use the default debug level, which is `info`. If you use a more verbose debug level, you substantially increase the amount of data output by `dbwriter`.

You can specify the following debug levels:

`warning`	Displays only severe errors and warnings.
`info`	Adds a number of informational messages. `info` is the default debug level.
`config`	Gives additional information that is related to `dbwriter` configuration, for example, about the processing of rules.
`fine`	Produces more information. If you choose this debug level, all SQL statements run by `dbwriter` are output.
`finer`	For debugging.

`finest`	For debugging.
`all`	Displays information for all levels. For debugging.

Diagnosing Problems

The grid engine system offers several reporting methods to help you diagnose problems. The following sections outline their uses.

Pending Jobs Not Being Dispatched

Sometimes a pending job is obviously capable of being run, but the job does not get dispatched. To diagnose the reason, the grid engine system offers a pair of utilities and options, `qstat -j` *job-id* and `qalter-w v` *job-id*.

- `qstat -j` *job-id*

  When enabled, `qstat -j` *job-id* provides a list of reasons why a certain job was not dispatched in the last scheduling run. This monitoring can be enabled or disabled. You might want to disable monitoring because it can cause undesired communication overhead between the `sge_schedd` daemon and `sge_qmaster`. See `schedd_job_info` in the `sched_conf`(5) man page. The following example shows output for a job with the ID `242059`:

```
% qstat -j 242059
scheduling info: queue "fangorn.q" dropped because it is temporarily not available
queue "lolek.q" dropped because it is temporarily not available
queue "balrog.q" dropped because it is temporarily not available
queue "saruman.q" dropped because it is full
cannot run in queue "bilbur.q" because it is not contained in its hard queuelist (-q)

cannot run in queue "dwain.q" because it is not contained in its hard queue list (-q)
has no permission for host "ori"
```

  This information is generated directly by the `sge_schedd` daemon. The generating of this information takes the current usage of the cluster into account. Sometimes this information does not provide what you are looking for. For example, if all queue slots are already occupied by jobs of other users, no detailed message is generated for the job you are interested in.

- `qalter -w v` *job-id*

This command lists the reasons why a job is not dispatchable in principle. For this purpose, a dry scheduling run is performed. All consumable resources, as well as all slots, are considered to be fully available for this job. Similarly, all load values are ignored because these values vary.

Job or Queue Reported in Error State `E`

Job or queue errors are indicated by an uppercase `E` in the `qstat` output.

A job enters the error state when the grid engine system tries to run a job but fails for a reason that is specific to the job.

A queue enters the error state when the grid engine system tries to run a job but fails for a reason that is specific to the queue.

The grid engine system offers a set of possibilities for users and administrators to gather diagnosis information in case of job execution errors. Both the queue and the job error states result from a failed job execution. Therefore the diagnosis possibilities are applicable to both types of error states.

- **User abort mail.** If jobs are submitted with the `qsub -m a` command, abort mail is sent to the address specified with the `-M` *user*`[@`*host*`]` option. The abort mail contains diagnosis information about job errors. Abort mail is the recommended source of information for users.
- **`qacct` accounting.** If no abort mail is available, the user can run the `qacct -j` command. This command gets information about the job error from the grid engine system's job accounting function.
- **Administrator abort mail.** An administrator can order administrator mails about job execution problems by specifying an appropriate email address. See under `administrator_mail` on the `sge_conf`(5) man page. Administrator mail contains more detailed diagnosis information than user abort mail. Administrator mail is the recommended method in case of frequent job execution errors.
- **Messages files.** If no administrator mail is available, you should investigate the `qmaster messages` file first. You can find entries that are related to a certain job by searching for the appropriate job ID. In the default installation, the `sge_qmaster messages` file is *sge-root*`/`*cell*`/spool/qmaster/messages`.

  You can sometimes find additional information in the messages of the `sge_execd` daemon from which the job was started. Use `qacct -j` *job-id* to discover the host from which the job was started, and search in *sge-root*`/`*cell*`/spool/`*host*`/messages` for the job ID.

Troubleshooting Common Problems

This section provides information to help you diagnose and respond to the cause of common problems.

- **Problem** — The output file for your job says, `Warning: no access to tty; thus no job control in this shell....`
  - **Possible cause** — One or more of your login files contain an `stty` command. These commands are useful only if a terminal is present.
  - **Possible solution** — No terminal is associated with batch jobs. You must remove all `stty` commands from your login files, or you must bracket such commands with an `if` statement. The `if` statement should check for a terminal before processing. The following example shows an `if` statement:

    ```
    /bin/csh:
    stty -g             # checks terminal status
    if ($status == 0)   # succeeds if a
    terminal is present
    <put all stty commands in here>
    endif
    ```

- **Problem** — The job standard error log file says `'tty':Ambiguous`. However, no reference to `tty` exists in the user's shell that is called in the job script.
  - **Possible cause** — `shell_start_mode` is, by default, `posix_compliant`. Therefore all job scripts run with the shell that is specified in the queue definition. The scripts do not run with the shell that is specified on the first line of the job script.
  - **Possible solution** — Use the `-S` flag to the `qsub` command, or change `shell_start_mode` to `unix_behavior`.
- **Problem** — You can run your job script from the command line, but the job script fails when you run it using the `qsub` command.
  - **Possible cause** — Process limits might be being set for your job. To test whether limits are being set, write a test script that performs `limit` and `limit -h` functions. Run both functions interactively, at the shell prompt and using the `qsub` command, to compare the results.
  - **Possible solution** — Remove any commands in configuration files that sets limits in your shell.
- **Problem** — Execution hosts report a load of `99.99`.
  1. **Possible cause** — The `sge_execd` daemon is not running on the host.

     Possible solution — As root, start up the `sge_execd` daemon on the execution host by running the *sge-root*`/`*cell*`/common/sgeexecd` script.
  2. **Possible cause** — A default domain is incorrectly specified.

Possible solution — As the grid engine system administrator, run the `qconf -mconf` command and change the `default_domain` variable to `none`.

3. **Possible cause** — The `sge_qmaster` host sees the name of the execution host as different from the name that the execution host sees for itself.

   Possible solution — If you are using DNS to resolve the host names of your compute cluster, configure `/etc/hosts` and NIS to return the fully qualified domain name (FQDN) as the primary host name. Of course, you can still define and use the short alias name, for example, `168.0.0.1 myhost.dom.com myhost`.

   If you are *not* using DNS, make sure that all of your `/etc/hosts` files and your NIS table are consistent, for example, `168.0.0.1 myhost.corp myhost` or `168.0.0.1 myhost`

- **Problem** — Every 30 seconds a warning that is similar to the following message is printed to *cell*`/spool/`*host*`/messages`:

  ```
  Tue Jan 23 21:20:46 2001|execd|meta|W|local
  configuration meta not defined - using global configuration
  ```

  But *cell*`/common/local_conf` contains a file for each host, with FQDN.

  - **Possible cause** — The host name resolving at your machine `meta` returns the short name, but at your master machine, `meta` with FQDN is returned.
  - **Possible solution** — Make sure that all of your `/etc/hosts` files and your NIS table are consistent in this respect. In this example, a line such as the following text could erroneously be included in the `/etc/hosts` file of the host `meta`:

    `168.0.0.1 meta meta.your.domain`

    The line should instead be:

    `168.0.0.1 meta.your.domain meta`.

- **Problem** — Occasionally you see `CHECKSUM ERROR`, `WRITE ERROR`, or `READ ERROR` messages in the `messages` files of the daemons.

  - **Possible cause** — As long as these messages do not appear in a one-second interval, you need not do anything. These messages typically can appear between 1 and 30 times a day.

- **Problem** — Jobs finish on a particular queue and return the following message in `qmaster/messages`:

  ```
  Wed Mar 28 10:57:15 2001|qmaster|masterhost|I|job 490.1
  finished on host exechost
  ```

  Then you see the following error messages in the execution host's `exechost/messages` file:

  ```
  Wed Mar 28 10:57:15 2001|execd|exechost|E|can't find directory
  "active_jobs/490.1" for reaping job 490.1
  ```

```
Wed Mar 28 10:57:15 2001|execd|exechost|E|can't remove directory
"active_jobs/490.1": opendir(active_jobs/490.1) failed:
Input/output error
```

- **Possible cause** — The *sge-root* directory, which is automounted, is being unmounted, causing the `sge_execd` daemon to lose its current working directory.
- **Possible solution** — Use a local spool directory for your `sge_execd` host. Set the parameter `execd_spool_dir`, using `QMON` or the `qconf` command.

- **Problem** — When submitting interactive jobs with the `qrsh` utility, you get the following error message:

```
% qrsh -l mem_free=1G error: error: no suitable queues
```

  However, queues are available for submitting batch jobs with the `qsub` command. These queues can be queried using `qhost -l mem_free=1G` and `qstat -f -l mem_free=1G`.

  - **Possible cause** — The message `error: no suitable queues` results from the `-w e` submit option, which is active by default for interactive jobs such as `qrsh`. Look for `-w e` on the `qrsh`(1) man page. This option causes the submit command to fail if the `sge_qmaster` does not know for sure that the job is dispatchable according to the current cluster configuration. The intention of this mechanism is to decline job requests in advance, in case the requests can't be granted.
  - **Possible solution** — In this case, `mem_free` is configured to be a consumable resource, but you have not specified the amount of memory that is to be available at each host. The memory load values are deliberately not considered for this check because memory load values vary. Thus they can't be seen as part of the cluster configuration. You can do one of the following:
    - **Omit this check generally** by explicitly overriding the `qrsh` default option `-w e` with the `-w n` option. You can also put this command into *sge-root*`/`*cell*`/common/sge_request`.
    - **If you intend to manage** `mem_free` as a consumable resource, specify the `mem_free` capacity for your hosts in `complex_values` of `host_conf` by using `qconf -me` *hostname*.
    - **If you don't intend to manage** `mem_free` as a consumable resource, make it a nonconsumable resource again in the `consumable` column of `complex`(5) by using `qconf -mc` *hostname*.

- **Problem** — `qrsh` won't dispatch to the same node it is on. From a `qsh` shell you get a message such as the following:

```
host2 [49]% qrsh -inherit host2 hostname
error: executing task of job 1 failed:

host2 [50]% qrsh -inherit host4 hostname
host4
```

- **Possible cause** — `gid_range` is not sufficient. `gid_range` should be defined as a range, not as a single number. The grid engine system assigns each job on a host a distinct `gid`.
- **Possible solution** — Adjust the `gid_range` with the `qconf -mconf` command or with `QMON`. The suggested range is as follows:

```
gid_range                20000-20100
```

- **Problem** — `qrsh -inherit -V` does not work when used inside a parallel job. You get the following message:

```
cannot get connection to "qlogin_starter"
```

  - **Possible cause** — This problem occurs with nested `qrsh` calls. The problem is caused by the `-V` option. The first `qrsh -inherit` call sets the environment variable `TASK_ID`. `TASK_ID` is the ID of the tightly integrated task within the parallel job. The second `qrsh -inherit` call uses this environment variable for registering its task. The command fails as it tries to start a task with the same ID as the already-running first task.
  - **Possible solution** — You can either unset `TASK_ID` before calling `qrsh -inherit`, or use the `-v` option instead of `-V`. This option exports only the environment variables that you really need.

- **Problem** — `qrsh` does not seem to work at all. Messages like the following are generated:

```
host2$ qrsh -verbose hostname
local configuration host2 not defined - using global configuration
waiting for interactive job to be scheduled ...
Your interactive job 88 has been successfully scheduled.
Establishing /share/gridware/utilbin/solaris64/rsh session
to host exehost ...
rcmd: socket: Permission denied
/share/gridware/utilbin/solaris64/rsh exited with exit code 1
reading exit code from shepherd ...
error: error waiting on socket for client to connect:
Interrupted system call
error: error reading return code of remote command
cleaning up after abnormal exit of
/share/gridware/utilbin/solaris64/rsh
host2$
```

  - **Possible cause** — Permissions for `qrsh` are not set properly.
  - **Possible solution** — Check the permissions of the following files, which are located in *sge-root*`/utilbin/`. Note that `rlogin` and `rsh` must be `setuid` and owned by root.

```
-r-s--x--x 1 root      root   28856  Sep 18  06:00  rlogin*
-r-s--x--x 1 root      root   19808  Sep 18  06:00  rsh*
-rwxr-xr-x 1 sgeadmin  adm   128160  Sep 18  06:00  rshd*
```

Note – The *sge-root* directory also needs to be NFS-mounted with the `setuid` option. If *sge-root* is mounted with `nosuid` from your submit client, `qrsh` and associated commands will not work.

- **Problem** – When you try to start a distributed make, `qmake` exits with the following error message:

```
qrsh_starter: executing child process
qmake failed: No such file or directory
```

  - **Possible cause** — The grid engine system starts an instance of `qmake` on the execution host. If the grid engine system environment, especially the `PATH` variable, is not set up in the user's shell resource file (`.profile` or `.cshrc`), this `qmake` call fails.
  - **Possible solution** — Use the `-v` option to export the `PATH` environment variable to the `qmake` job. A typical `qmake` call is as follows:

```
qmake -v PATH -cwd -pe make 2-10 --
```

- **Problem** — When using the `qmake` utility, you get the following error message:

```
waiting for interactive job to be scheduled ...timeout (4 s)
expired while waiting on socket fd 5

Your "qrsh" request could not be scheduled, try again later.
```

  - **Possible cause** — The `ARCH` environment variable might be set incorrectly in the shell from which `qmake` was called.
  - **Possible solution** – Set the `ARCH` variable correctly to a supported value that matches an available host in your cluster, or else specify the correct value at submit time, for example, `qmake -v ARCH=solaris64 ...`

CHAPTER 9

Configuring DBWriter

The`dbwriter` component writes and deletes the reporting data in the reporting database. It performs the following tasks:

- Reads raw data from reporting files and writes this raw data into the reporting database.
- Calculates derived values. You can configure which values are calculated and the rules of how to calculate them.
- Deletes outdated data. You can configure how long to keep the data.

The `sge_qmaster` component generates the reporting files. You can configure the generation of the reporting files, see the attribute *reporting_params* in the man page `sge_conf(5)`, and the attribute *report_variables* in the man page `sge_host(5)`.

Setup

The installation procedure sets up these parameters. A script for starting up the `dbwriter` is provided with the reporting module. Please see the N1 Grid Engine 6 Installation Guide for details. The following parameters have to be set for `dbwriter`.

Database System

The `dbwriter` can connect to different brands of database systems (supported systems are PostgreSQL and Oracle). The following parameters have to be set:

- *DRIVER* — to the name of the JDBC driver to use, for example, `org.postgresql.Driver`
- *DRIVERJAR* — to the jar archive containing the JDBC driver, for example, `lib/postgres43.jar`.

Database Server

Which database on which host to use is set by configuring the JDBC URL: The URL parameter is set to the JDBC URL of the database to use. Follow the guidelines of the database vendor for the syntax to use. For, example for a PostgreSQL database: `jdbc:postgresql://<hostname>:5432/arco`

Base Directory for Reporting Files

The path where `dbwriter` will find reporting files is set in the `REPORTING_FILE` variable. The base directory is typically set to `$SGE_ROOT/$SGE_CELL/common`

Configuration

The task of setting up these parameters will be done by the installation procedure. You can configure the behavior of the `dbwriter` with a number of command line parameters.

Interval

The *-interval* parameter sets the interval, in which `dbwriter` looks for new reporting files. If a reporting file is found, it is read and data is written to the reporting database.

Pid

The *-pid* parameter defines the path to the pid file. The `dbwriter` writes at startup a pid file. This contains the process id of the `dbwriter`. At shutdown of the `dbwriter` this file will be deleted.

PidCmd

The *-pidCmd* parameter defines a command which will be executed by the `dbwriter` to determine it's process id. This command should print it's parent process id to `stdout`. The `dbwriter` is a java application. The java virtual machine cannot determine its own process id. The default value of the *pidCmd* is `$SGE_ROOT/utilbin/$ARCH/checkprog -ppid`.

Continuous Mode

The *-continuous* parameter switches on the continuous mode. Without *-continuous*, `dbwriter` will perform its tasks just once. If continuous mode is switched on, it will run continuously and perform its tasks in each interval set with the *-interval* switch.

Debug Level

You can use the *-debug* option to configure the amount of messages output by `dbwriter`. A parameter to the *-debug* option is the debug level. In general, using the default debug level (`-info`) should be the preferred choice. Using more verbose debug levels greatly increases the amount of data output by `dbwriter`. You can specify the following debug levels:

- `-warning`: Display only severe errors and warnings
- info: Add a number of informational messages. This level is the default, if the -display switch isn't used
- `-config`: Give additional information that is related to `dbwriter` configuration, e.g. about the processing of rules (derived values or delete rules)
- `-fine`: Output more information. If this level is chosen, all SQL statements executed by `dbwriter` will be output.
- `-finer`: for debugging
- `-finest`: for debugging
- `-all`: Display information for all levels (only for debugging purposes).

Reporting File

N1 Grid Engine 6 writes one report file containing data of different types:

- host load values and consumables
- queue consumables
- job logging
- job accounting
- sharetree usage

The `dbwriter` command line parameter *-reporting* has to specify the path to the reporting file. The `dbwriter` component automatically parses the reporting file; once it has completed processing and has stored all the information into the database, it deletes the reporting file.

Calculation of Derived Values

At `dbwriter` startup, and in continuous mode once an hour, derived values are calculated. You can configure which values to calculate in an XML file, which is by default in `$SGE_ROOT/dbwriter/database/<database_type>/dbwriter.xml`. `<database_type>` defines the type of database being used; currently, Oracle and Postgres are supported. The path to the configuration file is passed to `dbwriter` using the *-calculation* parameter.

The configuration file uses an XML format, and contains entries of rules for both derived values and deleted values (described in the next section). The rules for derived values have the following format.

Derived Values Format

1. The top-level start tag is `<derive>` It must be specified with three attributes:
   - object — which can be host, queue, user, group, department or project. Based on this attribute, the derived value is ultimately stored in one of: `sge_host_values`, `sge_queue_values`, `sge_user_values`, `sge_group_values`, `sge_department_values`, `sge_project_values`.
   - interval — which can be hour, day, month, or year.
   - variable — which is the name of the new derived value.
2. A second-level start tag, either `<sql>` or `<auto>`, describing the way the value should be derived. These tags are shown in detail as follows.
3. `<sql>` – This tag contains an SQL statement used for calculating the derived values. The exact syntax of the entries depends upon the type of database being used.
4. `<auto>` - for certain simple derived values, this tag can be used instead of a full SQL query. This tag has two attributes:
   - function — which gives the aggregate function to apply to the variable. This can be any function valid for the type of database being used. Some typical functions are AVG, SUM, VALUE, COUNT, MIN or MAX.
   - variable — which can be any variable tracked in the following tables: `sge_host_values`, `sge_queue_values`, `sge_user_values`, `sge_group_values`, `sge_department_values`, `sge_project_values` the variable specified must be from the table indicated by the object attribute of the enclosing `<derive>` tag, for example, if the object is host, the variable must be found in `sge_host_values`.
5. Two end tags matching the two start tags

Examples

Here is an example of a derivation rule using the <sql> tag. The sge_queue table has a composed primary key comprising q_qname and q_hostname. For a rule specified for the queue object_type, a query will be made for each entry in the sge_queue table, the placeholders __key_0__ will be replaced by the queue name,ilwill be replaced by the hostname.

```
<!--
        average queue utilization per hour
        -->
<derive object="queue" interval="hour" variable="h_utilized">
        <sql>
        SELECT DATE_TRUNC( 'hour', qv_time_start)
                        AS time_start,
                DATE_TRUNC( 'hour', qv_time_start) + INTERVAL '1 hour'
                        AS time_end,
                AVG(qv_dvalue * 100 / qv_dconfig)
                        AS value
                FROM sge_queue_values
                WHERE qv_variable = 'slots' AND
                        qv_parent = (SELECT q_id FROM sge_queue
                                WHERE q_qname = __key_0__
                                        AND q_hostname = __key_1__)
                        AND qv_time_start &lt;= '__time_end__' AND
                        qv_time_end &gt; '__time_start__'
                GROUP BY time_start
</sql>
</derive>
```

Here is an example of a derivation rule using the <auto> tag.

```
<!-- average load per hour -->
<derive object="host" interval="hour" variable="h_load">
        <auto function="AVG" variable="np_load_avg" />
</derive>
```

Deleting Outdated Records

At dbwriter startup, and in continuous mode once an hour, outdated records will be deleted. Which values to calculate can be configured in an XML file, by default in $SGE_ROOT/dbwriter/database/<database_type>/dbwriter.xml. <database_type> is the type of database being used; currently, Oracle and Postgres are supported. The path to the configuration file is passed to dbwriter using the *-deletion* parameter.

The configuration file uses an XML format, and contains entries of rules for both derived values and deleted values. Deletion rules are of the following format.

- A top-level start tag <delete> with three attributes:
  - scope — which specifies the type of data to be deleted. Valid entries are *job*, *job_log*, *share_log*, *ip, queue_values*, ip, department_values, *user_values*, *group_values*. Based on this attribute, the values are deleted from the table with

the same name with `sge_ prepended`.

- time_range — which gives the unit of time_amount.
- time_amount — which is the number of units (time_range) a record is kept.

- An optional second-level start tag `<sub_scope>`, which specifies an additional condition for deletion. A subscope can be configured for all `*_values` scopes and the `share_log` scope.
- One or Two end tags matching the two start tags

If a subscope is configured for a **_values* rule, it contains a space separated list of variables to delete. If a subscope is specified for the `share_log`, it contains a space separated list of sharetree nodes to delete.

Examples

The following rule indicates that the four variables given in the subscope should be deleted from the table `sge_host_values` after 7 days.

```
<delete scope="host_values" time_range="day" time_amount="7">
        <sub_scope>np_load_avg</sub_scope>
        <sub_scope>cpu</sub_scope>
        <sub_scope>mem_free</sub_scope>
        <sub_scope>virtual_free</sub_scope>
</delete>
```

The following rule says to delete all variables from the table `sge_host_values` after 2 years.

```
<delete scope="host_values" time_range="year" time_amount="2"/>
```

The following rule says to delete all records for user `fred` after 1 month

```
<delete scope="share_log" time_range="month" time_amount="1">
        <sub_scope>fred</sub_scope>
</delete>
```

Index

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X