Sun™ ONE Grid Engine Administration and User's Guide

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054 U.S.A.
650-960-1300

Part No. 816-2077-12
October 2002, Revision A

Send comments about this document to: docfeedback@sun.com

Please
Recycle

Adobe PostScript

Contents

Preface

The *Sun Grid Engine 5.3 Administration and User's Guide* is a comprehensive manual that provides background information about the product, installation instructions, and instructions on how to use the product fully.

How This Book Is Organized

Because this guide is intended both for users of the Sun Grid Engine 5.3 product and for system administrators whose product responsibilities are not always the same as those of users, this guide is divided into four parts. Each part contains information of special interest to the user or to the administrator.

Descriptions of the parts and their intended audiences follow.

- Part 1 – Background and Definitions

  Intended for users and administrators alike, this part of the *Sun Grid Engine 5.3 Administration and User's Guide* provides a detailed explanation of the product's uses, components, terminology, and so forth.

- Part 2 – Getting Started

  Intended for those who will install the product—administrators, generally—this part of the guide includes detailed instructions for "quick-start," full, and upgrade installations.

- Part 3 – Using Sun Grid Engine 5.3 Software

  This part of the guide is intended both for the user and the administrator. Included are instructions and background information that cover many tasks.

- Part 4 – Administration

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

Using UNIX Commands

This document might not contain information on basic UNIX® commands and procedures such as shutting down the system, booting the system, and configuring devices.

See one or more of the following for this information:

- *Solaris Handbook for Sun Peripherals*
- AnswerBook2™ online documentation for the Solaris™ operating environment
- Other software documentation that you received with your system

Typographic Conventions

Typeface	Meaning	Examples
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `% You have mail.`
`AaBbCc123`	What you type, when contrasted with on-screen computer output	`% su` `Password:`
AaBbCc123	Book titles, new words or terms, words to be emphasized. Replace command-line variables with real names or values.	Read Chapter 6 in the *User's Guide.* These are called *class* options. You *must* be superuser to do this. To delete a file, type `rm` *filename.*

Shell Prompts

Shell	Prompt
C shell	*machine-name*%
C shell superuser	*machine-name*#
Bourne shell and Korn shell	$
Bourne shell and Korn shell superuser	#

Related Documentation

Application	Title	Part Number
Reference	*Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual*	816-4767-10

Accessing Sun Documentation Online

A broad selection of Sun system documentation is located at:

`http://www.sun.com/products-n-solutions/hardware/docs`

A complete set of Solaris documentation and many other titles are located at:

`http://docs.sun.com`

At that site, you will also find information on how to order *printed* copies of this guide.

Sun Welcomes Your Comments

Sun is interested in improving its documentation and welcomes your comments and suggestions. You can email your comments to Sun at:

`docfeedback@sun.com`

Please include the part number (816-2077-12) of your document in the subject line of your email.

PART I Background and Definitions

This part of the *Sun Grid Engine 5.3 Administration and User's Guide* consists of a single chapter:

- **Chapter 1** – "Introduction to Sun Grid Engine 5.3" on page 1.

The brevity of the chapter may mislead the reader about its importance to users and administrators alike, as both will be well served by becoming familiar with the chapter's content. Included in the chapter are the following.

- A description of the primary role of Sun™ Grid Engine 5.3 software within complex computing environments
- A list of the major components of the product and definitions of the functions of each
- A glossary of terms that are important to know in a Sun Grid Engine 5.3 environment

CHAPTER 1

Introduction to Sun Grid Engine 5.3

This chapter provides background information about the Sun Grid Engine 5.3 system that is useful to users and administrators alike. In addition to a description of the product's role in managing what could otherwise be a chaotic world of clustered computers, this chapter includes the following topics.

- A brief description of grid computing
- An overview of `QMON`, the Sun Grid Engine 5.3 graphical user interface
- An explanation of each of the important components of the product
- A detailed list of client commands that are available to users and administrators
- A complete glossary of Sun Grid Engine 5.3 terminology

What Is Grid Computing?

Conceptually, a grid is quite simple. It is a collection of computing resources that perform tasks. It appears to users as a large system, providing a single point of access to powerful distributed resources. Users treat the grid as a single computational resource. Resource management software, such as Sun Grid Engine, accepts jobs submitted by users and schedules them for execution on appropriate systems in the grid based upon resource management policies. Users can literally submit thousands of jobs at a time without being concerned about where they run.

No two grids are alike; one size does not fit all situations. There are three key classes of grids, which scale from single systems to supercomputer-class compute farms that utilize thousands of processors. *Cluster grids*, consisting of a many computational resources working together to provide a single point of access to users in a single project or department, are what the Sun Grid Engine 5.3 system helps you to create and manage.

(Two other types of more complex grids—*campus grids* and g*lobal grids*—are created and managed by a related product from Sun, Sun Grid Engine, Enterprise Edition.)

Sun Grid Engine 5.3 software orchestrates the delivery of computational power based upon enterprise resource *policies* set by the organization's technical and management staff. The Sun Grid Engine system uses these policies to examine the available computational resources within the grid, gathers these resources, and then allocates and delivers them automatically in a way that optimizes usage across the grid.

Managing Workload by Managing Resources and Policies

Sun Grid Engine software provides the user with the means to submit computationally demanding tasks to the Sun Grid Engine system for transparent distribution of the associated workload. The user can submit batch jobs, interactive jobs, and parallel jobs to the Sun Grid Engine system.

The product also supports checkpointing programs. Checkpointing jobs migrate from workstation to workstation without user intervention on load demand.

For the administrator, the software provides comprehensive tools for monitoring and controlling Sun Grid Engine jobs.

How the System Operates

The Sun Grid Engine system accepts jobs—users' requests for computer resources—from the outside world, puts them in a holding area until they can be executed, sends them from the holding area to an execution device, manages them during execution, and logs the record of their execution when they are finished.

As an analogy, imagine a large "money-center" bank in one of the world's capitol cities.

Matching Resources to Requests

In the bank building's lobby are dozens upon dozens of customers, each with different requirements, who are waiting to be served. One customer merely wants to withdraw a small amount of money from his account. Arriving just after him is

another customer who has an appointment with one of the bank's investment specialists; she is seeking advice before undertaking a complicated venture. In front of both of them in the long line is another customer who intends to apply for a large loan—as do the eight customers in front of *her*.

Different customers and different intentions require different types and levels of the bank's resources. Perhaps, on this particular day, the bank has many employees who have sufficient time available to handle the one customer's simple withdrawal of money from his account. But on that day, only one or two loan officers are on hand to help the many loan applicants. On another day, the situation may be reversed.

The effect, of course, is that customers must wait for service—even though many of them could probably receive immediate service if only their requirements were immediately discerned and matched to available resources.

If the Sun Grid Engine system were the bank manager, it would organize the service differently.

- Upon entering the bank lobby, customers would be asked to declare their name, their affiliations (such as representing a company), and their requirements.
- The customers' time of arrival would be recorded.
- Based on the information that the customers provided in the lobby, those whose requirements match suitable and immediately available resources, those whose requirements have the highest priority, and those who have been waiting in the lobby for the longest time would be served.
- Of course, in a "Sun Grid Engine bank," one bank employee may be able to provide assistance to several customers at the same time. The Sun Grid Engine system would try to assign new customers to the least loaded and most suitable bank employee.

Jobs and Queues: The Sun Grid Engine World

In a Sun Grid Engine system, *jobs* correspond to bank customers, jobs wait in a computer holding area instead of a lobby, and *queues* located on computer servers take the place of bank employees, providing services for jobs. As in the case of bank customers in the analogy, the requirements of each of the jobs—which typically consist of available memory, execution speed, available software licenses, and similar needs—may be very different and only certain queues may be able to provide the corresponding service.

Corresponding to the analogy, Sun Grid Engine software arbitrates available resources and job requirements in the following fashion.

- A user who submits a job through the Sun Grid Engine system declares a requirement profile for the job. In addition, the identity of the user and his or her affiliation with *projects* or *user groups* is retrieved by the system. The time that the user submitted the job is also stored.
- The moment, literally, that a queue becomes available for execution of a new job, the Sun Grid Engine system determines suitable jobs for the queue and immediately dispatches the job with the highest priority or longest waiting time.
- Sun Grid Engine queues may allow concurrent execution of many jobs. The Sun Grid Engine system will try to start new jobs in the least loaded and suitable queue.

Sun Grid Engine 5.3 Components

Figure FIGURE 1-1 displays the most important Sun Grid Engine components and their interaction in the system. The following sections explain the functions of the components.

Hosts

Four types of hosts are fundamental to the Sun Grid Engine 5.3 system.

- Master
- Execution
- Administration
- Submit

Master Host

The master host is central for the overall cluster activity. It runs the master daemon, `sge_qmaster`, and the scheduler daemon, `sge_schedd`. Both daemons control all Sun Grid Engine components, such as queues and jobs, and maintain tables about the stttus of the components, about user access permissions, and the like.

By default, the master host is also an administration host and submit host. See the sections relating to those hosts.

Execution Host

Execution hosts are nodes that have permission to execute Sun Grid Engine jobs. Therefore, they are hosting Sun Grid Engine queues and run the Sun Grid Engine execution daemon, `sge_execd`.

Administration Host

Permission can be given to hosts to carry out any kind of administrative activity for the Sun Grid Engine system.

Submit Host

Submit hosts allow for submitting and controlling *batch jobs only*. In particular, a user who is logged into a submit host can submit jobs via `qsub`, can control the job status via `qstat`, and can use the Sun Grid Engine OSF/1 Motif graphical user's interface, `QMON`, which is described in the section, “`QMON`, the Sun Grid Engine Graphical User Interface” on page 9.

Note – A host may belong to more than one of the above described classes.

Daemons

Four daemons provide the functionality of the Sun Grid Engine 5.3 system.

`sge_qmaster` – the Master Daemon

The center of the cluster’s management and scheduling activities, `sge_qmaster` maintains tables about hosts, queues, jobs, system load, and user permissions. It receives scheduling decisions from `sge_schedd` and requests actions from `sge_execd` on the appropriate execution hosts.

`sge_schedd` – the Scheduler Daemon

The scheduling daemon maintains an up-to-date view of the cluster’s status with the help of `sge_qmaster`. It makes the following scheduling decision:

- Which jobs are dispatched to which queues

It then forwards these decisions to `sge_qmaster`, which initiates the required actions.

`sge_execd` – the Execution Daemon

The execution daemon is responsible for the queues on its host and for the execution of jobs in these queues. Periodically, it forwards information such as job status or load on its host to `sge_qmaster`.

`sge_commd` – the Communication Daemon

The communication daemon communicates over a well-known TCP port. It is used for all communication among Sun Grid Engine components.

Queues

A Sun Grid Engine queue is a container for a class of jobs allowed to execute on a particular host concurrently. A queue determines certain job attributes; for example, whether it may be migrated. Throughout their lifetimes, running jobs are associated with their queue. Association with a queue affects some of the things that can happen to a job. For example, if a queue is suspended, all the jobs associated with that queue are also suspended.

In the Sun Grid Engine system, there is no need to submit jobs directly to a queue. You only need to specify the requirement profile of the job (e.g., memory, operating system, available software, etc.) and Sun Grid Engine software will dispatch the job to a suitable queue on a low-loaded host automatically. If a job is submitted to a particular queue, the job will be bound to this queue and to its host, and thus Sun Grid Engine daemons will be unable to select a lower-loaded or better-suited device.

Client Commands

Sun Grid Engine's command line user interface is a set of ancillary programs (commands) that enable you to manage queues, submit and delete jobs, check job status, and suspend/enable queues and jobs. The Sun Grid Engine system makes use of the following set of ancillary programs.

- `qacct` – This command extracts arbitrary accounting information from the cluster logfile.
- `qalter` – This command changes the attributes of submitted, but pending, jobs.

- `qconf` – This command provides the user interface for cluster and queue configuration.
- `qdel` – This command provides the means for a user, operator, or manager to send signals to jobs or subsets thereof.
- `qhold` – This command holds back submitted jobs from execution.
- `qhost` – This command displays status information about Sun Grid Engine execution hosts.
- `qlogin` – This command initiates a `telnet` or similar login session with automatic selection of a low-loaded and suitable host.
- `qmake` – This command is a replacement for the standard UNIX `make` facility. It extends `make` by its ability to distribute independent `make` steps across a cluster of suitable machines.
- `qmod` – This command enables the owner to suspend or enable a queue (all currently active processes associated with this queue are also signaled).
- `qmon` – This command provides an X-windows Motif command interface and monitoring facility.
- `qresub` – This command creates new jobs by copying running or pending jobs.
- `qrls` – This command releases jobs from holds previously assigned to them; e.g., via `qhold` (see above).
- `qrsh` – This command can be used for various purposes, such as the following.
  - To provide remote execution of interactive applications via the Sun Grid Engine system—comparable to the standard UNIX facility, `rsh`
  - To allow for the submission of batch jobs which, upon execution, support terminal I/O (standard/error output and standard input) and terminal control
  - To provide a batch job submission client which remains active until the job has been finished
  - To allow for the Sun Grid Engine software-controlled remote execution of the tasks of parallel jobs
- `qselect` – This command prints a list of queue names corresponding to specified selection criteria. The output of `qselect` is usually fed into other Sun Grid Engine commands to apply actions on a selected set of queues.
- `qsh` – This command opens an interactive shell (in an `xterm`) on a low-loaded host. Any kind of interactive jobs can be run in this shell.
- `qstat` – This command provides a status listing of all jobs and queues associated with the cluster.
- `qsub` – This command is the user interface for submitting a job to the Sun Grid Engine system.

- `qtcsh` – This command is a fully compatible replacement for the widely known and used Unix C-Shell (`csh`) derivative, `tcsh`. It provides a command shell with the extension of transparently distributing execution of designated applications to suitable and lightly loaded hosts via Sun Grid Engine software.

All programs communicate with `sge_qmaster` via `sge_commd`. This is reflected in the schematic view of the component interaction in the Sun Grid Engine system, depicted in FIGURE 1-1.

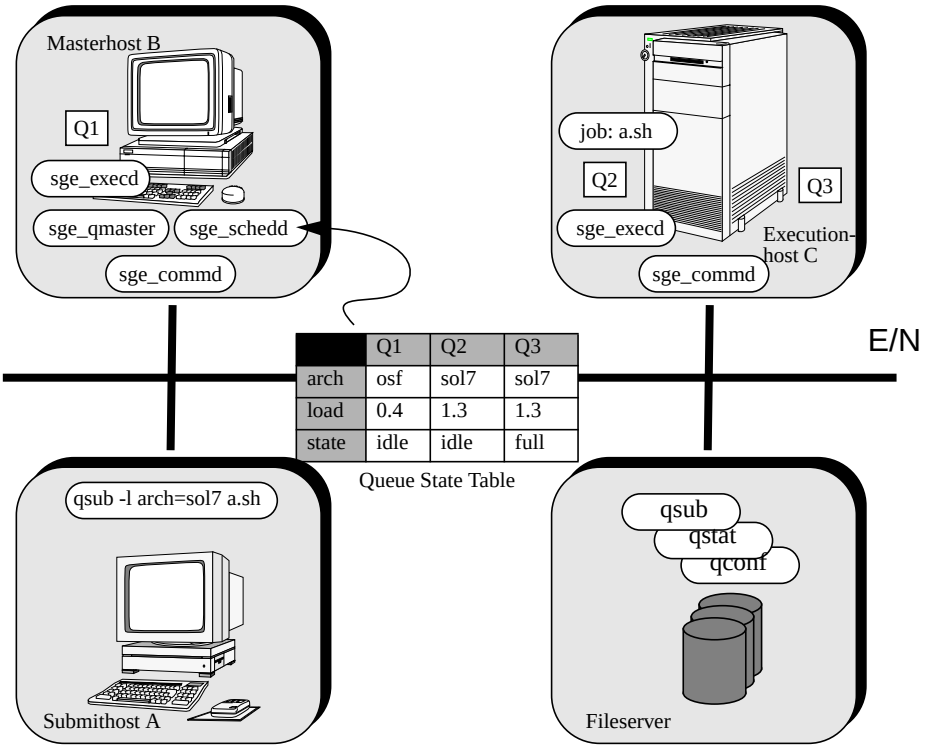


FIGURE 1-1 Component Interaction in the Sun Grid Engine System

QMON, the Sun Grid Engine Graphical User Interface

Using QMON, the graphical user interface (GUI) tool, you can accomplish most—if not all—Sun Grid Engine 5.3 tasks. FIGURE 1-2 shows the QMON Main menu, which is often the starting point for both user and administrator functions. Each icon on the Main menu is a GUI button that you press to initiate a variety of tasks. The name of each button, which appears as text on the screen when you pass the mouse pointer over it, is also descriptive of its function.

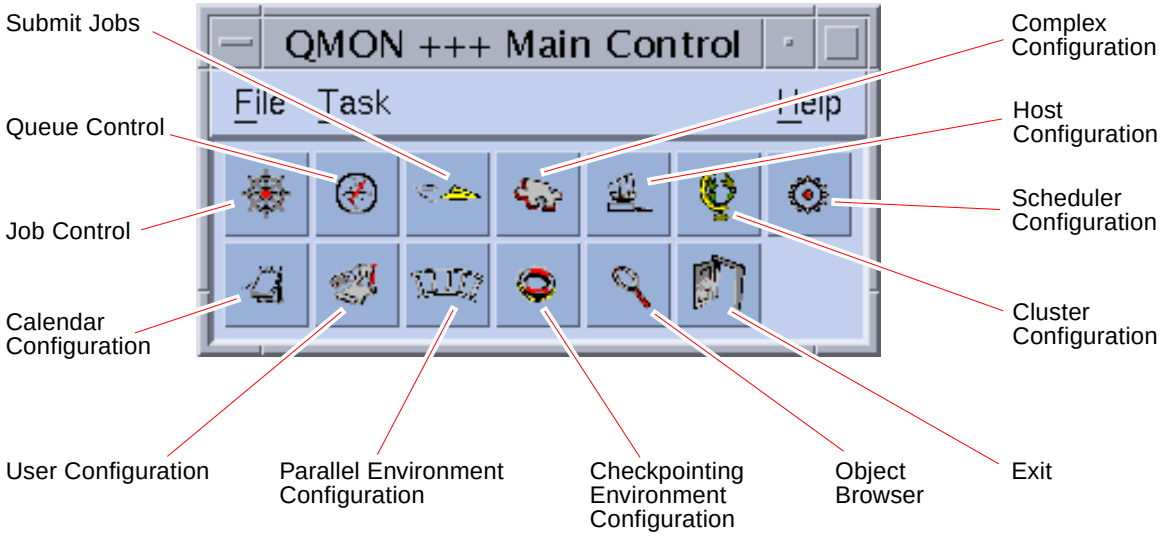


FIGURE 1-2 QMON Main Menu, Defined

Customizing QMON

The look and feel of qmon is largely defined by a specifically designed resource file. Reasonable defaults are compiled in and a sample resource file is available under <sge_*root*>/qmon/Qmon.

The cluster administration may install site specific defaults in standard locations such as `/usr/lib/X11/app-defaults/Qmon`, by including `qmon` specific resource definitions into the standard `.Xdefaults` or `.Xresources` files or by putting a site specific `Qmon` file to a location referenced by standard search paths such as `XAPPLRESDIR`. Ask your administrator if any of the above is relevant in your case,

In addition, the user can configure personal preferences by either copying and modifying the `Qmon` file into the home directory (or to another location pointed to by the private `XAPPLRESDIR` search path) or by including the necessary resource definitions into the user's private `.Xdefaults` or `.Xresources` files. A private `Qmon` resource file may also by installed via the `xrdb` command during operation or at start-up of the X11 environment, e.g. in a `.xinitrc` resource file.

Refer to the comment lines in the sample `Qmon` file for detailed information on the possible customizations.

Another means of customizing `qmon` has been explained for the Job Control and Queue Control customization dialogue boxes shown in FIGURE 5-3 and in FIGURE 5-13. In both dialogue boxes, you can use the Save button to store the filtering and display definitions configured with the customization dialogue boxes to the file, `.qmon_preferences`, in the user's home directory. Upon being restarted, `qmon` reads this file and reactivates the previously defined behavior.

Glossary of Sun Grid Engine Terms

The glossary provides a short overview on frequently used terms in the context of Sun Grid Engine and resource management in general. Many of the terms have not been used so far, but will appear in other parts of the Sun Grid Engine documentation.

access list A list of users and UNIX groups who are permitted, or denied, access to a resource such as a queue or a certain host. Users and groups may belong to multiple access lists and the same access lists can be used in various contexts.

Array job A job consisting of a range of independent identical tasks. Each task is very similar to a separate job. Job array tasks only differ by a unique task identifier (an integer number).

cell A separate Sun Grid Engine cluster with a separate configuration and master machine. Cells can be used to loosely couple separate administrative units.

checkpointing A procedure which saves the execution status of a job into a so called *checkpoint* thereby allowing for the job to be aborted and resumed later without loss of information and already completed work. The process is called *migration*, if the checkpoint is moved to another host before execution resumes.

checkpointing environment A Sun Grid Engine configuration entity, which defines events, interfaces and actions being associated with a certain method of checkpointing.

cluster A collection of machines, called hosts, on which Sun Grid Engine functions occur.

complex A set of attributes that can be associated with a queue, a host, or the entire cluster.

group A UNIX group.

hard resource requirements The resources which must be allocated before a job may be started. Contrasts with *soft resource requirements*.

host A machine on which Sun Grid Engine functions occur.

job A batch job is a UNIX shell script that can be executed without user intervention and does not require access to a terminal.

An interactive job is a session started with the Sun Grid Engine commands `qsh` or `qlogin` that will open an *xterm* window for user interaction or provide the equivalent of a remote login session, respectively.

job class A set of jobs that are equivalent in some sense and treated similarly. In Sun Grid Engine a job class is defined by the identical requirements of the corresponding jobs and the characteristics of the queues being suitable for those jobs.

manager A user who can manipulate all aspects of Sun Grid Engine. The superusers of the master host and of any other machine being declared as an administrative host have manager privileges. Manager privileges can be assigned to non-root user accounts as well.

migration The process of moving a checkpoint from one host to another before execution of the job resumes.

operator Users who can perform the same commands as managers except that they cannot change the configuration but rather are supposed to maintain operation.

owner Users who may suspend/unsuspend and disable/enable the queues they own. Typically users are owners of the queues that reside on their workstations.

parallel environment A Sun Grid Engine configuration entity, which defines the necessary interfaces for Sun Grid Engine to correctly handle parallel jobs.

parallel job A job which consists of more than one closely correlated task. Tasks may be distributed across multiple hosts. Parallel jobs usually use communication tools such as shared memory or message passing (MPI, PVM) to synchronize and correlate tasks.

policy A set of rules and configurations which the Sun Grid Engine administrator can use define the behavior of Sun Grid Engine. Policies will be implemented automatically by Sun Grid Engine.

priority The relative level of importance of a Sun Grid Engine job compared to others.

queue A container for a certain class and number of jobs being allowed to execute on a Sun Grid Engine execution host concurrently.

resource A computational device consumed or occupied by running jobs. Typical examples are memory, CPU, I/O bandwidth, file space, software licenses, etc.

soft resource requirements Resources which a job needs but which do not have to be allocated before a job may be started. Allocated to a job on an as available basis. Contrast with *hard resource requirements*.

suspension The process of holding a running job but keeping it on the execution machine (in contrast to checkpointing, where the job is aborted). A suspended job still consumes some resources, such as swap memory or file space.

user May submit jobs to and execute jobs with Sun Grid Engine if he or she has a valid login on at least one submit host and an execution host.

userset An access list (see above).

PART II Getting Started

This part of the *Sun Grid Engine 5.3 Administration and User's Guide* consists of a single chapter.

- **Chapter 2** – "Installation" on page 15

Included in the chapter are instructions for a first-time installation of the Sun Grid Engine 5.3 product, as well as instructions for upgrading preceding versions of the product to the new release.

CHAPTER 2

Installation

This chapter describes and provides detailed instructions for three installation tasks:

- "Quick-start" installation—not recommended for all sites (see "Overview of Quick-Start Installation" on page 15
- Full, fresh installation of the Sun Grid Engine 5.3 software (see "Overview of Full Installation" on page 22)
- Secure installation with special encryption features (see "How To Install and Set Up a CSP-Secured System" on page 36)

Note – The instructions in this chapter presume that you are installing the software on a computer running the Solaris™ Operating Environment. Any difference in functionality created by other operating system architecture that Sun Grid Engine runs on is documented in files starting with the string, `arc_depend_` in the *<sge_root>*`/doc` directory. The remainder of the file name indicates the operating system architectures to which the comments in the files apply.

Overview of Quick-Start Installation

Note – This section describes the conditions that *must* prevail at your site for the *quick-start* installation procedure to be applicable. If your environment does not include *all* of the prerequisites outlined, then you *cannot* use the quick-start installation procedure. In this case, proceed to the section, "Overview of Full Installation" on page 22 for detailed information on how to install Sun Grid Engine 5.3 software under more restricted conditions.

Prerequisites for Quick-Start Installation

Note – These instructions are for a *fresh* Sun Grid Engine 5.3 installation only. For instructions on how to upgrade an existing installation of an older version of the Sun Grid Engine product, see the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Release Notes.*

This section describes several conditions that must exist or be created at your site before you undertake the quick-start installation procedure.

- Installation accounts
- The installation directory
- Communication port number

Installation Accounts

For the quick-start installation procedure, an *Administrator* account must exist or you must create one according to the following guidelines.

- The Administrator can be an existing administrative login or a new login, such as `sgeadmin`.

  This account will own all of the files in the Sun Grid Engine installation and spooling directories and it can be used to configure and administer the cluster once it is installed.

- This user should *not* be *root*.

  However, if you intend to use *root* for file ownership, the user *root* must have full write permissions an all hosts in the directory where the Sun Grid Engine program is installed. Usually a shared (NFS) filesystem is not exported for the user *root* to allow write permission.

- This account must exist *prior* to installation.

The Installation Directory

To prepare for use of the Administrator account, you should create the installation directory, preferably on a network-wide share file system.

▼ How To Create the Installation Directory

- **Use the following sequence of commands.**

```
% mkdir -p <install_dir>
% chown <adminuser> <install_dir>
% chmod 755 <install_dir>
```

Note – The directory created by this or a similar procedure is referred to as the Sun Grid Engine *root directory* for the remainder of this manual.

Communication Port Number

The Sun Grid Engine system uses a TCP port for communication. *All* hosts in the cluster must use the same port number. You can place he port number in any of in several locations. Two examples follow.

- NIS services or NIS+ database

  Add the following to the services database.

```
% sge_commd  535/tcp
```

- `/etc/services` on each machine

  If NIS is not running at your site, then you can add the above services to the `/etc/services` file on each machine.

It is best to use a privileged port below 600 to avoid conflicts with applications that bind ports near or below 1024, or ports higher than 1024, dynamically.

Reading the Distribution Media

The Sun Grid Engine 5.3 software is distributed as an archive file through Internet download. The distribution consists of a tape archive (`tar`) directly written on the medium.

▼ How To Unpack the Sun Grid Engine 5.3 Distribution

1. **Log in as the account you selected for the installation (see section "Prerequisites for Quick-Start Installation" on page 16) to the host from which you plan to read in the Sun Grid Engine distribution media.**
2. **Change your working directory to the Sun Grid Engine root directory.**
3. **Use the following command to read in the distribution media.**

```
% cd sge_root_dir
% tar -xvpf distribution_source
```

In the commands above, *sge_root_dir* is the path name of the Sun Grid Engine root directory and *distribution_source* is the name of the tape archive file on hard disk.

Installing a Default Sun Grid Engine System for Your Cluster

A default Sun Grid Engine system consists of a *master host* and an arbitrary number of *execution hosts*. The master host controls the overall cluster activity while the execution hosts control the execution of the jobs being assigned to them by the master host. A single host may concurrently act as a master host and as an execution host.

Note – Install the master host first and then conclude with the installation of the execution hosts in arbitrary sequence.

▼ How To Install the Master Host

1. **Select a machine as the master host.**

   Follow these guidelines in making your selection of the master host machine.
   - The selected system should not be overloaded with other tasks.
   - The master host should provide for enough available main memory to run the necessary Sun Grid Engine daemons.

     The required amount strictly depends on the size of your cluster and the number of jobs in the system to be expected. For clusters up to a few dozen hosts and in the order of 100 jobs, 10 megabytes of free memory should be sufficient.

     For very large clusters—in the order of 1,000 or more hosts and several 10,000 jobs—you may well need 1 gigabyte of memory.

2. **Log in to the selected machine.**

   For an installation featuring all capabilities, you must install using the *root* account (files still may be owned by the Administrator account created in the section, “Prerequisites for Quick-Start Installation” on page 16).

   For a test installation, you can also install as the Administrator user, but then only the Administrator will be able to run jobs and the Sun Grid Engine system will have restricted capabilities with respect to monitoring system load and system control.

3. **Change directory (`cd`) to the Sun Grid Engine root directory.**

4. **Execute the master host installation procedure with the following command.**

   ```
   % ./install_qmaster
   ```

   If the event of errors, the installation procedure prints a description of the error condition.

5. **Respond to the installation script with the names of hosts that you want to install.**

   The installation script asks you to provide a list of hosts you initially want to install. List all such hosts, as these hosts will be added as *submit* hosts and *administrative* hosts.

   The installation of the *execution* hosts (see the next section) requires that all hosts are administrative hosts. If you plan to install the Sun Grid Engine program on many hosts, the installation script will give you the opportunity to provide the path to a file that contains the list of all host names, with one host per line.

6. **Respond to all remaining installation script prompts.**

   The installation procedure requires some additional information. Most questions will provide useful defaults, which you can confirm by pressing Return.

▼ How To Install the Execution Hosts

1. **Select which account to use for the installation, according to the following guidelines.**

   - As with the master host installation, you should install the execution hosts by using the *root* account to have access to all Sun Grid Engine facilities.

     Installing as *root* still retains file ownership by the Administrator account created in the section, “Prerequisites for Quick-Start Installation” on page 16.

- Installation by way of the Administrator account is only useful for test purposes and prohibits users other than Administrator to run jobs. It does not allow the Sun Grid Engine program to provide full system monitoring and control capabilities.

2. **Log in as the account selected for the installation to one of the execution hosts specified during the master host installation procedure.**

3. **Change directory (`cd`) to the Sun Grid Engine root directory.**

4. **Enter the following command to initiate the execution host installation procedure:.**

```
% ./install_execd
```

   Any errors are indicated by the installation procedure.

5. **Respond to installation script prompts.**

   The installation procedure will ask you whether default queues should be configured for your host. The queues defined in this case will have the characteristics described in the section, “The Default System Configuration” on page 20.

6. **As the installation script notifies you of successful completion of the execution host installation procedure, proceed likewise with each execution host that you named during the master host installation.**

   As soon as you are through with the list, your default Sun Grid Engine system is configured on your cluster and is ready to be used. To get some practice using the program, see the section, “How To Run a Simple Job from the Command Line” on page 70.

   The following section is an overview of the default configuration that has been installed.

The Default System Configuration

Note – The following is a description of the Sun Grid Engine system as configured in your environment by the quick-start installation procedure. It is a minimal setup for testing purposes and may be changed or extended later on at any time.

After completion of the master host and execution host installations, the following basic Sun Grid Engine system has been configured on your cluster.

- Master Host

  The host on which you ran the master host installation procedure is configured to be the master host of your cluster. No shadow master hosts are configured to take over the master host‘s tasks in case of failure.

- Execution Hosts

  During the master host installation, you are asked for a list of machines on which you want to install the Sun Grid Engine execution agent. During installation of these execution hosts, you can allow the installation procedure to create *queues* automatically on these hosts. A queue describes the profile (a list of attributes and requirements) of the jobs that can be run on that particular host. The queues being configured for the execution hosts by default have the following important characteristics.

  - Queue name: <*unqualifedhostname*>.*q*
  - Slots (concurrent jobs): <*number_of_processors*>
  - The queues provide unlimited system resources (such as memory, CPU-time etc.) to the jobs.
  - The queues do not enforce any access restrictions for particular users or user groups. Any user with a valid account can run jobs in the queues.
  - A load threshold of 1.75 per CPU will be configured (i.e., 1.75 processes attempting on average to get access to each CPU).

Note – The queue configurations, as any other Sun Grid Engine configuration, can be changed on-the-fly at any later stage while the system is in operation.

Note – If you invoked the execution host installation procedure on the master host also, the master host acts both as master and as execution host.

- Administrative Accounts and Hosts

  The master host and all execution hosts are configured to be allowed to execute administrative Sun Grid Engine commands. The only users that are allowed to administer Sun Grid Engine are the user *root* and the Administrator account described in the section, “Prerequisites for Quick-Start Installation” on page 16. If an unprivileged user installs the Sun Grid Engine program, he or she is added to the list of Sun Grid Engine administrators.

- Submit Accounts and Hosts

  If you installed the program under the *root* account, any user with a valid account can submit and control Sun Grid Engine jobs. The user under which you installed Sun Grid Engine software will be the only user to whom access is permitted

otherwise (see “Prerequisites for Quick-Start Installation” on page 16). The tasks of submitting jobs, controlling the Sun Grid Engine system activity, or deleting jobs can be executed from either the master host or from any execution host.

- Daemons

  The following daemons are started up during system installation on the different hosts or may be invoked during normal system operation respectively.

  - `sge_qmaster` runs on the master host only. It is the central cluster activity control daemon.
  - `sge_schedd` is also invoked on the master host only. This daemon is responsible for distributing the workload in the Sun Grid Engine cluster.
  - `sge_execd` is responsible for executing the jobs on an execution host and, therefore, is running on all execution hosts.
  - One instance of `sge_shepherd` is run for each job being actually executed on a host. `sge_shepherd` controls the jobs process hierarchy and collects accounting data after the job has finished.
  - `sge_commd` runs on each execution host and on the master host. The network of all `sge_commd`s forms the network communication backbone of the Sun Grid Engine cluster.

Overview of Full Installation

Note – These instructions are for a *fresh* Sun Grid Engine 5.3 installation only. For instructions on how to upgrade an existing installation of an older version of the Sun Grid Engine product, see the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Release Notes*.

Full installation consists of the following broad tasks.

- Planning the Sun Grid Engine configuration and environment
- Reading the Sun Grid Engine distribution files from an external medium onto a workstation
- Running an installation script on the master host and every execution host in the Sun Grid Engine system
- Registering information about administrative and submit hosts
- Verifying the installation

Installation should be done by someone familiar with the SolarisOperating Environment. The entire process is done in three phases.

Phase 1 - Planning

Note – If you want to install the system with enhanced security, see the section, “Installing with Increased Security” on page 35 before you continue installation.

The planning phase of installation consists of the following tasks.

- Deciding whether your Sun Grid Engine environment will be a single cluster or a collection of sub-clusters called *cells*
- Selecting the machines that will be Sun Grid Engine hosts. Determine what kind(s) of host(s) each machine will be— master host, shadow master host, administration host, submit host, execution host, or a combination
- Making sure that all Sun Grid Engine users have the same user ids on all submit and execution hosts
- Deciding what the Sun Grid Engine directory organization will be—for example, organizing directories as a complete tree on each workstation, directories cross-mounted, or a partial directory tree on some workstations—and where each Sun Grid Engine root directory will be located
- Deciding on the site’s queue structure
- Deciding whether network services will be defined as an NIS file or local to each workstation in `/etc/services`
- Completing the installation worksheet (refer to 1., “Before beginning installation, write down your installation plan in a table similar to the one below.” on page 31) to use in subsequent installation steps

Phase 2 - Installing the Software

The installation phase consists of the following tasks.

- Creating the installation directory and load the distribution files into it
- Installing the master host
- Installing all execution hosts
- Registering all administrative hosts
- Registering all submit hosts

Phase 3 - Verifying the Installation

The verification phase consists of the following tasks.

- Checking that the daemons are running on the master host
- Checking that the daemons are running on all execution hosts

- Checking that Sun Grid Engine executes simple commands
- Submitting test jobs

Planning the Installation

Before you begin installing the Sun Grid Engine 5.3 software, you must carefully plan how you will do it to achieve the results that perfectly fit your environment. This section will help you to make vital decisions that will affect the rest of the procedure.

Prerequisite Tasks

The following sections describe the information you will need to install a production Sun Grid Engine system.

The Installation Directory *<sge_root>*

Prepare a directory to read in the contents of the Sun Grid Engine distribution media. This directory will be called the Sun Grid Engine *root directory* and later on, while the Sun Grid Engine system is in operation, will be used to store the current cluster configuration and all further data that needs to be spooled to disk.

Use a path name for the directory that is a correct reference on all hosts. For example, if the file system is mounted using automounter, set *<sge_root>* to `/usr/SGE`, not `/tmp_mnt/usr/SGE`. (Throughout this document, the *<sge_root>* environment variable is used when referencing the installation directory.)

<sge_root> is the top level of the Sun Grid Engine directory tree. Each Sun Grid Engine component in a cell (see the section, “Cells” on page 29) needs read access to *<sge_root>*`/`*<cell>*`/common` on startup. See the section, “File Access Permissions” on page 27, for a description of required permissions.

For ease of installation and administration, this directory should be readable on all hosts you intend to execute the Sun Grid Engine installation procedure on. You may, for example, select a directory available via a network file system (such as NFS). If you choose to select filesystems local to the hosts you will have to copy the installation directory to each host before you start the installation procedure for the particular machine.

Spool Directories Under the Root Directory

- On the Sun Grid Engine master host, spool directories are maintained under <*sge_root*>/<*cell*>/spool/qmaster and <*sge_root*>/<*cell*>/spool/schedd.
- On each execution host, a spool directory called <*sge_root*>/<*cell*>/spool/<exec_host> is maintained.

You do not need to export these directories to other machines. However, exporting the entire <*sge_root*> tree and making it write-accessible for the master and all executable hosts will enhance ease of administration.

Directory Organization

Decide what the Sun Grid Engine directory organization will be (for example, a complete tree on each workstation, directories cross-mounted, a partial directory tree on some workstations) and where each Sun Grid Engine root directory, <*sge_root*>, will be located.

Note – Since a change of the installation directory and/or the spool directories basically requires a new installation of the system (although all important information from the previous installation can be preserved), you should use extra care to select a suitable installation directory upfront.

By default, the Sun Grid Engine installation procedure will install the Sun Grid Engine system, manuals, spool areas and the configuration files in a directory hierarchy (see FIGURE 2-1, “Sample Directory Hierarchy” on page 26) under the installation directory. If you accept this default behavior, you should install/select a directory which allows the access permissions described in “File Access Permissions” on page 27.

You can select the spool areas to place in other locations during the primary installation (see Chapter 6, “Host and Cluster Configuration” on page 167 for instructions).

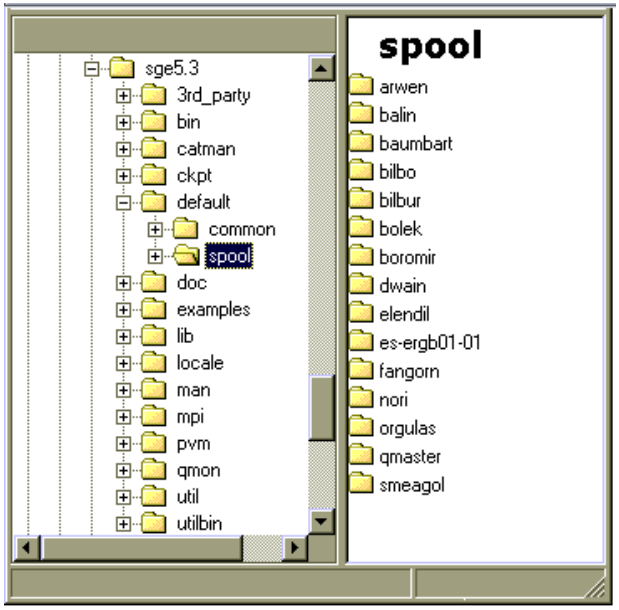


FIGURE 2-1 Sample Directory Hierarchy

Disk Space Requirements

The Sun Grid Engine directory tree has certain fixed disk space requirements, as follows.

- 40 MB for the installation kit (including documentation) without any binaries
- Between 10 and 15 MB for each set of binaries (except for the architecture Cray, where the binaries consume approximately 35 MB)

The ideal disk space for Sun Grid Engine log files follows.

- 30-200 MB for the master host spool directories, depending on the size of the cluster
- 10-20 MB for each execution host

Note – The spool directories of the master host and the execution hosts are configurable and do not have to reside under <*sge_root*> (where they are located by default). Changing the location of the spool directories should be done after the primary installation (see Chapter 6, “Host and Cluster Configuration” on page 141 for instructions).

Installation Accounts

You can install Sun Grid Engine either under the root account or under an unprivileged (e.g., your own) account. If you install under an unprivileged account, this installation will only allow for that particular user to run Sun Grid Engine jobs. Access will be denied to all other accounts. Installing under the root account resolves this restriction; however, root permission is required for the complete installation procedure.

File Access Permissions

If you install as root, you may have a problem configuring root read / write access for all hosts on a shared file system, and thus you may have problems putting <*sge_root*> onto a network-wide file system. You can force Sun Grid Engine software to run the entire file handling of all Sun Grid Engine components through a non-root administrative user account (called `sgeadmin`, for example). Thus you only need read / write access to the shared root file system for this particular user. The Sun Grid Engine installation procedure will ask whether you want file handling under an administrative user account. If you answer **`Yes`** and provide a valid *user name*, file handling will be performed via this user name. Otherwise, the user name under which you run the installation procedure will be used.

You have to make sure in all cases that the account used for file handling has read / write access on all hosts to the Sun Grid Engine root directory. Also, the installation procedure assumes that the host from which you will read in the Sun Grid Engine distribution media can access this directory.

Network Services

Determine whether your site’s network services are defined as a NIS file or local to each workstation in `/etc/services`. If your site uses NIS, find out the NIS server host so that you can add entries to the services NIS map.

The Sun Grid Engine service is sge_commd. To add the service to your NIS map, choose a reserved, unused port number—one that is below 1024. The following is an example of an `sge_commd` entry.

```
sge_commd 536/tcp
```

Master Host

This is the host from which Sun Grid Engine is controlled. It runs the master daemon, `sge_qmaster`. For very large clusters—those that include many hundreds or thousands of hosts and tens of thousands of jobs in the system at any time—1 gigabyte or more unused main memory may be required, and two CPUs may be beneficial.

- It must be a stable platform.
- It must *not* be excessively busy with other processing.
- It must have at least 20 Mbytes of unused main memory to run the Sun Grid Engine daemons.
- *Optionally*, it should have the Sun Grid Engine directory, *<sge_root>*, local to it to cut down on network traffic.

Shadow Master Hosts

These hosts back up the functionality of `sge_qmaster` in case the master host or the master daemon fails. To be a shadow master host, a machine must have the following characteristics.

- It must run `sge_shadowd`.
- It must share `sge_qmaster`'s status, job, and queue configuration information that is logged to disk. In particular, the shadow master hosts need read/write root or admin user access to the `sge_qmaster`'s spool directory and to the *<sge_root>*`/`*<cell>*`/common` directory.
- The *<sge_root>*`/`*<cell>*`/common/shadow_masters` file must contain a line defining the host as a shadow master host.

The shadow master host facility is activated for a host as soon as these conditions are met. So you do not need to restart Sun Grid Engine daemons to make a host into a shadow host

Execution Hosts

These hosts run the jobs that are submitted to Sun Grid Engine. You will run an installation script on each execution host.

Administrative Hosts

Sun Grid Engine operators and managers perform administrative tasks such as reconfiguring queues or adding Sun Grid Engine users from these hosts. The master host installation script automatically makes the master host an administrative host.

Submit Hosts

Sun Grid Engine jobs may be submitted and controlled from submit hosts. The master host installation script automatically makes the master host a submit host.

Cells

You may set up Sun Grid Engine as a single cluster or a collection of loosely coupled clusters called *cells*. The `SGE_CELL` environment variable indicates the cluster being referenced. When Sun Grid Engine is installed as a single cluster, `SGE_CELL` is not set and the value `default` is assumed for the cell value.

User Names

In order for Sun Grid Engine to verify that users submitting jobs have permission to submit them and to use the execution hosts they need, users' names must be identical on the submit and execution hosts involved. This requirement may necessitate changing user ids on some machines.

Note – The user ids on the master host are not relevant for permission checking and do not have to match or even do not have to exist.

Queues

Plan the queue structure that meets your site's needs. This means determining what queues should be placed on which execution hosts, whether you need queues for sequential, interactive, parallel and other job types, how many job slots are needed in each queue, and other queue configuration decisions.

It is also possible for the Sun Grid Engine administrator to let the installation procedure create a default queue structure, which is suitable for getting acquainted with the system and as starting point for later tuning.

Note – Despite the directory Sun Grid Engine is installed to, all settings created by the Sun Grid Engine installation procedure can be changed during operation of the system on the fly.

In case you are already familiar with Sun Grid Engine or you previously have decided on the queue structure you want to impose on your cluster, you should not allow the installation procedure to install a default queue structure for you. But instead, you should prepare a document specifying that queue structure and you should proceed to Chapter 7, “Configuring Queues and Queue Calendars” on page 163, directly after completing the installation process.

▼ How To Plan the Installation

1. **Before beginning installation, write down your installation plan in a table similar to the one below.**

Parameter	Value
<sge_root>	
admin user	
admin group	
`sge_commd` port number	
Master host	
Shadow master hosts	
Execution hosts	
Administrative hosts	
Submit hosts	

1. **Ensure that the file system(s) and directories that will contain the Sun Grid Engine distribution and the spool and configuration files are set up properly by setting the access permissions as defined above.**

▼ How To Read the Distribution Media

Sun Grid Engine software is distributed as archive file through Internet download. The Web distribution is also provided in `tar` file format eventually compressed with `compress` (extension `.Z`) or with `gzip` (extension `.gz`). You must uncompress the file (use `uncompress` or `gunzip`) before proceeding with the following steps.

1. **Provide access to the distribution media and log in to a system—preferably a system that has direct connection to a file server.**
2. **Create the installation directory as described in the section, "The Installation Directory <sge_root>" on page 24 to read in the Sun Grid Engine installation kit, making sure that the access permissions for the installation directory are set properly.**
3. **Execute the following procedure from the command prompt.**

```
% cd install_dir
% tar -xvpf distribution_source
```

where *install_dir* is the path name of the installation directory and *distribution_source* is the name of the uncompressed tape archive file on hard disk. This will read in the Sun Grid Engine installation kit.

▼ How To Install the Master Host

1. **Log in to the master host as `root`.**
2. **Depending on whether the directory where the installation kit resides is visible from the master host, do one of the following.**
   a. **If the directory where the installation kit resides *is* visible from the master host, change directories (`cd`) to the installation directory and then proceed to Step 3.**
   b. **If the directory is *not* visible and cannot be made visible, do the following.**
      i. **Create a local installation directory on the master host.**
      ii. **Copy the installation kit to the local installation directory via the network (e.g., by using `ftp` or `rcp`).**
      iii. **Change directories (`cd`) to the local installation directory.**
3. **Execute the following instruction.**

Note – You must add the `-csp` flag to the following command if you are performing an installation via the Certificate Security Protocol method (see "How To Install and Set Up a CSP-Secured System" on page 36).

```
% ./install_qmaster
```

This will initiate the master installation procedure. You will be asked several questions and may be required to execute some administrative actions. The questions and the action items are self-explanatory.

Note – It is convenient to have a second terminal session active to execute administrative tasks.

The master installation procedure creates the appropriate directory hierarchy required by `sge_qmaster` and `sge_schedd`. The procedure starts up the Sun Grid Engine components `sge_commd`, `sge_qmaster` and `sge_schedd` on the master host. The master host is also registered as host with administrative and submit permission.

If you believe that something went wrong, you can abort and repeat the installation procedure at any time.

▼ How To Install Execution Hosts

1. **Log in as `root` to the execution host.**
2. **As for the master installation, either copy the installation kit to a local installation directory or use a network installation directory.**
3. **Change directories (`cd`) to the installation directory and execute the following command.**

Note – You must add the `-csp` flag to the following command if you are performing an installation via the Certificate Security Protocol method (see "How To Install and Set Up a CSP-Secured System" on page 36).

```
% ./install_execd
```

This will initiate the execution host installation procedure. The behavior and handling of the execution host installation procedure is very similar to the one for the master host.

4. **Respond to the prompts from the installation script.**

Note – You may use the master host also for execution of jobs. You just need to carry out the execution host installation for the master machine.

Note – If you use a very slow machine as master host, or if your cluster is considerably large, you should use the master machine for the master task only.

The execution host installation procedure creates the appropriate directory hierarchy required by `sge_execd`. The procedure starts up the Sun Grid Engine components `sge_commd` and `sge_execd` on the execution host.

▼ How To Install Administration and Submit Hosts

The master host is implicitly allowed to execute administrative tasks and to submit, monitor, and delete jobs. It does not require any kind of additional installation as administration or submit host. As opposed to this, *pure* administration and submit hosts do require registration.

- **From an administrative host (e.g., the master host) and through an administrative account (e.g., the super user account), enter the following commands.**

```
% qconf -ah admin_host_name[,...]
% qconf -as submit_host_name[,...]
```

Refer to the section, “About Daemons and Hosts” on page 143 for more details and other means to configure the different host types.

Installing with Increased Security

You can set up your system more securely by using the following instructions. These instructions will help you set up your system with *Certificate Security Protocol* (CSP)-based encryption.

Both Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 products can take advantage of this secure setup, and these instructions apply to both products. For the sake of brevity, these instructions cite only the Sun Grid Engine product.

Instead of transfering messages in clear text, the messages within this more secure system are encrypted with a secret key. The secret key is exchanged via a public/private key protocol. The user presents his or her certificate through the Sun Grid Engine system to prove identity, and receives the certificate from the Sun Grid Engine system to be sure he or she is communicating to the correct system. After this initial announcement phase, the communication is transparently continued in encrypted form. The session is valid only for a certain period, after which the session must be reannounced.

Additional Setup Required

The steps required to set up the Certificate Security Protocol enhanced version of the Sun Grid Engine system are very similar to the standard setup. You generally follow the instructions in the sections, “How To Plan the Installation” on page 31, “How To Read the Distribution Media” on page 31, “How To Install the Master Host” on page 32, “How To Install Execution Hosts” on page 33, and “How To Install Administration and Submit Hosts” on page 34.

However, the following additional tasks are necessary.

- Generation of the Certificate Authority (CA) system keys and certificates on the master host

  This is done by calling the installation script with the `-csp` flag.

- Distribution of the system keys and certificates to the execution and submit hosts

  It is the task of the system administrator to do it in a secure way; that is, the keys must be transmitted to the execution host and submit hosts in a secure manner, such as via `ssh` .

- Generation of user keys and certificates

This can be done automatically by the system administrator after master installation.

- Admittance of new users by the system administrator

▼ How To Install and Set Up a CSP-Secured System

1. **Install the Sun Grid Engine system as outlined in the sections, “Overview of Full Installation” on page 22, “Planning the Installation” on page 24, “How To Install the Master Host” on page 32, “How To Install Execution Hosts” on page 33, and “How To Install Administration and Submit Hosts” on page 34—with the following exception: use the additional flag, `-csp`, when invoking the various installation scripts.**

   For example, where the basic installation instruction for installing the master host tells you to call the script by entering `./install_qmaster`, you would amend that instruction by adding the `-csp` flag. Therefore, to install a CSP-secured system, you would change the master host installation procedure by entering the following.

   ```
   % ./install_qmaster -csp
   ```

2. **Respond to the prompts from the installation script.**

   To generate the CSP certificates and keys, you must supply the following information.

   - Two-letter country code—for example, US for the United States
   - State
   - Location—such as a city
   - Organization
   - Organizational unit
   - CA email address

   As the installation proceeds, the Certificate Authority is created. A Sun Grid Engine specific CA is created at the master host. The directories that contain security relevant information are as follows.

   - Under `$SGE_ROOT/`*{default |$SGE_CELL}*`/common/sgeCA`, the publicly accessible CA and daemon certificate are stored.
   - Under `/var/sgeCA/`*{sge_service | port$COMM_PORT}*`/`*{default | $SGE_CELL}*`/private`, the corresponding private keys are stored.
   - Under `/var/sgeCA/`*{sge_service | port$COMM_PORT}*`/`*{default | $SGE_CELL}*`/userkeys/$USER`, user keys and certificates are stored.

During this process, the script output will appear similar to the example in CODE EXAMPLE 2-1.

CODE EXAMPLE 2-1 CSP Installation Script—Directory Creation

```
Initializing Certificate Authority (CA) for OpenSSL security framework
------------------------------------------------------------------------

Creating /scratch2/eddy/sge_sec/default/common/sgeCA
Creating /var/sgeCA/port6789/default
Creating /scratch2/eddy/sge_sec/default/common/sgeCA/certs
Creating /scratch2/eddy/sge_sec/default/common/sgeCA/crl
Creating /scratch2/eddy/sge_sec/default/common/sgeCA/newcerts
Creating /scratch2/eddy/sge_sec/default/common/sgeCA/serial
Creating /scratch2/eddy/sge_sec/default/common/sgeCA/index.txt
Creating /var/sgeCA/port6789/default/userkeys
Creating /var/sgeCA/port6789/default/private
Hit Return to continue >>
```

After setting up the directories, the CA-specific certificate and private key are generated. The Sun Grid Engine system uses either pseudo random data from a special file or, if available, `/dev/random` for seeding the pseudo random number generator (PRNG). (For more detailed information regarding random numbers, see `http://www.openssl.org/support/faq.html` and `http://www.cosy.sbg.ac.at/~andi`.)

After the installation of the CA infrastructure, application certificates, user certificates, and private keys are created and signed by the CA for the admin user, for the pseudo daemon user, and for the user, `root`. The script—whose output is similar to the example in CODE EXAMPLE 2-2—first queries for site information.

CODE EXAMPLE 2-2 CSP Installation Script—Information Collection

```
Creating CA certificate and private key
---------------------------------------

Please give some basic parameters to create the distinguished name (DN)
for the certificates.

We will ask for

     - the two letter country code
     - the state
     - the location, e.g city or your buildingcode
     - the organization (e.g. your company name)
     - the organizational unit, e.g. your department
     - the email address of the CA administrator (you!)

Hit Return to continue >>

Please enter your two letter country code, e.g. >US< >> DE
Please enter your state >> Bavaria
Please enter your location, e.g city or buildingcode >> Regensburg
Please enter the name of your organization >> Myorg
Please enter your organizational unit, e.g. your department >> Mydept
Please enter the email address of the CA administrator >> admin@my.org


You selected the following basic data for the distinguished name of
your certificates:

Country code:         C=DE
State:                ST=Bavaria
Location:             L=Regensburg
Organization:         O=Myorg
Organizational unit:  OU=Mydept
CA email address:     emailAddress=admin@my.org


Do you want to use these data (y/n) [y] >>
```

After you confirm that the information you supplied is correct, the installation program continues with the CA certificate and private key generation, beginning with setting up the CA infrastructure. The script output is similar to the example in CODE EXAMPLE 2-3.

CODE EXAMPLE 2-3 CSP Installation Script—CA Infrastructure Creation

```
Creating RANDFILE from >/kernel/genunix< in
>/var/sgeCA/port6789/default/private/rand.seed<

1513428 semi-random bytes loaded
Creating CA certificate and private key

Using configuration from /tmp/sge_ca14364.tmp
Generating a 1024 bit RSA private key
.....++++++
.............++++++
writing new private key to '/var/sgeCA/port6789/default/private/cakey.pem'
-----
Hit Return to continue >>
```

After the installation of the CA infrastructure, the CA creates and signs application and user certificates and private keys for the pseudo daemon user and for the `root` user. Script output is similar to that shown in (which continues to the next pages). Note that some of the lines in the example are abbreviated to fit each single line on these pages. The abbreviated portions are indicated by (...).

CODE EXAMPLE 2-4 CSP Installation Script—Certificate and Private Key Creation

```
Creating Daemon certificate and key
-----------------------------------

Creating RANDFILE from >/kernel/genunix< in >/var/sgeCA/(…)/rand.seed<

1513428 semi-random bytes loaded
Using configuration from /tmp/sge_ca14364.tmp
Generating a 1024 bit RSA private key
.............++++++
.............++++++
writing new private key to '/var/sgeCA/port6789/default/private/key.pem'
-----
Using configuration from /tmp/sge_ca14364.tmp
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
countryName           :PRINTABLE:'DE'
stateOrProvinceName   :PRINTABLE:'Bavaria'
localityName          :PRINTABLE:'Regensburg'
```

CODE EXAMPLE 2-4 CSP Installation Script—Certificate and Private Key Creation *(Continued)*

```
organizationName      :PRINTABLE:'Myorg'
organizationalUnitName:PRINTABLE:'Mydept'
uniqueIdentifier      :PRINTABLE:'root'
commonName            :PRINTABLE:'SGE Daemon'
emailAddress          :IA5STRING:'none'
Certificate is to be certified until Mar  5 13:50:57 2003 GMT (365 days)

Write out database with 1 new entries
Data Base Updated
created and signed certificate for SGE daemons
Creating RANDFILE from >/kernel/genunix< in>/var/(...)/userkeys/root/rand.seed<

1513428 semi-random bytes loaded
Using configuration from /tmp/sge_ca14364.tmp
Generating a 1024 bit RSA private key
............++++++
.................++++++
writing new private key to '/var/sgeCA/port6789/default/userkeys/root/key.pem'
-----
Using configuration from /tmp/sge_ca14364.tmp
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
countryName           :PRINTABLE:'DE'
stateOrProvinceName   :PRINTABLE:'Bavaria'
localityName          :PRINTABLE:'Regensburg'
organizationName      :PRINTABLE:'Myorg'
organizationalUnitName:PRINTABLE:'Mydept'
uniqueIdentifier      :PRINTABLE:'root'
commonName            :PRINTABLE:'SGE install user'
emailAddress          :IA5STRING:'none'
Certificate is to be certified until Mar  5 13:50:59 2003 GMT (365 days)

Write out database with 1 new entries
Data Base Updated
created and signed certificate for user >root< in >/var/(...)/userkeys/root<
Creating RANDFILE from >/kernel/genunix< in >/(...)/userkeys/eddy/rand.seed<
1513428 semi-random bytes loaded
Using configuration from /tmp/sge_ca14364.tmp
Generating a 1024 bit RSA private key
.............++++++
..................................................++++++
writing new private key to '/var/sgeCA/port6789/default/userkeys/eddy/key.pem'
-----
Using configuration from /tmp/sge_ca14364.tmp
Check that the request matches the signature
Signature ok
```

CODE EXAMPLE 2-4 CSP Installation Script—Certificate and Private Key Creation *(Continued)*

```
The Subjects Distinguished Name is as follows
countryName           :PRINTABLE:'DE'
stateOrProvinceName   :PRINTABLE:'Bavaria'
localityName          :PRINTABLE:'Regensburg'
organizationName      :PRINTABLE:'Myorg'
organizationalUnitName:PRINTABLE:'Mydept'
uniqueIdentifier      :PRINTABLE:'root'
commonName            :PRINTABLE:'SGE install user'
emailAddress          :IA5STRING:'none'
Certificate is to be certified until Mar  5 13:50:59 2003 GMT (365 days)

Write out database with 1 new entries
Data Base Updated
created and signed certificate for user >root< in >/var/(…)/userkeys/root<
Creating RANDFILE from >/kernel/genunix< in >/var/(…)/userkeys/eddy/rand.seed<

1513428 semi-random bytes loaded
Using configuration from /tmp/sge_ca14364.tmp
Generating a 1024 bit RSA private key
.............++++++
.....................................................++++++
writing new private key to '/var/sgeCA/port6789/default/userkeys/eddy/key.pem'
-----
Using configuration from /tmp/sge_ca14364.tmp
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
countryName           :PRINTABLE:'DE'
stateOrProvinceName   :PRINTABLE:'Bavaria'
localityName          :PRINTABLE:'Regensburg'
organizationName      :PRINTABLE:'Myorg'
organizationalUnitName:PRINTABLE:'Mydept'
uniqueIdentifier      :PRINTABLE:'eddy'
commonName            :PRINTABLE:'SGE admin user'
emailAddress          :IA5STRING:'none'
Certificate is to be certified until Mar  5 13:51:02 2003 GMT (365 days)

Write out database with 1 new entries
Data Base Updated
created and signed certificate for user >root< in >/var/(…)/userkeys/root<
Creating RANDFILE from >/kernel/genunix< in >/var/(…)/userkeys/eddy/rand.seed<

1513428 semi-random bytes loaded
Using configuration from /tmp/sge_ca14364.tmp
Generating a 1024 bit RSA private key
.............++++++
.....................................................++++++
```

CODE EXAMPLE 2-4 CSP Installation Script—Certificate and Private Key Creation *(Continued)*

```
writing new private key to '/var/sgeCA/port6789/default/userkeys/eddy/key.pem'
-----
Using configuration from /tmp/sge_ca14364.tmp
Check that the request matches the signature
Signature ok
The Subjects Distinguished Name is as follows
countryName           :PRINTABLE:'DE'
stateOrProvinceName   :PRINTABLE:'Bavaria'
localityName          :PRINTABLE:'Regensburg'
organizationName      :PRINTABLE:'Myorg'
organizationalUnitName:PRINTABLE:'Mydept'
uniqueIdentifier      :PRINTABLE:'eddy'
commonName            :PRINTABLE:'SGE admin user'
emailAddress          :IA5STRING:'none'
Certificate is to be certified until Mar  5 13:51:02 2003 GMT (365 days

Write out database with 1 new entries
Data Base Updated
created and signed certificate for user >eddy< in >/var/(...)/userkeys/eddy<
Hit Return to continue >>
```

After the security related setup of the master host, `sge_qmaster`, is completed, the script prompts you to continue with the remainder of the installation procedure, similar to the example in CODE EXAMPLE 2-5.

CODE EXAMPLE 2-5 CSP Installation Script—Continuation of Installation

```
SGE startup script
--------------------


Your system wide SGE startup script is installed as:

     "/scratch2/eddy/sge_sec/default/common/rcsge"

Hit Return to continue >>
```

3. **Do one of the following.**

   a. **If you believe that the shared file system is *not* secure enough to hold the CSP security information in a place that can be accessed by the execution daemons, proceed to Step 4.**

b. If you believe that the shared file system *is* secure enough, continue with the basic installation procedure as outlined in the section, "How To Install Execution Hosts" on page 33.

Remember to add the `-csp` flag when you call the "`./inst`" script for the execution host installation.

After completing all remaining installation steps, turn to the instructions in the section, "How To Generate Certificates and Private Keys for Users" on page 45.

4. (Optional) If the shared file system is not secure enough to hold the CSP security information in a place that can be accessed by the execution daemons as well, you must transfer the directory containing the daemon's private key and the random file to the *execution* host.

a. As `root` on the *master* host, enter the following commands to prepare to copy the private keys to the machines you will set up as execution hosts.

```
# umask 077
# cd /
# tar cvpf /var/sgeCA/port6789.tar /var/sgeCA/port6789/default
```

b. As `root` on each *execution* host, enter the following commands to copy the files.

```
# umask 077
# cd /
# scp <masterhost>:/var/sgeCA/port6789.tar .
# umask 022
# tar xvpf /port6789.tar
# rm /port6789.tar
```

c. Enter the following command to verify the file permissions.

```
# ls -lR /var/sgeCA/port6789/
```

The output should look similar to the example in CODE EXAMPLE 2-6.

CODE EXAMPLE 2-6 File Permission Verification

```
/var/sgeCA/port6789/:
total 2
drwxr-xr-x   4 eddy     other        512 Mar  6 10:52 default
/var/sgeCA/port6789/default:
total 4
drwx------   2 eddy     staff        512 Mar  6 10:53 private
drwxr-xr-x   4 eddy     staff        512 Mar  6 10:54 userkeys
/var/sgeCA/port6789/default/private:
total 8
-rw-------   1 eddy     staff        887 Mar  6 10:53 cakey.pem
-rw-------   1 eddy     staff        887 Mar  6 10:53 key.pem
-rw-------   1 eddy     staff       1024 Mar  6 10:54 rand.seed
-rw-------   1 eddy     staff        761 Mar  6 10:53 req.pem
/var/sgeCA/port6789/default/userkeys:
total 4
dr-x------   2 eddy     staff        512 Mar  6 10:54 eddy
dr-x------   2 root     staff        512 Mar  6 10:54 root
/var/sgeCA/port6789/default/userkeys/eddy:
total 16
-r--------   1 eddy     staff       3811 Mar  6 10:54 cert.pem
-r--------   1 eddy     staff        887 Mar  6 10:54 key.pem
-r--------   1 eddy     staff       2048 Mar  6 10:54 rand.seed
-r--------   1 eddy     staff        769 Mar  6 10:54 req.pem
/var/sgeCA/port6789/default/userkeys/root:
total 16
-r--------   1 root     staff       3805 Mar  6 10:54 cert.pem
-r--------   1 root     staff        887 Mar  6 10:54 key.pem
-r--------   1 root     staff       2048 Mar  6 10:53 rand.seed
-r--------   1 root     staff        769 Mar  6 10:54 req.pem
```

d. Continue with Sun Grid Engine installation by entering the following commands.

```
# cd $SGE_ROOT
# ./install_execd -csp
```

e. Follow the remainder of installation instructions beginning with Step 3 in the section, “How To Install Execution Hosts” on page 33, remembering to add the `-csp` flag to launch installation scripts.

After completing all remaining installation steps, turn to the instructions in the section, “How To Generate Certificates and Private Keys for Users” on page 45.

▼ How To Generate Certificates and Private Keys for Users

To use the CSP-secured system, the user must have access to a user-specific certificate and private key. The most convenient method of doing this is to create a text file identifying the users.

1. **Create and save a text file that identifies users.**

   Use the format of the file, `myusers.txt`, shown in the following example. (The fields of the file are *UNIX_username:Gecos_field:email_address*.)

```
eddy:Eddy Smith:eddy@my.org
sarah:Sarah Miller:sarah@my.org
leo:Leo Lion:leo@my.org
```

2. **As root on the master host, enter the following command.**

```
# $SGE_ROOT/util/sgeCA/sge_ca -usercert myusers.txt
```

3. **Confirm by entering the following command.**

```
# ls -l /var/sgeCA/port6789/default/userkeys
```

   This directory listing should produce output similar to the following.

```
dr-x------   2 eddy  staff        512 Mar  5 16:13 eddy
dr-x------   2 sarah staff        512 Mar  5 16:13 sarah
dr-x------   2 leo   staff        512 Mar  5 16:13 leo
```

4. **Tell each user that you have listed in the file (`myusers.txt` in the example) to install the security-related files into their `$HOME/.sge` directories by entering the following commands.**

```
% source $SGE_ROOT/default/common/settings.csh
% $SGE_ROOT/util/sgeCA/sge_ca -copy
```

The users should see the following confirmation (user `eddy` in the example).

```
Certificate and private key for user eddy have been installed
```

For every Sun Grid Engine installation, a subdirectory for the corresponding `COMMD_PORT` number is installed. The following example, based on the `myusers.txt` file, results from issuing the command preceding the output.

```
% ls -lR $HOME/.sge
/home/eddy/.sge:
total 2
drwxr-xr-x   3 eddy staff        512 Mar  5 16:20 port6789

/home/eddy/.sge/port6789:
total 2
drwxr-xr-x   4 eddy staff        512 Mar  5 16:20 default

/home/eddy/.sge/port6789/default:
total 4
drwxr-xr-x   2 eddy staff        512 Mar  5 16:20 certs
drwx------   2 eddy staff        512 Mar  5 16:20 private

/home/eddy/.sge/port6789/default/certs:
total 8
-r--r--r--   1 eddy staff       3859 Mar  5 16:20 cert.pem

/home/eddy/.sge/port6789/default/private:
total 6
-r--------   1 eddy staff        887 Mar  5 16:20 key.pem
-r--------   1 eddy staff       2048 Mar  5 16:20 rand.seed
```

▼ How To Check Certificates

- **Depending on what you want to do, enter one or more of the following commands.**

Display a Certificate

Type the following as one string (the command is too long to fit on one line in this Guide), with a space between the `-in` and `~/.sge` components.

```
% $SGE_ROOT/utilbin/$ARCH/openssl x509 -in
~/.sge/port6789/default/certs/cert.pem -text
```

Check Issuer

Type the following as one string (the command is too long to fit on one line in this Guide), with a space between the `-in` and `~/.sge` components.

```
% $SGE_ROOT/utilbin/$ARCH/openssl x509 -issuer -in
~/.sge/port6789/default/certs/cert.pem -noout
```

Check Subject

Type the following as one string (the command is too long to fit on one line in this Guide), with a space between the `-in` and `~/.sge` components.

```
% $SGE_ROOT/utilbin/$ARCH/openssl x509 -subject -in
~/.sge/port6789/default/certs/cert.pem -noout
```

Show Email of Certificate

Type the following as one string (the command is too long to fit on one line in this Guide), with a space between the `-in` and `~/.sge` components.

```
% $SGE_ROOT/utilbin/$ARCH/openssl x509 -email -in
~/.sge/default/port6789/certs/cert.pem -noout
```

Show Validity

Type the following as one string (the command is too long to fit on one line in this Guide), with a space between the `-in` and `~/.sge` components.

```
% $SGE_ROOT/utilbin/$ARCH/openssl x509 -dates -in
~/.sge/default/port6789/certs/cert.pem -noout
```

Show Fingerprint

Type the following as one string (the command is too long to fit on one line in this Guide), with a space between the `-in` and `~/.sge` components.

```
% $SGE_ROOT/utilbin/$ARCH/openssl x509 -fingerprint -in
~/.sge/port6789/default/certs/cert.pem -noout
```

▼ How To Verify the Installation

On the Master Host

1. **Log in to the master host.**
2. **Execute one of the following commands, depending on the operating system you are running.**

a. On BSD-based UNIX systems, enter the following command.

```
% ps -ax
```

b. On systems running a UNIX System 5--based operating system (such as the Solaris Operating Environment), enter the following command.

```
% ps -ef
```

3. Look through the output for `sge` **strings that are similar to the following examples.**

On a BSD-based UNIX system, you should see output similar to the following.

```
14673 p1 S <  2:12 /gridware/sge/bin/solaris/sge_commd
14676 p1 S <  4:47 /gridware/sge/bin/solaris/sge_qmaster
14678 p1 S <  9:22 /gridware/sge/bin/solaris/sge_schedd
```

In the case of a UNIX System 5-based system, you should see output similar to the following.

```
root 439 1 0 Jun 2 ? 3:37 /gridware/sge/bin/solaris/sge_commd
root 439 1 0 Jun 2 ? 3:37 /gridware/sge/bin/solaris/sge_qmaster
root 446 1 0 Jun 2 ? 3:37 /gridware/sge/bin/solaris/sge_schedd
```

If you *do not see* the appropriate string, one or more Sun Grid Engine daemons required on the master host are not running on this machine (you can look into the file <*sge_root*>/<*cell*>/`common/act_qmaster` whether you really are on the master host). Go on to the next step.

4. (Optional) Restart the daemons by hand.

See the section, “About Daemons and Hosts” on page 169 for instructions on how to proceed.

On the Execution Hosts

1. Log in to the execution hosts on which you ran the Sun Grid Engine execution host installation procedure.

2. Refer to Step 2 in the command host procedure to determine the appropriate `ps` **command for your system, and enter that command.**

3. **Look for an** `sge` **string in the output.**

   On a BSD-based UNIXsystem, you should see output similar to the following.

```
14685 p1 S <    1:13 /gridware/sge/bin//sge_commd
14688 p1 S <    4:27 /gridware/sge/bin/solaris/sge_execd
```

   In the case of a UNIX System 5-based system, such as the Solaris Operating Environment, you should see output similar to the following.

```
root 169 1 0 Jun 22 ? 2:04 /gridware/sge/bin/solaris/sge_commd
root 171 1 0 Jun 22 ? 7:11 /gridware/sge/bin/solaris/sge_execd
```

   If you *do not see* similar output, one or more daemons required on the execution host are not running. Go on to the next step.

4. **(Optional) Restart the daemons by hand.**

   See the section, “About Daemons and Hosts” on page 169 for instructions on how to proceed.

Trying Commands

If both the necessary daemons run on the master and execution hosts the Sun Grid Engine system should be operational. Check by issuing a trial command.

1. **Log in to either the master host or another administrative host.**

   Make sure to include the path where you installed the Sun Grid Engine binaries into your standard search path.

2. **From the command line, enter the following command.**

```
% qconf -sconf
```

   This `qconf` command displays the current global cluster configuration (see the section, “The Basic Cluster Configuration” on page 157). If this command fails, most probably either your `SGE_ROOT` environment variable is set inappropriately or `qconf` fails to contact the `sge_commd` associated with `sge_qmaster`. Go on to the next step.

3. **Check whether the script files,** *<sge_root>*`/`*<cell>*`/common/settings.csh` **or** *<sge_root>*`/`*<cell>*`/common/settings.sh` **set the environment variable,** `COMMD_PORT`**.**

   If so, make sure that the environment variable `COMMD_PORT` is set to that particular value before you try the above command again. If the `COMMD_PORT` variable is not used in the settings files, the services database (e.g., `/etc/services` or the NIS services map) on the machine from which you executed the command must provide a `sge_commd` entry. If this is not the case, add such an entry to the machine's services database and give it the same value as is configured on the Sun Grid Engine master host, and proceed to the next step.

4. **Retry the** `qconf` **command.**

Preparing To Submit Jobs

Before you start submitting batch scripts to the Sun Grid Engine system, check if your site's standard and your personal shell resource files (`.cshrc`, `.profile` or `.kshrc`) contain commands such as `stty` (batch jobs do not have a terminal connection by default and, therefore, calls to `stty` will result in an error).

1. **Log in to the master host.**
2. **Enter the following command.**

```
% rsh an_exec_host date
```

an_exec_host refers to one of the already installed execution hosts that you are going to use (you should check on all execution hosts if your login or home directories differ from host to host). The `rsh` command should give you an output very similar to the `date` command executed locally on the master host. If there are any additional lines containing error messages, you must eliminate the cause of the errors before you are able to run a batch job successfully.

For all command interpreters you can check on an actual terminal connection before you execute a command such as `stty`. The following is a Bourne-/Korn-Shell example how to do this:

```
tty -s
if [ $? = 0 ]; then
   stty erase ^H
fi
```

The C-Shell syntax is very similar:

```
tty -s
if ( $status = 0 ) then
   stty erase ^H
endif
```

3. **Submit one of the sample scripts contained in the** <*sge_root*>`/examples/jobs` **directory.**

   Enter the following command.

   ```
   % qsub script_path
   ```

4. **Use the Sun Grid Engine** `qstat` **command to monitor the job's behavior.**

   See "Submitting Batch Jobs" on page 75 for more information about submitting and monitoring batch jobs.

5. **After the job has finished execution, check your home directory for the redirected stdout/stderr files,** <*script_name*>`.e`<*job_id*> **and** <*script_name*>`.o`<*job_id*> **with** <*job_id*> **being a consecutive unique integer number assigned to each job.**

   In case of problems, see Chapter 11, "Troubleshooting."

PART III Using Sun Grid Engine 5.3 Software

Intended primarily for the user—that is, one who does not also perform the duties of a system administrator (see Part 4, "Administration" on page 139)—this part of the *Sun Grid Engine 5.3 Administration and User's Guide* consists of three chapters.

- **Chapter 3** – "Navigating Through the Sun Grid Engine 5.3 Program" on page 55

  This chapter introduces you to some Sun Grid Engine 5.3 basics, and includes instructions on how to list various resources.

- **Chapter 4** – "Submitting Jobs" on page 69

  This chapter provides complete instructions for submitting jobs by way of the Sun Grid Engine 5.3 system, and begins with a "practice" job submission that acquaints you with the process.

- **Chapter 5** – "Checkpointing, Monitoring, and Controlling Jobs" on page 111

  This chapter explains the concepts of job control and includes instructions for accomplishing various job control tasks.

Each chapter in Part 3 includes both background information about, and detailed instructions for, accomplishing a myriad of tasks by way of the Sun Grid Engine 5.3 system.

CHAPTER 3

Navigating Through the Sun Grid Engine 5.3 Program

This chapter introduces you to some basic Sun Grid Engine 5.3 concepts and terminology that will help you begin to use the software. For complete background information about the product, including a comprehensive glossary, see Chapter 1, "Introduction to Sun Grid Engine 5.3" on page 1.

This chapter also includes instructions for accomplishing the following tasks.

Sun Grid Engine User Types and Operations

User types are divided into four categories in Sun Grid Engine.

- **Managers** – Managers have full capabilities to manipulate Sun Grid Engine. By default, the superusers of any machine hosting a queue have manager privileges.
- **Operators** – The operators can perform many of the same commands as the manager, with the exception of making configuration changes by adding, deleting, or modifying queues, for example.

- **Owners** – The queue owners are allowed to suspend or enable the owned queues or jobs within them, but have no further management permissions.
- **Users** – Users have certain access permissions, as described in "User Access Permissions" on page 66, but no cluster or queue management capabilities.

TABLE 3-1 shows the Sun Grid Engine 5.3 command capabilities that are available to the different user categories.

TABLE 3-1 User Categories and Associated Command Capabilities

Command	Manager	Operator	Owner	User
`qacct`	Full	Full	Own jobs only	Own jobs only
`qalter`	Full	Full	Own jobs only	Own jobs only
`qconf`	Full	No system setup modifications	Show only configurations and access permissions	Show only configurations and access permissions
`qdel`	Full	Full	Own jobs only	Own jobs only
`qhold`	Full	Full	Own jobs only	Own jobs only
`qhost`	Full	Full	Full	Full
`qlogin`	Full	Full	Full	Full
`qmod`	Full	Full	Own jobs and owned queues only	Own jobs only
`qmon`	Full	No system setup modifications	No configuration changes	No configuration changes
`qrexec`	Full	Full	Full	Full
`qselect`	Full	Full	Full	Full
`qsh`	Full	Full	Full	Full
`qstat`	Full	Full	Full	Full
`qsub`	Full	Full	Full	Full

Queues and Queue Properties

In order to be able to optimally utilize the Sun Grid Engine system at your site, you should become familiar with the queue structure and the properties of the queues that are configured for your Sun Grid Engine system.

The QMON Browser

Sun Grid Engine features a graphical user interface (GUI) command tool, the QMON browser. The QMON browser provides a myriad of Sun Grid Engine functions, including job submission, job control, and important information gathering.

▼ How To Launch the QMON Browser

- **From the command line, enter the following command.**

```
% qmon
```

After a message window is displayed, the QMON main control panel appears, similar to the following (see FIGURE 1-2 to identify the meaning of the icons).

FIGURE 3-1 QMON Main Control Menu

Many instructions in this manual call for using the QMON browser. The names of the icon buttons, which are descriptive of their functions, appear on screen as you pass the mouse pointer over them.

(For instructions on how to customize the QMON browser, see “Customizing QMON” on page 9.)

The Queue Control QMON Dialogue Box

The QMON Queue Control dialogue box displayed and described in the section, “How To Control Queues with QMON” on page 132 provides a quick overview on the installed queues and their current status.

▼ How To Display a List of Queues

- **Enter the following command.**

```
% qconf -sql
```

▼ How To Display Queue Properties

You can use either QMON or the command line to display queue properties.

Using the QMON Browser

1. **From the main QMON menu, click the Browser icon.**
2. **Click the Queue button.**
3. **In the Queue Control dialog, move the mouse pointer over the icon for the appropriate queue.**

   FIGURE 3-2 is a partial example of the Queue property information that is displayed.

   FIGURE 3-2 QMON Browser Display of Queue Properties

From the Command Line

- **Enter the following command.**

```
% qconf -sq queue_name
```

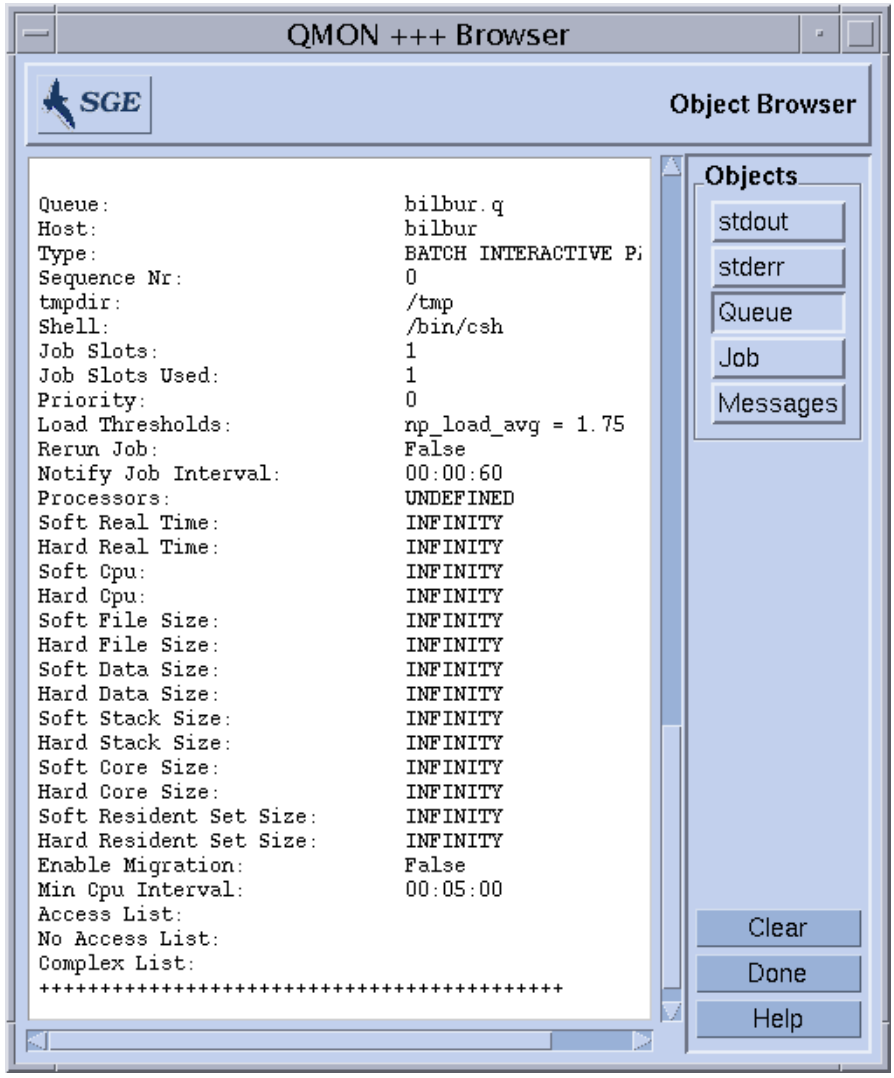
Information similar to that shown in FIGURE 3-2 is displayed.

Interpreting Queue Property Information

You can find a detailed description of each queue property in the queue_conf manual page and in the queue_conf section of the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual.*

Following is a list of some of the most important parameters.

- qname – The queue name as requested.



- hostname – The host of the queue.
- processors – The processors of a multi processor system, to which the queue has access.
- qtype – The type of job which is allowed to run in this queue. Currently, this is either batch, interactive, checkpointing, parallel or any combination thereof or transfer alternatively
- slots – The number of jobs which may be executed concurrently in that queue.
- owner_list – The owners of the queue as explained in the section, “Managers, Operators and Owners” on page 67

- user_lists – The user or group identifiers in the user access lists (see “User Access Permissions” on page 66) enlisted under this parameter may access the queue.
- xuser_lists – The user or group identifiers in the user access lists (see “User Access Permissions” on page 66) enlisted under this parameter may *not* access the queue.
- complex_list – The complexes enlisted under this parameter are associated with the queue and the attributes contained in these complexes contribute to the set of requestable attributes for the queue (see “Requestable Attributes” on page 62).
- complex_values – Assigns capacities as provided for this queue for certain complex attributes (see “Requestable Attributes” on page 62).

Host Functionality

Clicking the Host Configuration button in the `QMON` Main menu displays an overview of the functionality that is associated with the hosts in your Sun Grid Engine cluster. However, without Sun Grid Engine manager privileges, you may not apply any changes to the presented configuration.

The host configuration dialogues are described in the section, “About Daemons and Hosts” on page 143. The following sections provide the commands to retrieve this kind of information from the command line.

▼ How To Find the Name of the Master Host

The location of the master host should be transparent for the user as the master host may migrate between the current master host and one of the shadow master hosts at any time.

- **Using a text editor, open the** *<sge_root>/<cell>*`/common/act_qmaster` **file.**

  The name of the current master host is in the file.

▼ How To Display a List of Execution Hosts

To display a list of hosts being configured as execution hosts in your cluster please use the commands:

```
% qconf -sel
% qconf -se hostname
% qhost
```

The first command displays a list of the names of all hosts being currently configured as execution hosts. The second command displays detailed information about the specified execution host. The third command displays status and load information about the execution hosts. Please refer to the `host_conf` manual page for details on the information displayed via `qconf` and to the `qhost` manual page for details on its output and further options.

▼ How To Display a List of Administration Hosts

The list of hosts with administrative permission can be displayed with the following command:.

```
% qconf -sh
```

▼ How To Display a List of Submit Hosts

The list of submit host can be displayed with the following command.

```
% qconf -ss
```

Requestable Attributes

When submitting a Sun Grid Engine job a requirement profile of the job can be specified. The user can specify attributes or characteristics of a host or queue which the job requires to run successfully. Sun Grid Engine will map these job requirements onto the host and queue configurations of the Sun Grid Engine cluster and will, therefore, find the suitable hosts for a job.

The attributes that can be used to specify the job requirements are either related to the Sun Grid Engine cluster (e.g., space required on a network shared disk), to the hosts (e.g., operating system architecture), or to the queues (e.g., permitted CPU time), or the attributes are derived from site policies such as the availability of installed software only on some hosts.

The available attributes include the queue property list (see “Queues and Queue Properties” on page 56), the list of global and host-related attributes (see “Complex Types” on page 186), as well as administrator-defined attributes. For convenience, however, the Sun Grid Engine administrator commonly chooses to define only a subset of all available attributes to be requestable.

The attributes being currently requestable are displayed in the Requested Resources sub-dialogue (see FIGURE 3-3) to the QMON Submit dialogue box (refer to the section, “Submitting Batch Jobs” on page 75 for detailed information on how to submit jobs). They are enlisted in the Available Resources selection list.

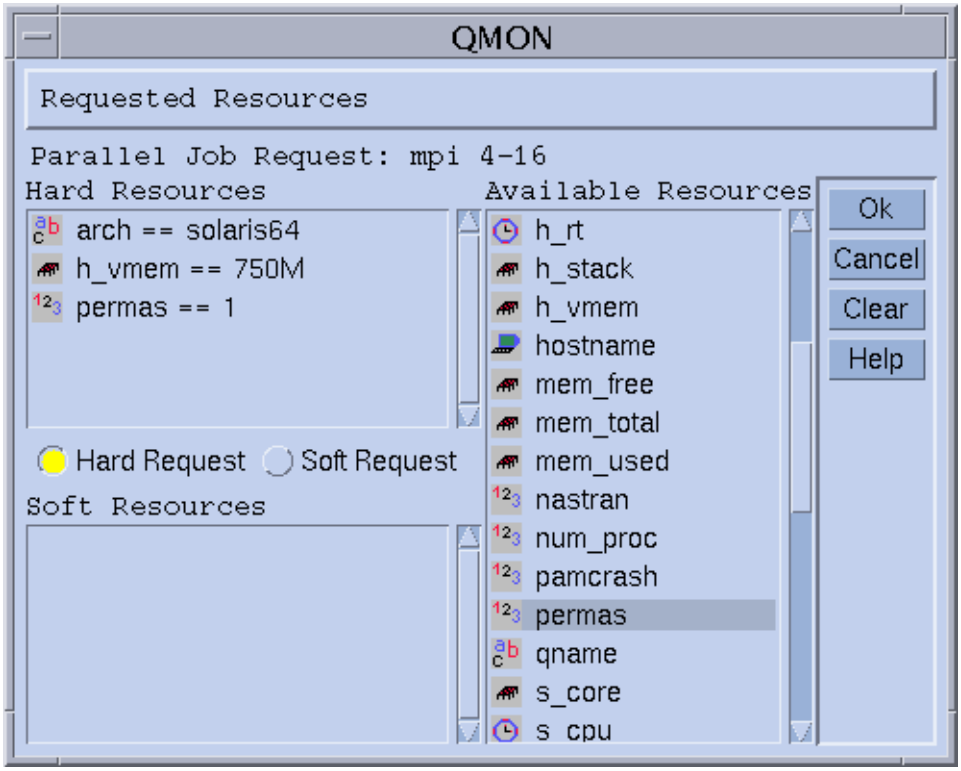


FIGURE 3-3 Requested Resources Dialogue Box

▼ How To Display a List of Requestable Attributes

1. **From the command line, display a list of configured *complexes* by entering the following command:**

```
% qconf -scl
```

A complex contains the definition for a set of attributes. There are three standard complexes:

- `global`– For the cluster global attributes (optional)
- `host` – For the host-specific attributes
- `queue` – For the queue property attributes

Any further complex names printed as a result of the above command refers to an administrator-defined complex (see Chapter 8, “The Complexes Concept” on page 183 or the complex format description in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for more information on complexes).

2. **Display the attributes of a particular complex by entering the following command:**

```
% qconf -sc complex_name[,...]
```

The output for the queue complex might for example look as shown in TABLE 3-2.

TABLE 3-2 `queue` Complex Attributes Displayed

#Name	Shortcut	Type	Value	Relop	Requestable	Consumable	Default
qname	q	STRING	NONE	==	YES	NO	NONE
hostname	h	HOST	unknown	==	YES	NO	NONE
tmpdir	tmp	STRING	NONE	==	NO	NO	NONE
calendar	c	STRING	NONE	==	YES	NO	NONE
priority	pr	INT	0	>=	NO	NO	0
seq_no	seq	INT	0	==	NO	NO	0
rerun	re	INT	0	==	NO	NO	0
s_rt	s_rt	TIME	0:0:0	<=	NO	NO	0:0:0
h_rt	h_rt	TIME	0:0:0	<=	YES	NO	0:0:0
s_cpu	s_cpu	TIME	0:0:0	<=	NO	NO	0:0:0
h_cpu	h_cpu	TIME	0:0:0	<=	YES	NO	0:0:0
s_data	s_data	MEMORY	0	<=	NO	NO	0
h_data	h_data	MEMORY	0	<=	YES	NO	0
s_stack	s_stack	MEMORY	0	<=	NO	NO	0
h_stack	h_stack	MEMORY	0	<=	NO	NO	0
s_core	s_core	MEMORY	0	<=	NO	NO	0
h_core	h_core	MEMORY	0	<=	NO	NO	0
s_rss	s_rss	MEMORY	0	<=	NO	NO	0
h_rss	h_rss	MEMORY	0	<=	YES	NO	0
min_cpu_interval	mci	TIME	0:0:0	<=	NO	NO	0:0:0
max_migr_time	mmt	TIME	0:0:0	<=	NO	NO	0:0:0

TABLE 3-2 queue Complex Attributes Displayed *(Continued)*

#Name	Shortcut	Type	Value	Relop	Requestable	Consumable	Default
qname	q	STRING	NONE	==	YES	NO	NONE
hostname	h	HOST	unknown	==	YES	NO	NONE
tmpdir	tmp	STRING	NONE	==	NO	NO	NONE
calendar	c	STRING	NONE	==	YES	NO	NONE
priority	pr	INT	0	>=	NO	NO	0
seq_no	seq	INT	0	==	NO	NO	0
max_no_migr	mnm	TIME	0:0:0	<=	NO	NO	0:0:0

The column `name` is basically identical to the first column displayed by the `qconf -sq` command. The queue attributes cover most of the Sun Grid Engine queue properties. The `shortcut` column contains administrator definable abbreviations for the full names in the first column. Either the full name or the shortcut can be supplied in the request option of a `qsub` command by the user.

The column `requestable` tells whether the Corresponding entry may be used in `qsub` or not. Thus the administrator can, for example, disallow the cluster's users to request certain machines / queues for their jobs directly, simply by setting the entries `qname` and / or `qhostname` to be not requestable. Doing this, implies that feasible user requests can be met in general by multiple queues, which enforces the load balancing capabilities of Sun Grid Engine.

The column `relop` defines the relation operation used in order to compute whether a queue meets a user request or not. The comparison executed is:

- `User_Request relop Queue/Host/...-Property`

If the result of the comparison is false, the user's job cannot be run in the considered queue. Let, as an example, the queue *q1* be configured with a soft cpu time limit (see the `queue_conf` and the `setrlimit` manual pages for a description of user process limits) of 100 seconds while the queue *q2* is configured to provide 1000 seconds soft cpu time limit.

The columns `consumables` and `default` are meaningful for the administrator to declare so called consumable resources (see the section, "Consumable Resources" on page 194). The user requests consumables just like any other attribute. The Sun Grid Engine internal bookkeeping for the resources is different, however.

Assume that a user submits the following request.

```
% qsub -l s_cpu=0:5:0 nastran.sh
```

The `s_cpu=0:5:0` request (see the `qsub` manual page for details on the syntax) asks for a queue which at least grants for 5 minutes of soft limit cpu time. Therefore, only queues providing at least 5 minutes soft CPU runtime limit are setup properly to run the job.

Note – Sun Grid Engine will only consider workload information in the scheduling process if more than one queue is able to run a job.

User Access Permissions

Access to queues and other Sun Grid Engine facilities (e.g., parallel environment interfaces; see “Parallel Jobs” on page 101) can be restricted for certain users or user groups by the Sun Grid Engine administrator.

Note – Sun Grid Engine automatically takes into account the access restrictions configured by the cluster administration. The following sections are only important if you want to query your personal access permission.

For the purpose of restricting access permissions, the administrator creates and maintains so called access lists (or in short *ACLs*). The ACLs contain arbitrary user and UNIX group names. The ACLs are then added to *access-allowed-* or *access-denied-*lists in the queue or in the parallel environment interface configurations (see `queue_conf` or `sge_pe` in *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* section 5, respectively).

User‘s belonging to ACLs which are enlisted in access-allowed-lists have permission to access the queue or the parallel environment interface. User‘s being members of ACLs in access-denied-lists may not access the concerning resource.

The Userset Configuration dialogue box opened via the User Configuration icon button in the `QMON` Main menu allows you to query for the ACLs you have access to via the Userset Configuration dialogue box. Refer to Chapter 9, “Managing User Access and Policies” on page 213 for details.

From the command line a list of the currently configured ACLs can be obtained by the command:

```
% qconf -sul
```

The entries in one or multiple access lists are printed with the command:

```
% qconf -su acl_name[,...]
```

The ACLs consist of user account names and UNIX group names with the UNIX group names being identified by a prefixed "@" sign. This way you can determine to which ACLs your account belongs.

Note – In case you have permission to switch your primary UNIX group with the `newgrp` command, your access permissions may change.

You can now check for those queues or parallel environment interfaces to which you have access or to which access is denied for you. Please query the queue or parallel environment interface configuration as described in "Queues and Queue Properties" on page 56 and "How To Configure PEs with QMON" on page 246. The access-allowed-lists are named `user_lists`. The access-denied-list have the names `xuser_lists`. If your user account or primary UNIX group is associated with a access-allowed-list you are allowed to access the concerning resource. If you are associated with a access-denied-list you may not access the queue or parallel environment interface. If both lists are empty every user with a valid account can access the concerning resource.

Managers, Operators and Owners

A list of Sun Grid Engine managers can be obtained by:

```
% qconf -sm
```

and a list of operators by:

```
% qconf -so
```

Note – The superuser of a Sun Grid Engine administration host is considered as manager by default.

The users, which are owners to a certain queue are contained in the queue configuration database as described in section “Queues and Queue Properties” on page 56. This database can be retrieved by executing:

```
% qconf -sq queue_name
```

The concerning queue configuration entry is called `owners`.

CHAPTER 4

Submitting Jobs

This chapter provides background information about, and instructions for, using Sun Grid Engine 5.3 to submit jobs for processing. The chapter begins with an example of running a simple job, and then continues with instructions for running more complex jobs.

Instructions for accomplishing the following tasks are included in this chapter.

Running a Simple Job

Use the information and instructions in this section to become familiar with basic procedures involved in submitting Sun Grid Engine 5.3 jobs.

Note – If you have installed the Sun Grid Engine program under an unprivileged account, you must log in as that particular user to be able to run jobs (see "Prerequisite Tasks" on page 24 for details).

▼ How To Run a Simple Job from the Command Line

Prior to executing any Sun Grid Engine command, you must first set your executable search path and other environmental conditions properly.

1. **Enter either of the the following commands, depending on your command interpreter.**

   a. If you are using either `csh` or `tcsh` as your command interpreter:

   ```
   % source sge_root_dir/default/common/settings.csh
   ```

   sge_root_dir specifies the location of the Sun Grid Engine root directory that was selected at the beginning of the installation procedure.

   b. If you are using `sh`, `ksh`, or `bash` as your command interpreter:

   ```
   # . sge_root_dir/default/common/settings.sh
   ```

   Note – You can add the above commands into your `.login`, `.cshrc`, or `.profile` files (whichever is appropriate) to guarantee proper Sun Grid Engine settings for all interactive session you will start later.

   ---

2. **Submit the following simple job script to your Sun Grid Engine cluster.**

   You can find the following job in the file, `examples/jobs/simple.sh` in your Sun Grid Engine root directory.

   ```
   #!/bin/sh
   #This is a simple example of a Sun Grid Engine batch script
   #
   # Print date and time
   date
   # Sleep for 20 seconds
   sleep 20
   # Print date and time again
   date
   # End of script file
   ```

Enter the following command, which assumes that `simple.sh` is the name of the script file in which the above script is stored, and the file is located in your current working directory.

```
% qsub simple.sh
```

The `qsub` command should confirm the successful job submission as follows.

```
your job 1 ("simple.sh") has been submitted
```

3. **Enter the following command to retrieve status information on your job.**

```
% qstat
```

You should receive a status report containing information about all jobs currently known to the Sun Grid Engine system and for each of them the so called *job ID* (the unique number being included in the submit confirmation), the name of the job script, the owner of the job, a state information (`r` means running), the submit or start time and eventually the name of the queue in which the job executes.

If no output is produced by the `qstat` command, no jobs are actually known to the system. For example, your job may already have finished. You can control the output of the finished jobs by checking their `stdout` and `stderr` redirection files. By default, these files are generated in the job owner's home directory on the host which has executed the job. The names of the files are composed of the job script file name, an appended dot sign followed by an "`o`" for the stdout file and an "`e`" for the stderr file and finally the unique job ID. Thus the `stdout` and `stderr` files of your job can be found under the names `simple.sh.o1` and `simple.sh.e1` respectively, if that job was the first ever executed in a newly installed Sun Grid Engine system.

▼ How To Submit Jobs From the Graphical User Interface, QMON

A more convenient method of submitting and controlling Sun Grid Engine jobs and of getting an overview of the Sun Grid Engine system is the graphical user interface, `QMON`. Among other facilities, `QMON` provides a job submission menu and a Job Control dialogue box for the tasks of submitting and monitoring jobs.

From the command line prompt, type the following command.

```
% qmon
```

During startup, a message window is displayed and then the QMON Main menu appears.

4. **Click the Job Control button and then the Submit Jobs button.**

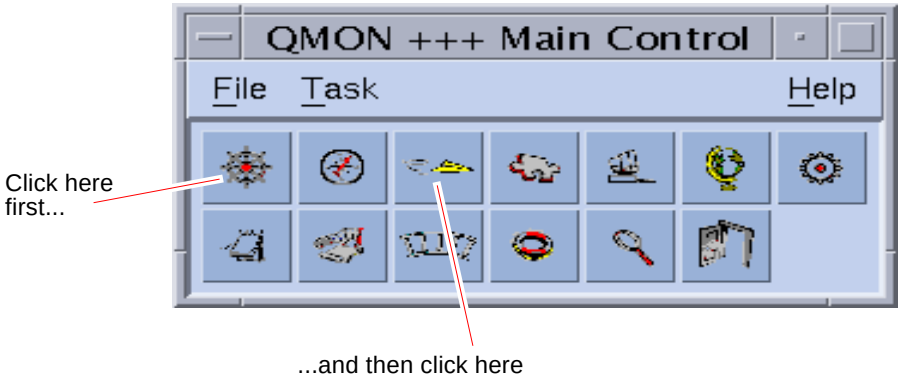


FIGURE 4-1 QMON Main Menu

The Job Submission and the Job Control dialogue boxes appear (see FIGURE 4-2 and FIGURE 4-3 respectively). The button names (such as Job Control) are displayed when you move the mouse pointer over the buttons.

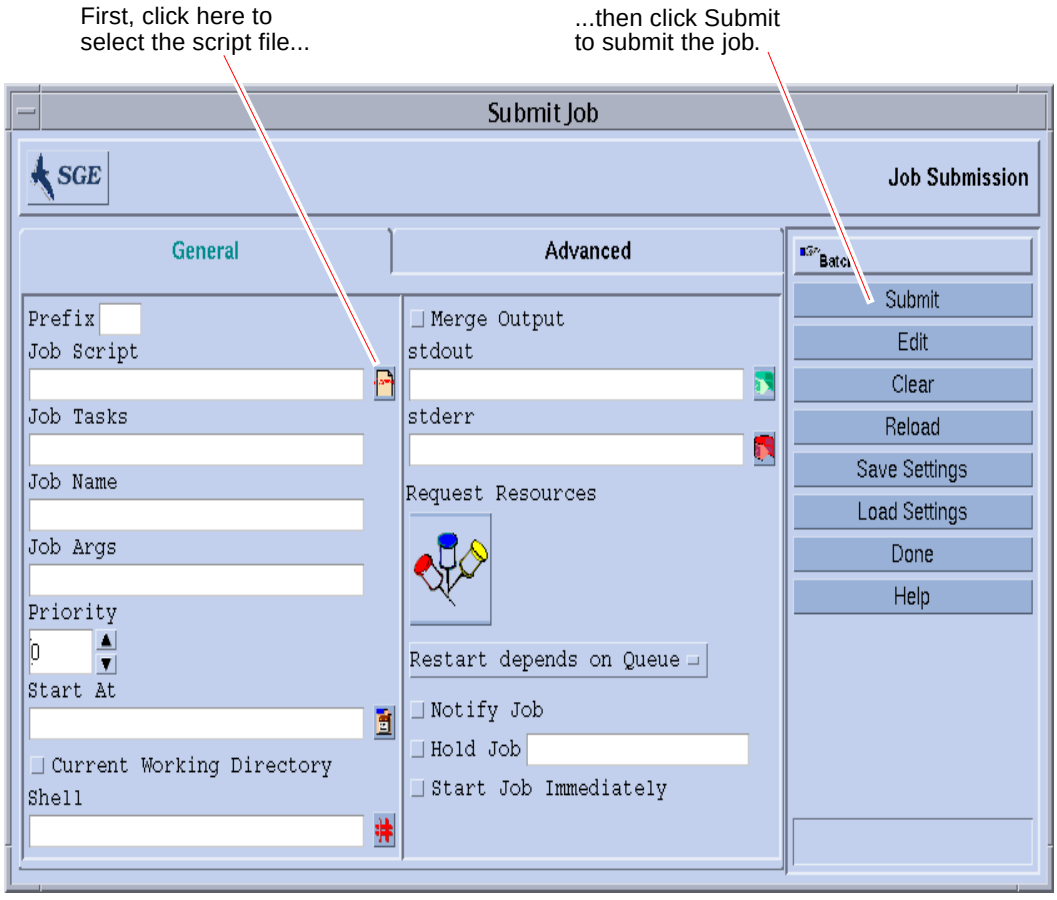


FIGURE 4-2 QMON Job Submission Dialogue Box

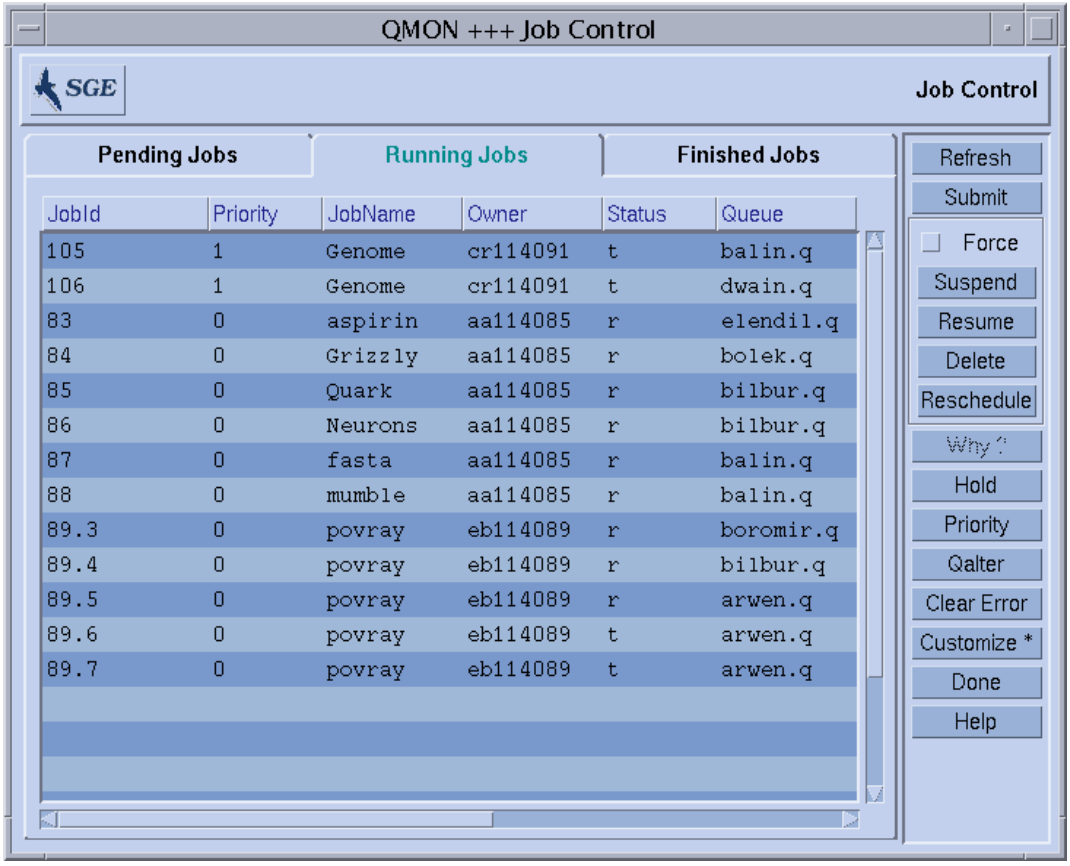


FIGURE 4-3 QMON Job Control Dialogue Box

5. **In the Job Submission menu, click the Job Script file selection icon to open a file selection box.**

6. **Click the appropriate file name to select your script file (e.g., the file *simple.sh* from the command line example).**

7. **Click the Submit button at the bottom of the Job Submission menu.**

   After a couple of seconds, you should be able to monitor your job in the Job Control panel. You will first see it under Pending Jobs, and it will quickly move to Running Jobs once it gets started.

Submitting Batch Jobs

The following sections describe how to submit more complex jobs through the Sun Grid Engine 5.3 program.

About Shell Scripts

Shell scripts, also called batch jobs, are in principal a sequence of command-line instructions assembled in a file. Script files are made executable by the `chmod` command. If scripts are invoked, a proper command interpreter is started (e.g., `csh, tcsh, sh`, or `ksh`) and each instruction is interpreted as typed in manually by the user executing the script. You can invoke arbitrary commands, applications, and other shell scripts from within a shell script.

The appropriate command interpreter is either invoked as `login-shell` or not, depending whether its name (`csh, tcsh, sh, ksh,`...) is contained in the value list of the `login_shells` entry of the Sun Grid Engine configuration in effect for the particular host and queue executing the job.

Note – The Sun Grid Engine configuration may be different for the various hosts and queues configured in your cluster. You can display the effective configurations via the `-sconf` and `-sq` options of the `qconf` command (refer to the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for detailed information).

If the command interpreter is invoked as `login-shell`, the environment of your job will be exactly the same as if you just have logged in and executed the script. In using `csh`, for example, `.login` and `.cshrc` will be executed in addition to the system default startup resource files (e.g., something like `/etc/login`) while only `.cshrc` will be executed if `csh` is not invoked as `login-shell`. Refer to the manual page of the command interpreter of your choice for a description of the difference between being invoked as `login-shell` or not.

Example of a Script File

CODE EXAMPLE 4-1 is an example of a simple shell script, which first compiles the application, `flow`, from its Fortran77 source and then executes it:.

```
#!/bin/csh

# This is a sample script file for compiling and
# running a sample FORTRAN program under Sun Grid Engine.

cd TEST

# Now we need to compile the program 'flow.f' and
# name the executable 'flow'.

f77 flow.f -o flow
```

CODE EXAMPLE 4-1 Simple Shell Script

Your local system user's guide will provide detailed information about building and customizing shell scripts (you might also want to look at the `sh`, `ksh`, `csh` or `tcsh` manual page). In the following sections, the emphasis is on specialities that are to be considered in order to prepare batch scripts for Sun Grid Engine.

In general, you can submit to Sun Grid Engine all shell scripts that you can execute from your command prompt by hand, as long as they do not require a terminal connection (except for the standard error and output devices, which are automatically redirected) and as long as they do not need interactive user intervention. Therefore, CODE EXAMPLE 4-1 is ready to be submitted to Sun Grid Engine and will perform the desired action.

Submitting Extended and Advanced Jobs with QMON

Before attempting a more complex form of job submission—*extended* or *advanced*—it is useful to understand some important background information about the process. The following sections provide that information.

Extended Example

The standard form of the Job Submission dialogue box (see FIGURE 4-2) provides the means to configure the following parameters for an extended job:

- A prefix string which is used for script-embedded Sun Grid Engine submit options (see the section, “Active Sun Grid Engine Comments” on page 90 for detailed information)
- The job script to be used

  If the associated file button is pushed, a file selection box is opened (see FIGURE 4-3)
- The task ID range for submitting array jobs (see “Array Jobs” on page 95)
- The name of the job (a default is set after a job script is selected)
- Arguments to the job script
- The job’s initial priority value

  Users without manager or operator permission may only lower their initial priority value.
- The time at which the job is to be considered eligible for execution

  If the associated file button is pushed, a helper dialogue box becomes available for entering the correctly formatted time is opened (see FIGURE 4-4)
- A flag indicating whether the job is to be executed in the current working directory (for identical directory hierarchies between the submit and the potential execution hosts only)
- The command interpreter to be used to execute the job script (see “How a Command Interpreter Is Selected” on page 89)

  If the associated button is pushed, a helper dialogue box becomes available for entering the command interpreter specifications of the job is opened (see FIGURE 4-5).
- A flag indicating whether the job’s standard output and standard error output are to be merged together into the standard output stream
- The standard output redirection to be used (see “Output Redirection” on page 89)

  A default is used if nothing is specified. If the associated file button is pushed, a helper dialogue box becomes available for entering the output redirection alternatives (“Output Redirection” on page 89).
- The standard error output redirection to be used—very similar to the standard output redirection
- The resource requirements of the job

  To define resource needs for your job, press the corresponding icon button. If resources have been requested for a job, the icon button changes its color.

- A selection list button defining whether the job can be restarted after being aborted by a system crash or similar events and whether the restart behavior depends on the queue or is demanded by the job
- A flag indicating whether the job is to be notified by `SIGUSR1` or `SIGUSR2` signals respectively if it is about to be suspended or cancelled
- A flag indicating that either a user hold or a job dependency is to be assigned to the job

  The job is not eligible for execution as long as any type of hold is assigned to it (see the section, “Monitoring and Controlling Sun Grid Engine Jobs” on page 117 for more information concerning holds). The input field attached to the Hold flag allows restricting the hold to only a specific range of task of an array job (see “Array Jobs” on page 95).
- A flag forcing the job to be either started immediately if possible or being rejected

  Jobs are not queued if this flag is selected.

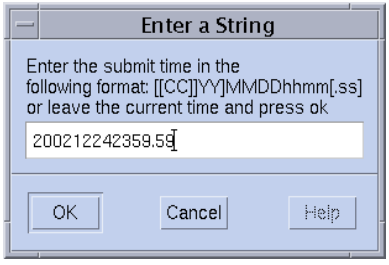


FIGURE 4-4 At Time Input Box

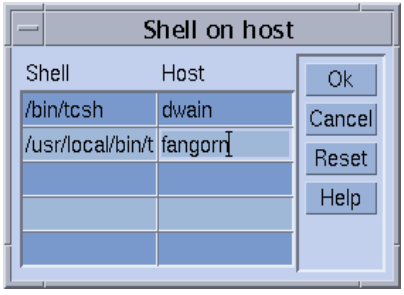


FIGURE 4-5 Shell Selection Box

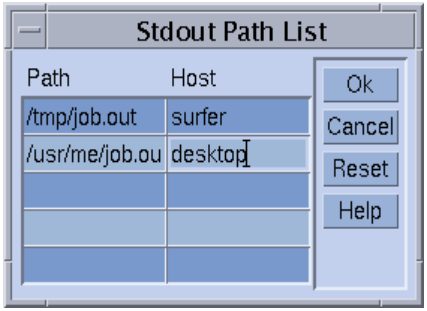


FIGURE 4-6 Output Redirection Box

The buttons at the right side of the Job Submission screen enable you to initiate various actions:

- **Submit** – Submit the job as specified in the dialogue box.
- **Edit** – Edit the selected script file in an X-terminal, either using `vi` or the editor as defined in the `$EDITOR` environment variable.
- **Clear** – Clear all settings in the Job Submission dialogue box, including any specified resource requests.
- **Reload** – Reload the specified script file, parse any script-embedded options (see the section, “Active Sun Grid Engine Comments” on page 90), parse default settings (see the section, “Default Requests” on page 94) and discard intermediate manual changes to these settings. This action is the equivalent to a Clear action with subsequent specifications of the previous script file. The option will only show an effect if a script file is already selected.
- **Save Settings** – Save the current settings to a file. A file selection box is opened to select the file. The saved files may either explicitly be loaded later (see below) or may be used as default requests (see the section, “Default Requests” on page 94).
- **Load Settings** – Load settings previously saved with the Save Settings button (see above). The loaded settings overwrite the current settings.
- **Done** – Closes the Job Submission dialogue box.
- **Help** – Display dialogue box-specific help.

FIGURE 4-7 shows the Job Submssion dialogue box with most of the parameters set.

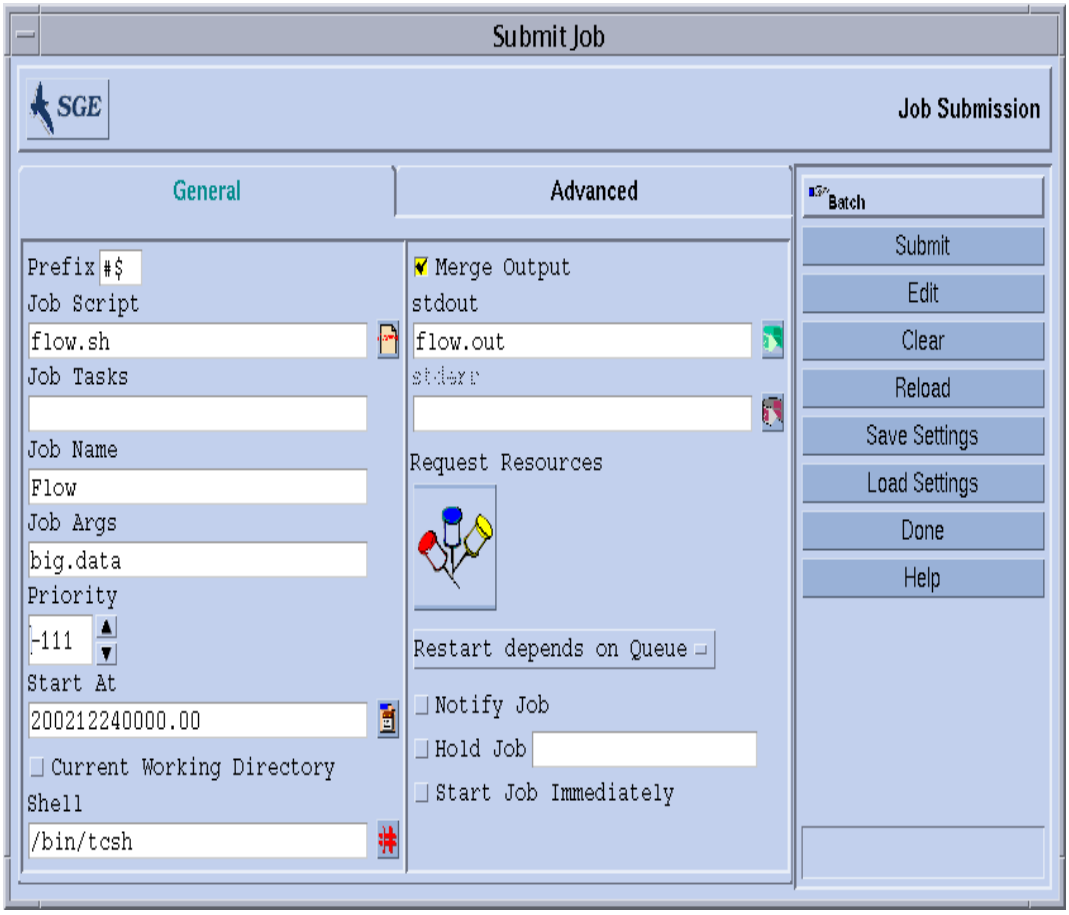


FIGURE 4-7 Extended Job Submission Example

The job configured in the example has the script file, `flow.sh`, which has to reside in the working directory of `QMON`. The job is called `Flow` and the script file takes the single argument, `big.data`. The job will be started with priority `-111` and is eligible for execution not before midnight of the 24th of December in the year 2002. The job will be executed in the submission working directory and will use the `tcsh` command interpreter. Finally, standard output and standard error output will be merged into the file, `flow.out`, which will be created in the current working directory also.

Advanced Example

The Advanced submission screen allows definition of the following additional parameters:

- A parallel environment interface to be used and the range of processes which is required (see the section, “Parallel Jobs” on page 101)
- A set of environment variables which are to be set for the job before it is executed

  If the associated icon button is pushed, a helper dialogue box becomes available for the definition of the environment variables to be exported (see FIGURE 4-8). Environment variables can be taken from `QMON`‘s runtime environment or arbitrary environment variable can be defined.
- A list of name/value pairs called Context (see FIGURE 4-9), which can be used to store and communicate job related information accessible anywhere from within a Sun Grid Engine cluster

  Context variables can be modified from the command line via the `-ac/-dc/-sc` options to `qsub,qrsh, qsh, qlogin`, or `qalter` and can be retrieved via `qstat -j`.
- The checkpointing environment to be used in case of a job for which checkpointing is desirable and suitable (see the section, “About Checkpointing Jobs” on page 111)
- An account string to be associated with the job

  The account string will be added to the accounting record kept for the job and can be used for later accounting analysis.
- The Verify flag, which determines the consistency checking mode for your job

  To check for consistency of the job request, Sun Grid Engine assumes an empty and unloaded cluster and tries to find at least one queue in which the job could run. Possible checking modes are:

  - **Skip** - No consistency checking at all.
  - **Warning** - Inconsistencies are reported, but the job is still accepted (may be desired if the cluster configuration is supposed to change after submission of the job).
  - **Error** - Inconsistencies are reported and the job will be rejected if any are encountered.
  - **Just verify** - The job will not be submitted, but an extensive report is generated about the suitability of the job for each host and queue in the cluster.
- The events about which the user is notified via electronic mail

  The events start/end/abortion/suspension of job are currently defined.
- A list of electronic mail addresses to which these notification mails are sent

If the associated button is pushed, a helper dialogue becomes available to define the mailing list (see FIGURE 4-10).

- A list of queue names which are requested to be the mandatory selection for the execution of the job.

  The Hard Queue List and the Soft Queue List are treated identically to a corresponding resource requirement as described in the bulleted list item, “The resource requirements of the job” on page 77.

- A list of queue names which are eligible as *master queue* for a parallel job.

  A parallel job is started in the master queue. All other queues to which the job spawns parallel tasks are called *slave queues.*

- An ID-list of jobs which need to be finished successfully before the job to be submitted can be started

  The newly created job *depends* on successful completion of those jobs.

FIGURE 4-8 Job Environment Definition

FIGURE 4-9 Job Context Definition

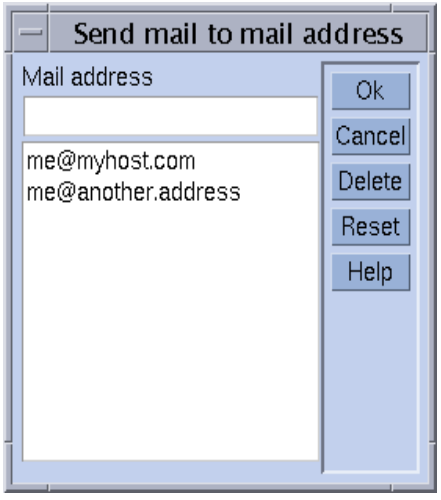


FIGURE 4-10 Mail Address Specification

The job defined in FIGURE 4-11 has the following additional characteristics as compared to the job definition from the section, “Extended Example” on page 77.

- The job requires the use of the parallel environment `mpi`. It needs at least 4 parallel processes to be created and can utilize up to 16 processes if available.
- Two environment variables are set and exported for the job.
- Two context variables are set.
- The account string `FLOW` is to be added to the job accounting record.
- The job is to be restarted if it fails in case of a system crash.
- Warnings should be printed if inconsistencies between the job request and the cluster configuration are detected
- Mail has to be sent to a list of two e-mail addresses as soon as the job starts and finishes.
- Preferably, the job should be executed in the queue `big_q`.

FIGURE 4-11 shows an example of an advanced job submission.

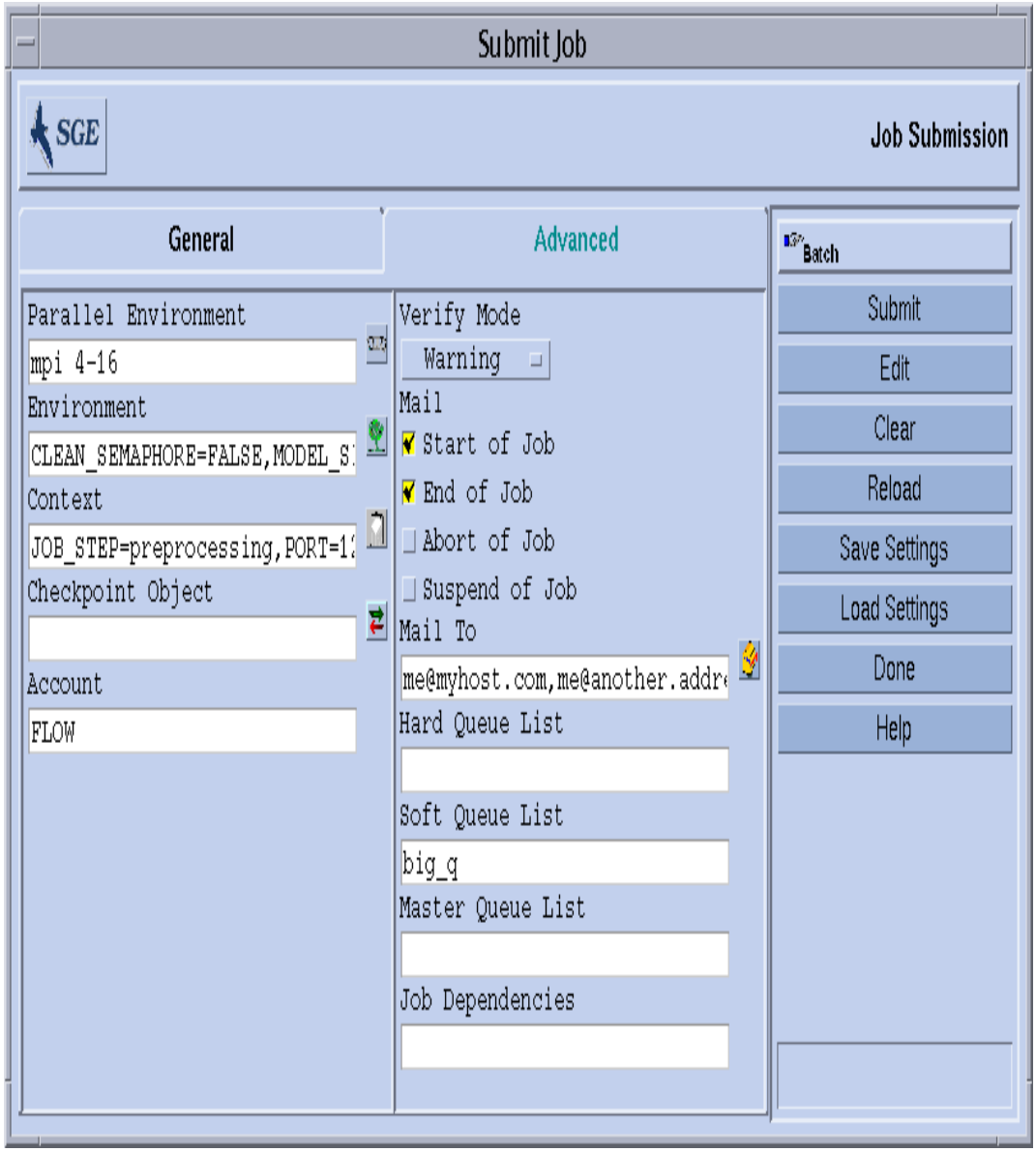


FIGURE 4-11 Advanced Job Submission Example

Resource Requirement Definition

In the examples so far the submit options used did not express any requirements for the hosts on which the jobs were to be executed. Sun Grid Engine assumes that such jobs can be run on any host. In practice, however, most jobs require certain prerequisites to be satisfied on the executing host in order to be able to complete successfully. Such prerequisites are enough available memory, required software to be installed or a certain operating system architecture. Also, the cluster administration usually imposes restrictions on the usage of the machines in the cluster. The CPU time allowed to be consumed by the jobs is often restricted, for example.

Sun Grid Engine provides the user with the means to find a suitable host for the user‘s job without a concise knowledge of the cluster‘s equipment and its utilization policies. All the user has to do is to specify the requirement of the user‘s jobs and let Sun Grid Engine manage the task of finding a suitable and lightly loaded host.

Resource requirements are specified via the *requestable attributes* explained in the section, “Requestable Attributes” on page 62. A very convenient way of specifying the requirements of a job is provided by `QMON`. The Requested Resources dialogue box, which is opened upon pressing the Requested Resources button in the Job Submission dialogue box (see FIGURE 4-12 for an example) only displays those attributes in the Available Resource selection list which currently are eligible. By double-clicking an attribute, the attribute is added to the Hard or Soft (see below) Resources list of the job and (except for `BOOLEAN` type attributes, which are just set to `True`) a helper dialogue box is opened to guide you in entering a value specification for the concerning attribute.

The example Requested Resources dialogue box displayed in FIGURE 4-12 shows a resource profile for a job in which a `solaris64` host with an available `permas` license offering at least 750 megabytes of memory is requested. If more than one queue fulfilling this specification is found, any defined soft resource requirements are taken into account (none in the example). However, if no queue satisfying both the hard and the soft requirements is found, any queue granting the hard requirements is considered to be suitable.

Note – Only if more than one queue is suitable for a job, load criteria determine where to start the job.

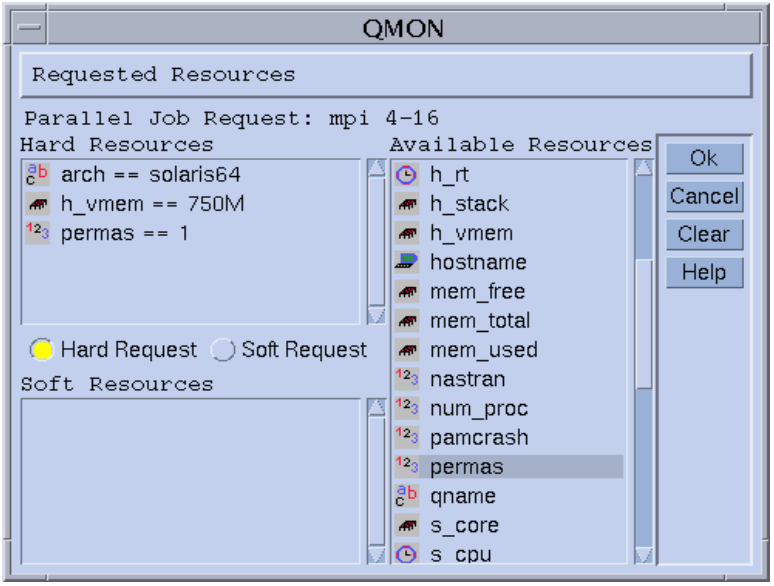


FIGURE 4-12 Requested Resources Dialogue Box

Note – The INTEGER attribute `permas` is introduced via an administrator extension to the “global” complex, the STRING attribute `arch` is imported from the “host” complex, while the MEMORY attribute `h_vmem` is imported from the “queue” complex.

An equivalent resource requirement profile can as well be submitted from the `qsub` command line:

```
% qsub -l arch=solaris64,h_vmem=750M,permas=1 \
permas.sh
```

Note – The implicit `-hard` switch before the first `-l` option has been skipped.

The notation *750M* for 750 Megabytes is an example for the Sun Grid Engine quantity syntax. For those attributes requesting a memory consumption you can specify either integer decimal, floating point decimal, integer octal and integer hexadecimal numbers appended by the so called multipliers:

- k – Multiplies the value by 1000.
- K – Multiplies the value by 1024.

- m – Multiplies the value by 1000 times 1000.
- M – Multiplies the value by 1024 times 1024.

Octal constants are specified by a leading 0 (zero) and digits ranging from 0 to 7 only. Specifying a hexadecimal constant requires to prepend the number by 0x and to use digits ranging from 0 to 9, a to f and A to F. If no multipliers are appended the values are considered to count as bytes. If using floating point decimals, the resulting value will be truncated to an integer value.

For those attributes imposing a time limit one can specify the time values in terms of hours, minutes or seconds and any combination. The hours, minutes and seconds are specified in decimal digits separated by colons. A time of `3:5:11` is translated to 11111 seconds. If a specifier for hours, minutes or seconds is 0 it can be left out if the colon remains. Thus a value of :5: is interpreted as 5 minutes. The form used in the Requested Resources dialogue box above is an extension, which is only valid within `QMON`.

How the Sun Grid Engine System Allocates Resources

As shown in the last section, it is important for you to know how Sun Grid Engine software processes resource requests and how it allocates resources. The following provides a schematic view of Sun Grid Engine software's resource allocation algorithm.

1. Read in and parse all default request files (see the section, "Default Requests" on page 94).
2. Process the script file for embedded options (see the section, "Active Sun Grid Engine Comments" on page 90).
3. Read all script embedding options when the job is submitted, regardless of their position in the script file.
4. Read and parse all requests from the command line.

As soon as all `qsub` requests are collected, *hard* and *soft* requests are processed separately (the hard first). The requests are evaluated, corresponding to the following order of precedence:

1. From left to right of the script / default request file
2. From top to bottom of the script / default request file
3. From left to right of the command line

In other words, the command line can be used to override the embedded flags.

The resources requested as hard are allocated. If a request is not valid, the submit is rejected. If one or more requests cannot be met at submit time (e.g., a requested queue is busy) the job is spooled and will be rescheduled at a later time. If all hard requests can be met, they are allocated and the job can be run.

The resources requested as soft are checked. The job can run even if some or all of these requests cannot be met. If multiple queues (already meeting the hard requests) provide parts of the soft resources list (overlapping or different parts) Sun Grid Engine software will select the queues offering the most soft requests.

The job will be started and will cover the allocated resources.

It is useful to gather some experience on how argument list options and embedded options or hard and soft requests influence each other by experimenting with small test script files executing UNIX commands such as `hostname` or `date`.

Extensions to Regular Shell Scripts

There are some extensions to regular shell scripts that will influence the behavior of the script if running under Sun Grid Engine control. The following sections describe these extensions.

How a Command Interpreter Is Selected

The command interpreter to be used to process the job script file can be specified at submit time (see, for example, FIGURE 4-7). However, if nothing is specified, the configuration variable, `shell_start_mode`, determines how the command interpreter is selected:

- If `shell_start_mode` is set to `unix_behavior`, the first line of the script file—if starting with a „#!“ sequence—is evaluated to determine the command interpreter. If the first line has no “`#!`“ sequence, the Bourne Shell `sh` is used by default.
- For all other settings of `shell_start_mode`, the default command interpreter as configured with the `shell` parameter for the queue in which the job is started is used (see the section, “Queues and Queue Properties” on page 56 and the `queue_conf` manual page).

Output Redirection

Since batch jobs do not have a terminal connection their standard output and their standard error output has to be redirected into files. Sun Grid Engine allows the user to define the location of the files to which the output is redirected, but uses defaults if nothing is specified.

The standard location for the files is in the current working directory where the jobs execute. The default standard output file name is <*Job_name*>.o<*Job_id*>, the default standard error output is redirected to <*Job_name*>.e<*Job_id*>. <*Job_name*> is either built from the script file name or can be defined by the user (see for example the `-N` option in the `qsub` manual page). <*Job_id*> is a unique identifier assigned to the job by Sun Grid Engine.

In case of array job tasks (see the section, "Array Jobs" on page 95), the task identifier is added to these filenames separated by a dot sign. Hence the resulting standard redirection paths are <*Job_name*>.o<*Job_id*>.<*Task_id*> and <*Job_name*>.e<*Job_id*>.<*Task_id*>.

In case the standard locations are not suitable, the user can specify output directions with `QMON` as shown in FIGURE 4-11 and FIGURE 4-6 or with the `-e` and `-o qsub` options. Standard output and standard error output can be merged into one file and the redirections can be specified on a per execution host basis. I.e., depending on the host on which the job is executed, the location of the output redirection files becomes different. To build custom but unique redirection file paths, pseudo environment variables are available which can be used together with the `qsub -e` and `-o` option. A list of these variables follows.

- `$HOME` – Home directory on execution machine
- `$USER` – User ID of job owner
- `$JOB_ID` – Current job ID
- `$JOB_NAME` – Current job name (see `-N` option)
- `$HOSTNAME` – Name of the execution host
- `$TASK_ID` – Array job task index number

These variables are expanded during runtime of the job into the actual values and the redirection path is built with them.

See the `qsub` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for further details.

Active Sun Grid Engine Comments

Lines with a leading "#" sign are treated as comments in shell scripts. Sun Grid Engine, however, recognizes special comment lines and uses them in a special way: the rest of such a script line will be treated as if it were part of the command line argument list of the Sun Grid Engine submit command `qsub`. The `qsub` options supplied within these special comment lines are also interpreted by the `QMON` Job Submission dialogue box and the corresponding parameters are preset when a script file is selected.

The special comment lines per default are identified by the "#$" prefix string. The prefix string can be redefined with the `qsub -C` option.

The described mechanism is called script embedding of submit arguments. The following is an example of a script file that makes use of script-embedded command line options.

```
#!/bin/csh
#Force csh if not Sun Grid Engine default shell
#$ -S /bin/csh
# This is a sample script file for compiling and
# running a sample FORTRAN program under Sun Grid Engine.
# We want Sun Grid Engine to send mail when the job begins
# and when it ends.
#$ -M EmailAddress
#$ -m b,e
# We want to name the file for the standard output
# and standard error.
#$ -o flow.out -j y
# Change to the directory where the files are located.
cd TEST
# Now we need to compile the program 'flow.f' and
# name the executable 'flow'.
f77 flow.f -o flow
# Once it is compiled, we can run the program.
flow
```

CODE EXAMPLE 4-2 Using Script-Embedded Command Line Options

Environment Variables

When a Sun Grid Engine job is run, a number of variables are preset into the job's environment, as listed below.

- ARC – The Sun Grid Engine architecture name of the node on which the job is running; the name is compiled-in into the sge_execd binary
- COMMD_PORT – Specifies the TCP port on which sge_commd(8) is expected to listen for communication requests
- SGE_ROOT – The Sun Grid Engine root directory as set for sge_execd before start-up or the default /usr/SGE
- SGE_CELL – The Sun Grid Engine cell in which the job executes
- SGE_JOB_SPOOL_DIR – The directory used by sge_shepherd(8) to store job-related data during job execution

- SGE_O_HOME – The home directory path of the job owner on the host from which the job was submitted
- SGE_O_HOST – The host from which the job was submitted
- SGE_O_LOGNAME – The login name of the job owner on the host from which the job was submitted
- SGE_O_MAIL – The content of the MAIL environment variable in the context of the job submission command
- SGE_O_PATH – The content of the PATH environment variable in the context of the job submission command
- SGE_O_SHELL – The content of the SHELL environment variable in the context of the job submission command
- SGE_O_TZ – The content of the TZ environment variable in the context of the job submission command
- SGE_O_WORKDIR – The working directory of the job submission command
- SGE_CKPT_ENV – Specifies the checkpointing environment (as selected with the qsub -ckpt option) under which a checkpointing job executes
- SGE_CKPT_DIR – Only set for checkpointing jobs; contains path ckpt_dir (see the checkpoint manual page) of the checkpoint interface
- SGE_STDERR_PATH – The path name of the file to which the standard error stream of the job is diverted; commonly used for enhancing the output with error messages from prolog, epilog, parallel environment start/stop or checkpointing scripts
- SGE_STDOUT_PATH – The path name of the file to which the standard output stream of the job is diverted; commonly used for enhancing the output with messages from prolog, epilog, parallel environment start/stop or checkpointing scripts
- SGE_TASK_ID – The task identifier in the array job represented by this task
- ENVIRONMENT – Always set to BATCH; this variable indicates that the script is run in batch mode
- HOME – The user's home directory path from the passwd file
- HOSTNAME – The host name of the node on which the job is running
- JOB_ID – A unique identifier assigned by the sge_qmaster when the job was submitted; the job ID is a decimal integer in the range to 99999
- JOB_NAME – The job name, built from the qsub *script filename*, a period, and the digits of the job ID; this default may be overwritten by qsub -N
- LOGNAME – The user's login name from the passwd file
- NHOSTS – The number of hosts in use by a parallel job
- NQUEUES – The number of queues allocated for the job (always 1 for serial jobs)
- NSLOTS – The number of queue slots in use by a parallel job

- PATH – A default shell search path of:
  /usr/local/bin:/usr/ucb:/bin:/usr/bin
- PE – The parallel environment under which the job executes (for parallel jobs only)
- PE_HOSTFILE – The path of a file containing the definition of the virtual parallel machine assigned to a parallel job by Sun Grid Engine

  See the description of the $pe_hostfile parameter in sge_pe for details on the format of this file. The environment variable is only available for parallel jobs.
- QUEUE – The name of the queue in which the job is running
- REQUEST – The request name of the job, which is either the job script file name or is explicitly assigned to the job via the qsub -N option
- RESTARTED – Indicates, whether a checkpointing job has been restarted; if set (to value 1), the job has been interrupted at least once and is thus restarted
- SHELL – The user's login shell from the passwd file

Note – This is not necessarily the shell in use for the job.

- TMPDIR – The absolute path to the job's temporary working directory
- TMP – The same as TMPDIR; provided for compatibility with NQS
- TZ – The time zone variable imported from sge_execd, if set
- USER – The user's login name from the passwd file.

▼ How To Submit Jobs from the Command Line

- **Enter the** qsub **command, along with appropriate arguments.**

For example, the simple job using the script file name, flow.sh—as described in the section, "How To Run a Simple Job from the Command Line" on page 70—could be submitted with the command:

```
% qsub flow.sh
```

To yield the equivalent result of the extended QMON job submission, however—as it is shown in FIGURE 4-7—would look as follows:

```
% qsub -N Flow -p -111 -a 200012240000.00 -cwd \
-S /bin/tcsh -o flow.out -j y flow.sh big.data
```

Further command line options can be added to constitute more complex requests. The advanced job request shown in FIGURE 4-11, for example, would look as follows:

```
% qsub -N Flow -p -111 -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
```

Default Requests

The last example in the above section demonstrates that advanced job requests may become rather complex and unhandy, in particular if similar requests need to be submitted frequently. To avoid the cumbersome and error prone task of entering such command-lines, the user can either embed `qsub` options in the script files (see “Active Sun Grid Engine Comments” on page 90) or can utilize so called *default requests*.

The cluster administration may setup a default request file for all Sun Grid Engine users. The user, on the other hand, can create a private default request file located in the user‘s home directory as well as application specific default request files located in the working directories.

Default request files simply contain the `qsub` options to be applied by default to the Sun Grid Engine jobs in a single or multiple lines. The location of the cluster global default request file is <`sge_`*root*>/<*cell*>/`common/sge_request`. The private general default request file is located under `$HOME/.sge_request`, while the application specific default request files are expected under `$cwd/.sge_request`.

If more than one of these files is available, they are merged into one default request with the following order of precedence:

1. Global default request file.
2. General private default request file.
3. Application-specific default request file.

Note – Script embedding and the `qsub` command line has higher precedence than the default request files. Thus, script embedding overwrites default request file settings, and the `qsub` command line options my overwrite these settings again.

Note – The `qsub -clear` option can be used at any time in a default request file, in embedded script commands and in the `qsub` command line to discard any previous settings.

An example of a private default request file is presented below.

```
-A myproject -cwd -M me@myhost.com -m b,e
-r y -j y -S /bin/ksh
```

Unless overwritten, for all jobs of the given user the account string would be *myproject*, the jobs would execute in the current working directory, mail notification would be sent at the beginning and end of the jobs to *me@myhost.com*, the jobs are to be restarted after system crashes, the standard output and standard error output are to be merged and the `ksh` is to be used as command interpreter.

Array Jobs

Parametrized and repeated execution of the same set of operations (contained in a job script) is an ideal application for the Sun Grid Engine *array job* facility. Typical examples for such applications are found in the Digital Content Creation industries for tasks such as rendering. Computation of an animation is split into frames, in this example, and the same rendering computation can be performed for each frame independently.

The array job facility offers a convenient way to submit, monitor and control such applications. Sun Grid Engine, on the other hand, provides an efficient implementation of array jobs, handling the computations as an array of independent tasks joined into a single job. The tasks of an array job are referenced through an array index number. The indices for all tasks span an index range for the entire array job which is defined during submission of the array job by a single `qsub` command.

An array job can be monitored and controlled (e.g., suspended, resumed, or cancelled) as a total or by individual task or subset of tasks, in which case the corresponding index numbers are suffixed to the job ID to reference the tasks. As tasks are executed (very much like regular jobs), they can use the environment variable `$SGE_TASK_ID` to retrieve their own task index number and to access input data sets designated for this task identifier.

▼ How To Submit an Array Job from the Command Line

- **Enter the `qsub` command with appropriate arguments.**

The following is an example of submitting an array job.

```
% qsub -l h_cpu=0:45:0 -t 2-10:2 render.sh data.in
```

The `-t` option defines the task index range. In this case, *2-10:2* specifies that *2* is the lowest and *10* is the highest index number while only every second index (the *:2* part of the specification) is used. Thus the array job consists of 5 tasks with the task indices 2, 4, 6, 8, and 10. Each task requests a hard CPU time limit of *45* minutes (the `-l` option) and will execute the job script *render.sh* once being dispatched and started by Sun Grid Engine. The tasks can use `$SGE_TASK_ID` to find out whether they are task 2, 4, 6, 8, or 10 and they can use their index number to find their input data record in the data file *data.in*.

▼ How To Submit an Array Job with `QMON`

- **Follow the instructions in "How To Submit Jobs From the Graphical User Interface, `QMON`" on page 71, additionally taking into account the following notes.**

Note – The submission of array jobs from `QMON` works virtually identically to how it was described in "How To Submit Jobs From the Graphical User Interface, `QMON`" on page 71. The only difference is that the Job Tasks input window shown in FIGURE 4-7 needs to contain the task range specification with the identical syntax as for the `qsub -t` option. Please refer to the `qsub` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for detailed information on the array index syntax.

The sections "Monitoring and Controlling Sun Grid Engine Jobs" on page 117 and "Controlling Sun Grid Engine Jobs from the Command Line" on page 130, as well as the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* sections about `qstat`, `qhold`, `qrls`, `qmod`, and `qdel`, contain the pertinent information about monitoring and controlling Sun Grid Engine jobs in general and array jobs in particular.

Note – Array jobs offer full access to all Sun Grid Engine facilities known for regular jobs. In particular they can be parallel jobs at the same time or can have interdependencies with other jobs.

Submitting Interactive Jobs

Submitting interactive jobs instead of batch jobs is useful in situations where your job requires your direct input to influence the results of the job. This is typically the case for X-windows applications, which are interactive by definition, or for tasks in which your interpretation of immediate results is required to steer the further computation.

Three methods exist in Sun Grid Engine system to create interactive job.

- `qlogin` – This is a telnet-like session that is started on a host selected by Sun Grid Engine software.
- `qrsh` – This is the equivalent of the standard UNIX `rsh` facility. Either a command is executed remotely on a host selected by the Sun Grid Engine system, or a remote rlogin (`rlogin`) session is started on a remote host if no command was specified for execution.
- `qsh` – This is an `xterm` that is brought up from the machine executing the job with the display set corresponding to your specification or the setting of the `DISPLAY` environment variable. If the `DISPLAY` variable is not set and if no display destination was defined specifically, Sun Grid Engine directs the `xterm` to the 0.0 screen of the X server on the host from which the interactive job was submitted.

Note – To function correctly, all the facilities need proper configuration of Sun Grid Engine cluster parameters. The correct `xterm` execution paths have to be defined for qsh and interactive queues have to be available for this type of jobs. Contact your system administrator whether your cluster is prepared for interactive job execution.

The default handling of interactive jobs differs from the handling of batch jobs in that interactive jobs are not queued if they cannot be executed by the time of submission. This is to indicate immediately, that not enough appropriate resources are available to dispatch an interactive job right after it was submitted. The user is notified in such cases that the Sun Grid Engine cluster is too busy currently.

This default behavior can be changed with the `-now no` option to `qsh`, `qlogin` and `qrsh`. If this option is given, interactive jobs are queued like batch jobs. Using `-now yes`, batch jobs submitted with `qsub` also can be handled like interactive jobs and are either dispatched for execution immediately or are rejected.

Note – Interactive jobs can only be executed in queues of the type INTERACTIVE (refer to “About Configuring Queues” on page 163 for details).

The subsequent sections outline the usage of the `qlogin` and `qsh` facilities. The `qrsh` command is explained in a broader context in the section, “Transparent Remote Execution” on page 103.

Submitting Interactive Jobs with QMON

The only type of interactive jobs that can be submitted from `QMON` are those bringing up an `xterm` on a host selected by Sun Grid Engine.

▼ How To Submit Interactive Jobs with QMON

- **Click the icon on top of the button column at the right side of the Job Submission dialogue box until the Interactive icon is displayed.**

  This prepares the Job Submission dialogue box to submit interactive jobs (see FIGURE 4-13 and FIGURE 4-14).

  The meaning and the usage of the selection options in the dialogue box is the same as explained for batch jobs in the section, “Submitting Batch Jobs” on page 75. The basic difference is that several input fields are set insensitive because they do not apply for interactive jobs

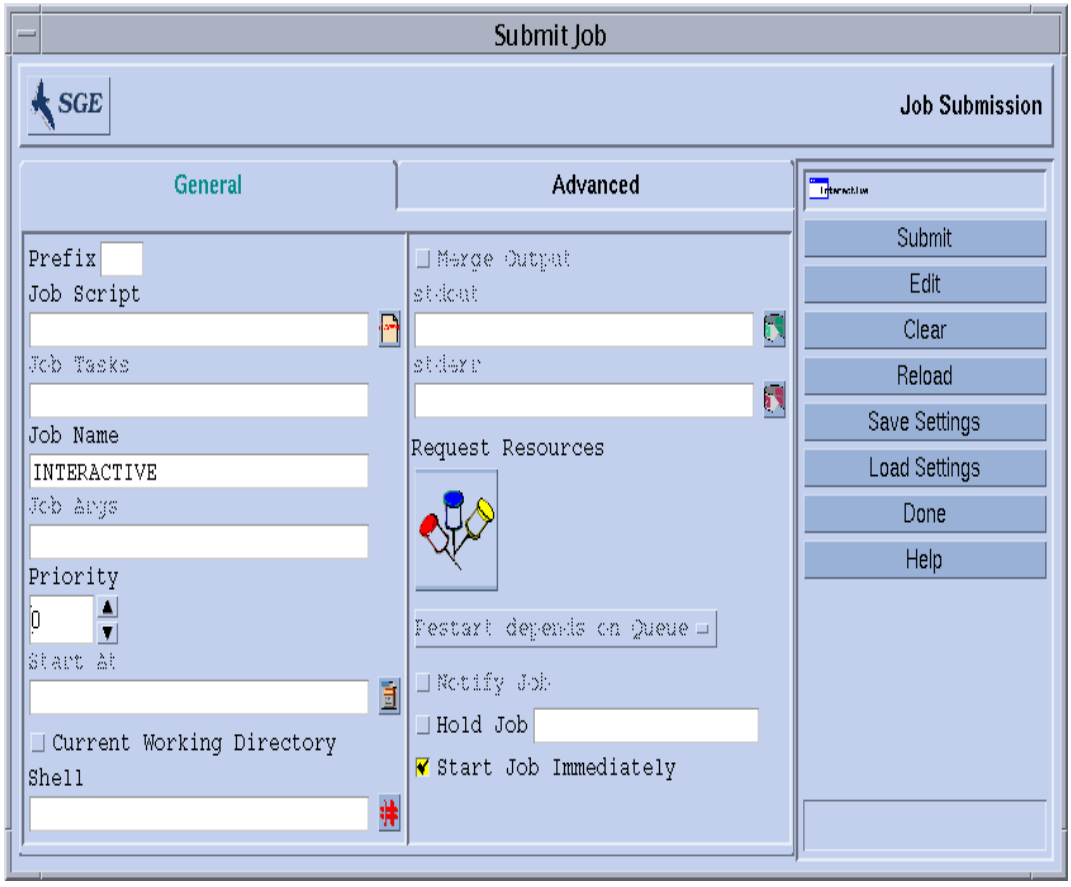


FIGURE 4-13 Interactive Job Submission Dialogue Box, General

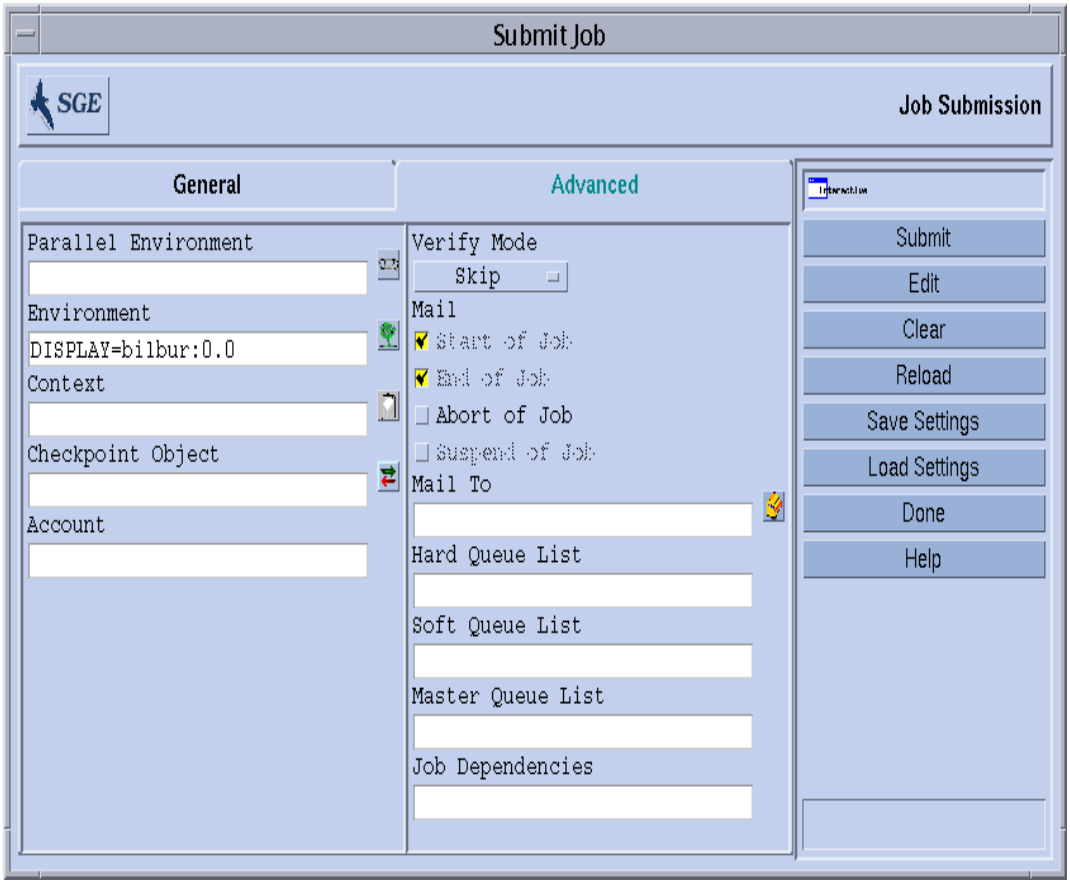


FIGURE 4-14 Interactive Job Submission Dialogue Box, Advanced

Submitting Interactive Jobs with `qsh`

`Qsh` is very similar to `qsub` and supports several of the `qsub` options, as well as the additional switch `-display` to direct the display of the `xterm` to be invoked (refer to

the qsh entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for details).

▼ How To Submit Interactive Jobs With qsh

- **Enter the following command to start an xterm on any available Sun Solaris 64bit operating system host.**

```
% qsh -l arch=solaris64
```

Submitting Interactive Jobs with qlogin

The qlogin command can be used from any terminal or terminal emulation to initiate an interactive session under the control of Sun Grid Engine.

▼ How To Submit Interactive Jobs With qlogin

- **Enter the following command to locate a low-loaded host with Star-CD license available and with at least one queue providing a minimum of 6 hours hard CPU time limit.**

```
% qlogin -l star-cd=1,h_cpu=6:0:0
```

Note – Depending on the remote login facility configured to be used by the Sun Grid Engine system, you may have to enter your user name, your password, or both at a login prompt.

Parallel Jobs

Sun Grid Engine provides means to execute parallel jobs using arbitrary message passing environments such as PVM or MPI (see the *PVM User's Guide* and the *MPI User's Guide* for details) or shared memory parallel programs on multiple slots in single queues or distributed across multiple queues and (for distributed memory

parallel jobs) across machines. An arbitrary number of different *parallel environment* (PE) interfaces may be configured concurrently at the same time. See Chapter 10, “Managing Parallel Environments” on page 245 for details about PEs.

How Sun Grid Engine Jobs Are Scheduled

Essentially, Sun Grid Engine 5.3 software uses two set of criteria to schedule jobs:

- Job Priorities
- Equal-share

Job Priorities

Concerning the order of scheduling precedence of different jobs, a *first-in-first-out* (fifo) rule is applied by default. All *pending* (not yet scheduled) jobs are inserted in a list, with the first submitted job being the head of the list, followed by the second submitted job, and so on. The job submitted first will be attempted to be scheduled first. If at least one suitable queue is available, the job will be scheduled. Sun Grid Engine software will try to schedule the second job afterwards no matter whether the first has been dispatched or not.

This order of precedence among the pending jobs may be overruled by the cluster administration via a *priority value* being assigned to the jobs. The actual priority value can be displayed by using the `qstat` command (the priority value is contained in the last column of the pending jobs display entitled `P`; refer to the section, “How To Monitor Jobs with `qstat`” on page 127 for details). The default priority value assigned to the jobs at submit time is 0. The priority values are positive and negative integers and the pending jobs list is sorted Correspondingly in the order of descending priority values. By assigning a relatively high priority value to a job, the job is moved to the top of the pending jobs list. Jobs with negative priority values are inserted even after jobs just submitted. If there are several jobs with the same priority value, the fifo rule is applied within that priority value category.

Equal-Share-Scheduling

The fifo rule sometimes leads to problems, especially if user’s tend to submit a series of jobs almost at the same time (e.g., via shell-script issuing one submit after the other). All jobs being submitted afterwards and being designated to the same group of queues will have to wait a very long time. *Equal-share-scheduling* avoids this problem by sorting jobs of users already owning a running job to the end of the precedence list. The sorting is performed only among jobs within the same priority

value category. Equal-share-scheduling is activated if the Sun Grid Engine scheduler configuration entry `user_sort` (refer to the `sched_conf` manual page for details) is set to `TRUE`.

Queue Selection

The Sun Grid Engine system does not dispatch jobs requesting nonspecific queues if they cannot be started immediately. Such jobs will be marked as spooled at the `sge_qmaster`, which will try to re-schedule them from time to time. Thus, such jobs are dispatched to the next suitable queue that becomes available.

As opposed to this, jobs that are requested by name to a certain queue will go directly to this queue, regardless of whether they can be started or they have to be spooled. Therefore, viewing Sun Grid Engine queues as computer science *batch queues* is only valid for jobs requested by name. Jobs submitted with nonspecific requests use the spooling mechanism of `sge_qmaster` for queueing, thus utilizing a more abstract and flexible queuing concept.

If a job is scheduled and multiple free queues meet its resource requests, the job is usually dispatched to the queue (among the suitable) belonging to the least loaded host. By setting the Sun Grid Engine scheduler configuration entry `queue_sort_method` to `seq_no`, the cluster administration may change this load dependent scheme into a fixed order algorithm: the queue configuration entry `seq_no` is used to define a precedence among the queues assigning the highest priority to the queue with the lowest sequence number.

Transparent Remote Execution

Sun Grid Engine provides a set of closely related facilities supporting transparent remote execution of certain computational tasks. The core tool for this functionality is the `qrsh` command described in section “Remote Execution with `qrsh`” on page 104. Building on top of `qrsh`, two high level facilities—`qtcsh` and `qmake`—allow the transparent distribution of implicit computational tasks via Sun Grid Engine, thereby enhancing the standard UNIX facilities `make` and `csh`. `qtcsh` is explained in the section, “Transparent Job Distribution with `qtcsh`” on page 105 and `qmake` is described in the section, “Parallel Makefile Processing with `qmake`” on page 107.

Remote Execution with qrsh

Qrsh is built around the standard rsh facility (see the information provided in <sge_*root*>/3rd_party for details on the involvement of rsh) and can be used for various purposes.

- To provide remote execution of interactive applications via Sun Grid Engine comparable to the standard UNIX facility, rsh (also called remsh for HP-UX).
- To offer interactive login session capabilities via Sun Grid Engine similar to the standard UNIX facility, rlogin (note that qlogin is still required as a Sun Grid Engine representation of the UNIX telnet facility).
- To allow for the submission of batch jobs which, upon execution, support terminal I/O (standard/error output and standard input) and terminal control.
- To offer a means for submitting a standalone program not embedded in a shell-script.
- To provide a batch job submission client which remains active while the job is pending or executing and which only finishes if the job has completed or has been cancelled.
- To allow for the Sun Grid Engine system-controlled remote execution of job tasks (such as the concurrent tasks of a parallel job) within the framework of the dispersed resources allocated by parallel jobs (see the section, "Tight Integration of PEs and Sun Grid Engine Software" on page 255).

By virtue of all these capabilities, qrsh is the major enabling infrastructure for the implementation of the qtcsh and the qmake facilities as well as for the so called tight integration of Sun Grid Engine with parallel environments such as MPI or PVM.

qrsh Usage

The general form of the qrsh command is:

```
% qrsh [options] program|shell-script [arguments] \
       [> stdout_file] [>&2 stderr_file] [< stdin_file]
```

qrsh understands almost all options of qsub and provides only a few additional ones.

- -now yes|no – This option controls whether the job is scheduled immediately and rejected if no appropriate resources are available, as usually desired for an interactive job—hence it is the default—or whether the job is queued like a batch job, if it cannot be started at submission time.

- `-inherit` – `qrsh` does not go through the Sun Grid Engine scheduling process to start a job-task, but it assumes that it is embedded inside the context of a parallel job which already has allocated suitable resources on the designated remote execution host. This form of `qrsh` commonly is used within `qmake` and within a tight parallel environment integration. The default is not to inherit external job resources.
- `-verbose` – This option presents output on the scheduling process. It is mainly intended for debugging purposes and therefore switched off per default.

Transparent Job Distribution with `qtcsh`

`qtcsh` is a fully compatible replacement for the widely known and used UNIX C-Shell (`csh`) derivative `tcsh` (`qmake` is built around `tcsh` - see the information provided in <`sge_`*root*>/`3rd_party` for details on the involvement of `tcsh`). It provides a command-shell with the extension of transparently distributing execution of designated applications to suitable and lightly loaded hosts via Sun Grid Engine. Which applications are to be executed remotely and which requirements apply for the selection of an execution host is defined in configuration files called `.qtask`.

Transparent to the user, such applications are submitted for execution to Sun Grid Engine via the `qrsh` facility. Since `qrsh` provides standard output, error output and standard input handling as well as terminal control connection to the remotely executing application, there are only three noticeable differences between executing such an application remotely as opposed to executing it on the same host as the shell.

- The remote host may be much better suited (more powerful, lower loaded, required hard/software resources installed) than the local host, which may not allow execution of the application at all. This is a desired difference, of course.
- There will be a small delay incurred by the remote startup of the jobs and by their handling through Sun Grid Engine.
- Administrators can restrict the usage of resources through interactive jobs (`qrsh`) and thus through `qtcsh`. If not enough suitable resources are available for an application to be started via the `qrsh` facility or if all suitable systems are overloaded, the implicit `qrsh` submission will fail and a corresponding error message will be returned (`Not enough resources ... try later`).

In addition to the *standard* use, `qtcsh` is a suitable platform for third party code and tool integration. Using `qtcsh` in its single-application execution form `qtcsh -c` *appl_name* inside integration environments presents a persistent interface that almost never has to be changed. All the required application, tool, integration, site and even user-specific configurations are contained in appropriately defined `.qtask` files. A further advantage is that this interface can be used from within shell scripts of any type, C programs and even Java applications.

qtcsh Usage

Invocation of qtcsh is exactly the same as for tcsh. Qtcsh extends tcsh in providing support for the .qtask file and by offering a set of specialized shell built-in modes.

The .qtask file is defined as follows. Each line in the file has the following format:

```
% [!]appl_name qrsh_options
```

The optional leading exclamation mark (!) defines the precedence between conflicting definitions in a cluster global .qtask file and the personal .qtask file of the qtcsh user. If the exclamation mark is missing in the cluster global file, an eventually conflicting definition in the user file will overrule. If the exclamation mark is in the cluster global file, the corresponding definition cannot be overwritten.

The rest of the line specifies the name of the application which, when typed on a command line in a qtcsh, will be submitted to Sun Grid Engine for remote execution, and the options to the qrsh facility, which will be used and which define resource requirements for the application.

Note – The application name must appear in the command line exactly like defined in the .qtask file. If it is prefixed with an absolute or relative directory specification it is assumed that a local binary is addressed and no remote execution is intended.

Note – Csh aliases, however, are expanded before a comparison with the application names is performed. The applications intended for remote execution can also appear anywhere in a qtcsh command line, in particular before or after standard I/O redirections.

Hence, the following examples are valid and meaningful syntax:

```
# .qtask file
netscape -v DISPLAY=myhost:0
grep -l h=filesurfer
```

Given this `.qtask` file, the following `qtcsh` command lines:

```
netscape
~/mybin/netscape
cat very_big_file | grep pattern | sort | uniq
```

will implicitly result in:

```
qrsh -v DISPLAY=myhost:0 netscape
~/mybin/netscape
cat very_big_file | qrsh -l h=filesurfer grep pattern | sort | uniq
```

`qtcsh` can operate in different modes influenced by switches where each of them can be on or off:

- Local or remote execution of commands (remote is default)
- Immediate or batch remote execution (immediate is default)
- Verbose or non-verbose output (non-verbose is default)

The setting of these modes can be changed using option arguments of `qtcsh` at start time or with the shell builtin command `qrshmode` at runtime. See the `qtcsh` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for more information.

Parallel Makefile Processing with `qmake`

`qmake` is a replacement for the standard UNIX `make` facility. It extends `make` by its ability to distribute independent `make` steps across a cluster of suitable machines. `qmake` is built around the popular GNU-make facility, `gmake`. See the information provided in <`sge_`*root*>/`3rd_party` for details on the involvement of `gmake`.

To ensure that a complex distributed `make` process can run to completion, `qmake` first allocates the required resources in an analogous form like a parallel job. `Qmake` then manages this set of resources without further interaction with the Sun Grid Engine scheduling. It distributes `make` steps as resources are or become available via the `qrsh` facility with the `-inherit` option enabled.

Since `qrsh` provides standard output, error output and standard input handling as well as terminal control connection to the remotely executing `make` step, there are only three noticeable differences between executing a `make` procedure locally or using `qmake`:

- Provided that the individual `make` steps have a certain duration and that there are enough independent `make` steps to be processed, the parallelization of the `make` process will be sped up significantly. This is a desired difference, of course.
- In the `make` steps to be started up remotely, there will be an implied small overhead caused by `qrsh` and the remote execution as such.
- To take advantage of the `make` step distribution of `qmake`, the user has to specify as a minimum the degree of parallelization; i.e., the number of concurrently executable `make` steps. In addition, the user can specify the resource characteristics required by the `make` steps, such as available software licenses, machine architecture, memory or CPU-time requirements.

The most common use in general of `make` certainly is the compilation of complex software packages. This may not be the major application for `qmake`, however. Program files are often quite small (as a matter of good programming practice) and hence compilation of a single program file, which is a single `make` step, often only takes a few seconds. Furthermore, compilation usually implies a lot of file access (nested include files) which may not be accelerated if done for multiple `make` steps in parallel, because the file server can become the bottleneck effectively serializing all the file access. So a satisfactory speed-up of the compilation process sometimes cannot be expected.

Other potential applications of `qmake` are more appropriate. An example is the steering of the interdependencies and the workflow of complex analysis tasks through make-files. This is common in some areas, such as EDA, and each `make` step in such environments typically is a simulation or data analysis operation with non-negligible resource and computation time requirements. A considerable speed-up can be achieved in such cases.

`qmake` Usage

The command-line syntax of `qmake` looks very similar to the one of `qrsh`:

```
% qmake [-pe pe_name pe_range][further options] \
        -- [gnu-make-options][target]
```

Note – The `-inherit` option is also supported by `qmake` as described later in this section.

Specific attention has to be paid on the usage of the `-pe` option and its relation to the `gmake -j` option. Both options can be used to express the amount of parallelism to be achieved. The difference is that `gmake` provides no possibility with `-j` to specify something like a parallel environment to use. Hence, `qmake` makes the assumption,

that a default environment for parallel makes is configured which is called `make`. Furthermore, `gmake´s -j` allows no specification of a range, but only for a single number. `Qmake` will interpret the number given with `-j` as a range of `1-`*<given_number>*. As opposed to this, `-pe` permits the detailed specification of all these parameters. Consequently, the following command line examples are identical.

```
% qmake -- -j 10
% qmake -pe make 1-10 --
```

While the following command lines cannot be expressed via the `-j` option:.

```
% qmake -pe make 5-10,16 --
% qmake -pe mpi 1-99999 --
```

Apart from the syntax, `qmake` supports two modes of invocation: interactively from the command-line (without `-inherit`) or within a batch job (with `-inherit`). These two modes initiate a different sequence of actions:

- **Interactive** – When `qmake` is invoked on the command-line, the `make` process as such is implicitly submitted to Sun Grid Engine via `qrsh` taking the resource requirements specified in the `qmake` command-line into account. Sun Grid Engine then selects a *master machine* for the execution of the parallel job associated with the parallel `make` job and starts the `make` procedure there. This is necessary, because the `make` process can be architecture dependent and the required architecure is specified in the `qmake` command-line. The `qmake` process on the master machine then delegates execution of individual `make` steps to the other hosts which have been allocated by Sun Grid Engine for the job and which are passed to `qmake` via the parallel environment hosts file.
- **Batch** – In this case, `qmake` appears inside a batch script with the `-inherit` option (if the `-inherit` option was not present, a new job would be spawned as described for the first case above). This results in `qmake` making use of the resources already allocated to the job into which `qmake` is embedded. It will use `qrsh -inherit` directly to start `make` steps. When calling `qmake` in batch mode, the specification of resource requirements or `-pe` and `-j` options is ignored.

Note – Also single CPU jobs have to request a parallel environment (`qmake -pe make 1 --`). If no parallel execution is required, call `qmake` with `gmake` command-line syntax (without Sun Grid Engine options and "`--`"), it will behave like `gmake`.

Refer to the `qmake` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for further detail on `qmake`.

CHAPTER 5

Checkpointing, Monitoring, and Controlling Jobs

After you have submitted jobs by way of the Sun Grid Engine 5.3 system, you need to be able to monitor and control them. This chapter provides both background information about, and instructions for, accomplishing these tasks.

Included in this chapter are instructions for the following specific taks.

About Checkpointing Jobs

This section explores two different types of job checkpointing.

- *User-level*
- *Kernel-level*

User-Level Checkpointing

Many application programs, especially those that normally consume considerable CPU time, have implemented checkpointing and restart mechanisms to increase fault tolerance. Status information and important parts of the processed data are repeatedly written to one or more files at certain stages of the algorithm. These files (called restart files) can be processed if the application is aborted and restarted at a later time and a consistent state can be reached, comparable to the situation just before the checkpoint. As the user mostly has to deal with the restart files in order to move them to a proper location, this kind of checkpointing is called *user-level* checkpointing.

For application programs that do not have an integrated (user-level) checkpointing, an alternative can be to use a so-called *checkpointing library* which can be provided by the public domain (see the *Condor* project of the University of Wisconsin, for example) or by some hardware vendors. Relinking an application with such a library installs a checkpointing mechanism in the application without requiring source code changes.

Kernel-Level Checkpointing

Some operating systems provide checkpointing support inside the operating system kernel. No preparations in the application programs and no re-linking of the application is necessary in this case. Kernel-level checkpointing is usually applicable for single processes as well as for complete process hierarchies. I.e., a hierarchy of interdependent processes can be checkpointed and restarted at any time. Usually both, a user command and a C-library interface are available to initiate a checkpoint.

Sun Grid Engine supports operating system checkpointing if available. Please refer to the Sun Grid Engine Release Notes for information on the currently supported kernel-level checkpointing facilities.

Migration of Checkpointing Jobs

Checkpointing jobs are interruptible at any time, since their restart capability ensures that only few work already done must be repeated. This ability is used to build Sun Grid Engine's migration and dynamic load balancing mechanism. If requested, checkpointing Sun Grid Engine jobs are aborted on demand and migrated to other machines in the Sun Grid Engine pool thus averaging the load in the cluster in a dynamic fashion. Checkpointing jobs are aborted and migrated for the following reasons.

- The executing queue or the job is suspended explicitly by a `qmod` or `qmon` command.

- The executing queue or the job is suspended automatically because a suspend threshold for the queue has been exceeded (see the section, “How To Configure Load and Suspend Thresholds” on page 197) and the checkpoint occasion specification for the job includes the suspension case (see the section, “How To Submit, Monitor, or Delete a Checkpointing Job from the Command Line” on page 114).

You can identify a job that is about to migrate by the state `m` for migrating in the `qstat` output. A migrating job moves back to `sge_qmaster` and is subsequently dispatched to another suitable queue if any is available.

Composing a Checkpointing Job Script

Shell scripts for kernel-level checkpointing show no difference from regular shell scripts.

Shell scripts for user-level checkpointing jobs differ from regular Sun Grid Engine batch scripts only in their ability to properly handle the case if they get restarted. The environment variable, `RESTARTED` is set for checkpointing jobs which are restarted. It can be used to skip over sections of the job script which should be executed during the initial invocation only.

Thus, a transparently checkpointing job script may look similar to CODE EXAMPLE 5-1.

```
#!/bin/sh
#Force /bin/sh in Sun Grid Engine
#$ -S /bin/sh

# Test if restarted/migrated
if [ $RESTARTED = 0 ]; then
     # 0 = not restarted
     # Parts to be executed only during the first
     # start go in here
     set_up_grid
fi

# Start the checkpointing executable
fem
#End of scriptfile
```

CODE EXAMPLE 5-1 Example of Checkpointing Job Script

It is important to note that the job script is restarted from the beginning if a user-level checkpointing job is migrated. The user is responsible for directing the program flow of the shell-script to the location where the job was interrupted and thus skipping those lines in the script which are critical to be executed more than once.

Note – Kernel-level checkpointing jobs are interruptible at any point of time and also the embracing shell script is restarted exactly from the point where the last checkpoint occurred. Therefore, the `RESTARTED` environment variable is of no relevance for kernel-level checkpointing jobs.

▼ How To Submit, Monitor, or Delete a Checkpointing Job from the Command Line

Enter the following command with the appropriate switches.

```
#qsub options arguments
```

Submitting a checkpointing job works the same way as for regular batch scripts, except for the `qsub -ckpt` and `-c` switches, which request a checkpointing mechanism and define the occasions at which checkpoints have to be generated for the job. The `-ckpt` option takes one argument which is the name of the checkpointing environment ("About Checkpointing Support" on page 238) to be used. The `-c` option is not mandatory and also takes one argument. It can be used to overwrite the definitions of the `when` parameter in the checkpointing environment configuration (see the `checkpoint` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for details).

The argument to the `-c` option can be one of the following one-letter selection (or any combination thereof) or a time value alternatively:

- `n` – No checkpoint is performed. This has highest precedence
- `s` – A checkpoint is only generated if the `sge_execd` on the jobs host is shut down.
- `m` – Generate checkpoint at minimum CPU interval defined in the corresponding queue configuration (see the `min_cpu_interval` parameter in the `queue_conf` manual page).
- `x` – A checkpoint is generated if the job gets suspended.

- `interval` – Generate checkpoint in the given interval but not more frequently than defined by `min_cpu_interval` (see above). The time value has to be specified as hh:mm:ss (two digit hours, minutes and seconds separated by colon signs).

The *monitoring* of checkpointing jobs just differs from regular jobs by the fact, that these jobs may migrate from time to time (signified by state `m` for migrating in the output of `qstat`, see above) and, therefore, are not bound to a single queue. However, the unique job identification number stays the same as well as the job name.

Deleting checkpointing jobs works just the same way as described in section "Controlling Sun Grid Engine Jobs from the Command Line" on page 130.

▼ How To Submit a Checkpointing Job with QMON

- **Follow the instructions in "Advanced Example" on page 81, taking note of the following additional information.**

Submission of checkpointing jobs via `QMON` is identical to the submission of regular batch jobs with the addition of specifying an appropriate checkpointing environment. As explained in the procedure, "Advanced Example" on page 81, the Job Submission dialogue box provides an input window for the checkpointing environment associated with a job. Aside to the input window there is an icon button, which opens the Selection dialogue box displayed in FIGURE 5-1. You can select a suitable checkpoint environment from the list of available ones with it. Ask your system administrator for information about the properties of the checkpointing environments installed at your site, or refer to the section, "About Checkpointing Support" on page 238.

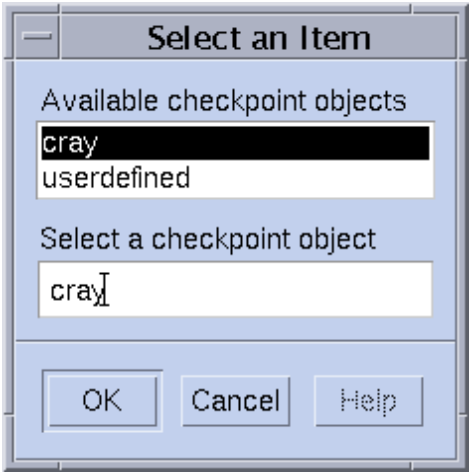


FIGURE 5-1 Checkpoint Object Selection

File System Requirements

When a checkpointing library based user-level or kernel-level checkpoint is written, a complete image of the virtual memory the process or job to be checkpointed covers needs to be dumped. Sufficient disk space must be available for this purpose. If the checkpointing environment configuration parameter `ckpt_dir` is set the checkpoint information is dumped to a job private location under `ckpt_dir`. If `ckpt_dir` is set to `NONE`, the directory in which the checkpointing job was started is used. Refer to the `checkpoint` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for detailed information about the checkpointing environment configuration.

Note – You should start a checkpointing job with the `qsub -cwd` script if `ckpt_dir` is set to `NONE`.

An additional requirement concerning the way how the file systems are organized is caused by the fact, that the checkpointing files and the restart files must be visible on all machines in order to successfully migrate and restart jobs. Thus NFS or a similar file system is required. Ask your cluster administration, if this requirement is met for your site.

If your site does not run NFS or if it is not desirable to use it for some reason, you should be able to transfer the restart files explicitly at the beginning of your shell script (e.g. via `rcp` or `ftp`) in the case of user-level checkpointing jobs.

Monitoring and Controlling Sun Grid Engine Jobs

In principle, there are three ways to monitor submitted jobs.

- With the Sun Grid Engine graphical user's interface, `QMON`
- From the command line with the `qstat` command
- By electronic mail

▼ How To Monitor and Control Jobs with `QMON`

The Sun Grid Engine graphical user's interface, `QMON`, provides a dialogue box specifically designed for controlling jobs.

- **In the `QMON` Main menu, press the Job Control button, then proceed according to the additional information detailed in the following sections.**

The general purpose of this dialogue box is to provide the means to monitor all running, pending and a configurable number of finished jobs known to the system or parts thereof. The dialogue box can also be used to manipulate jobs, i.e. to change their priority, to suspend, resume and to cancel them. Three list environments are displayed, one for the running jobs, another for the pending jobs waiting to be dispatched to an appropriate resource and the third for recently finished jobs. You can select between the three list environments via clicking to the corresponding tab labels at the top of the screen.

In its default form (see FIGURE 5-2) it displays the columns JobId, Priority, JobName and Queue for each running and pending job.

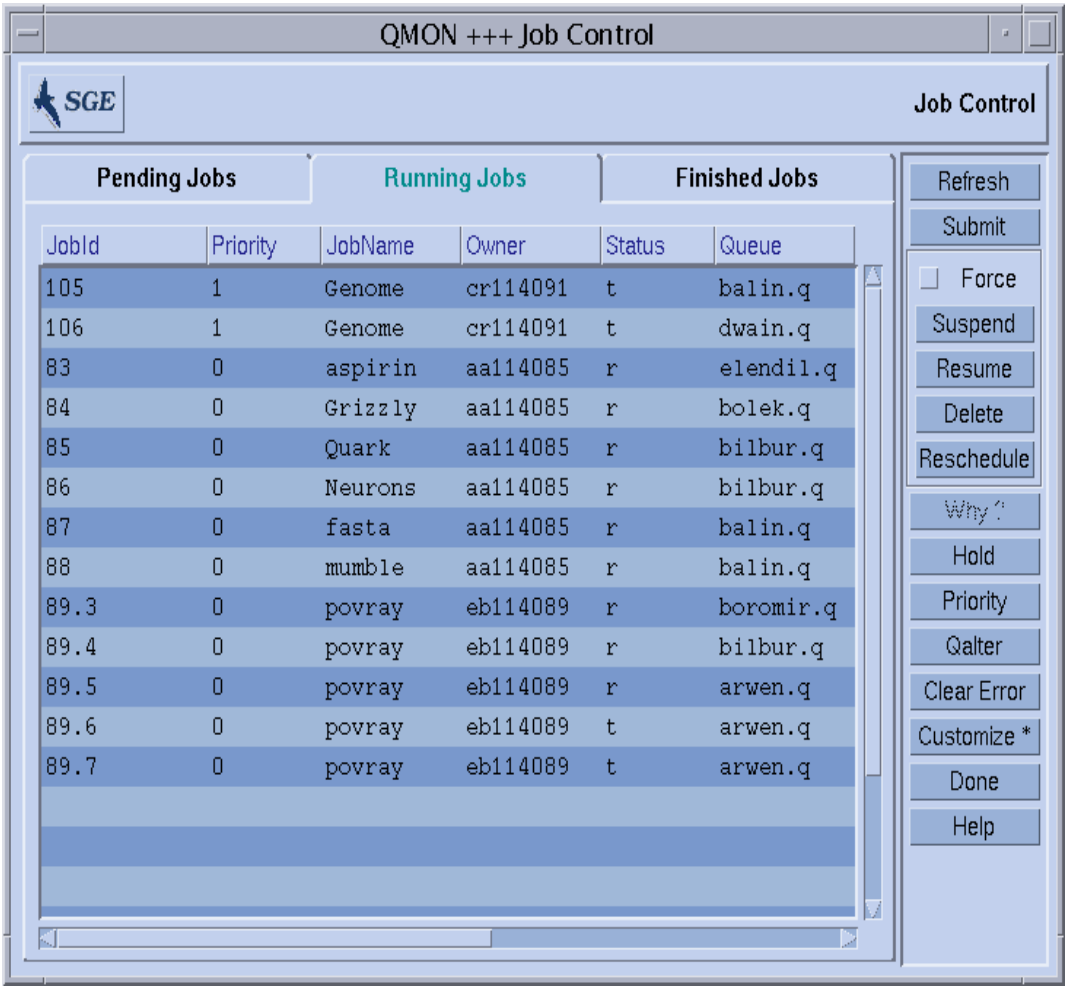


FIGURE 5-2 Job Control Dialogue Box—Standard Form

You can configure the set of information displayed with a Customization dialogue box, (see FIGURE 5-3), which is opened upon pushing the Customize button in the Job Control dialogue box.

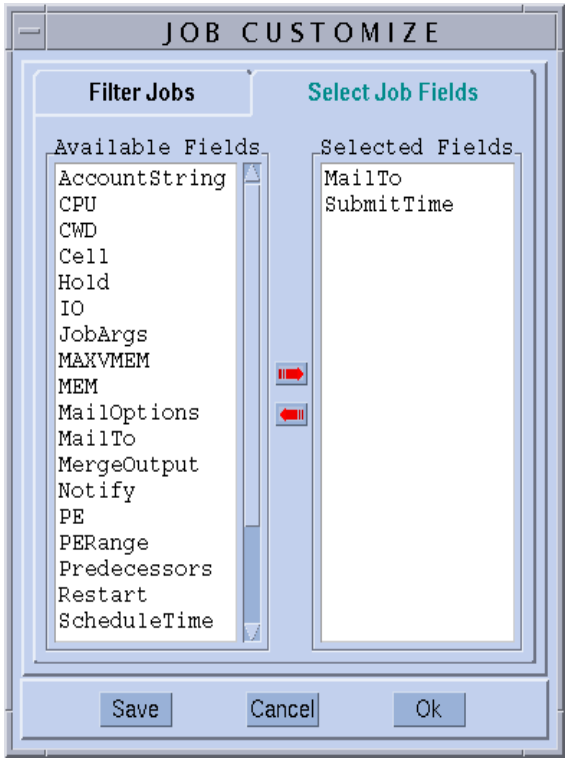


FIGURE 5-3 Job Control Customization Dialogue Box

With the Customization dialogue box it is possible to select further entries of the Sun Grid Engine job object to be displayed and to filter the jobs of interest. The example in FIGURE 5-3 selects the additional fields, MailTo and Submit Time.

The Job Control dialogue box displayed in FIGURE 5-4 depicts the enhanced look after the customization has been applied in case of the Finished Jobs list.

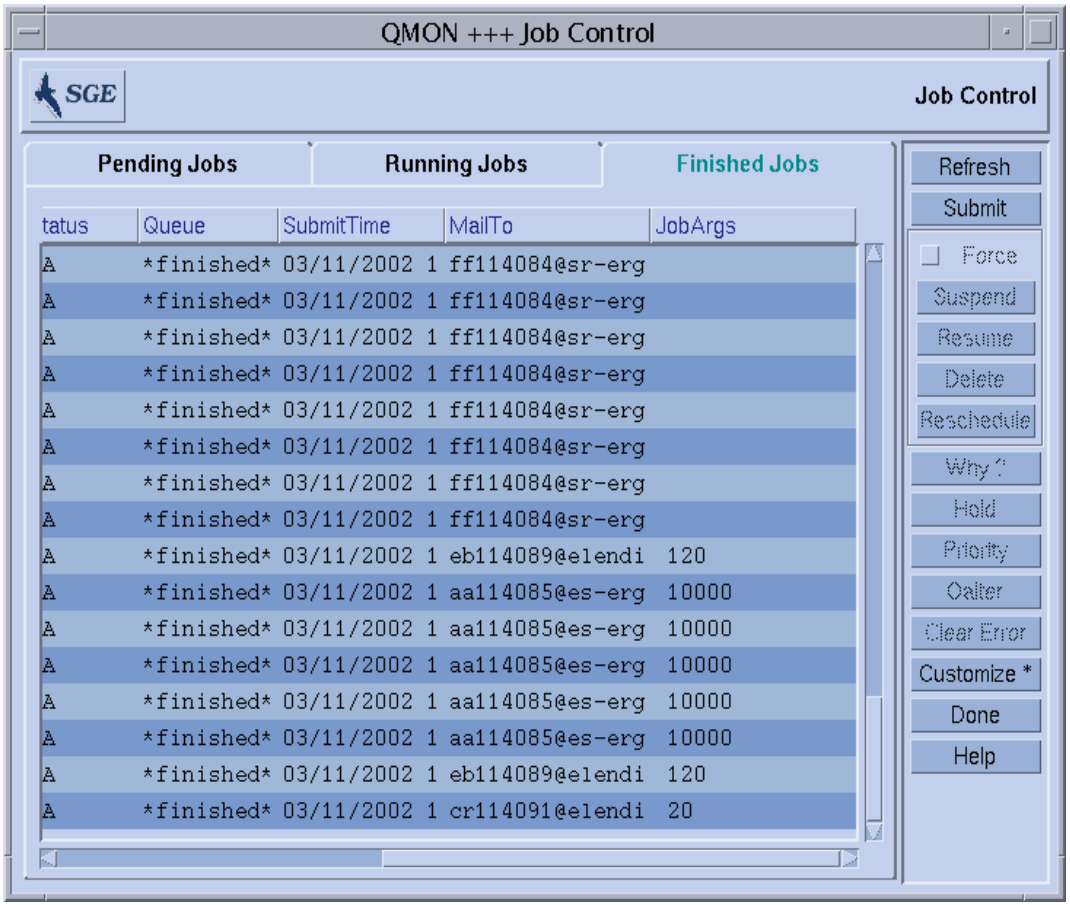


FIGURE 5-4 Job Control Dialogue Box Finished Jobs—Enhanced

The example of the filtering facility in FIGURE 5-5 selects only those jobs owned by `ferstl` which run or are suitable for architecture solaris64.

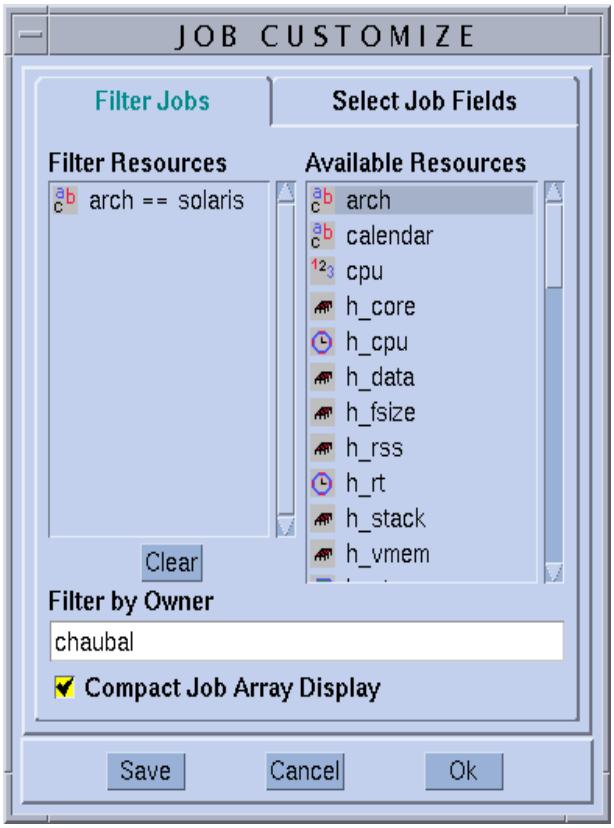


FIGURE 5-5 Job Control Filtering

The resulting Job Control dialogue box showing Pending Jobs is displayed in FIGURE 5-6.

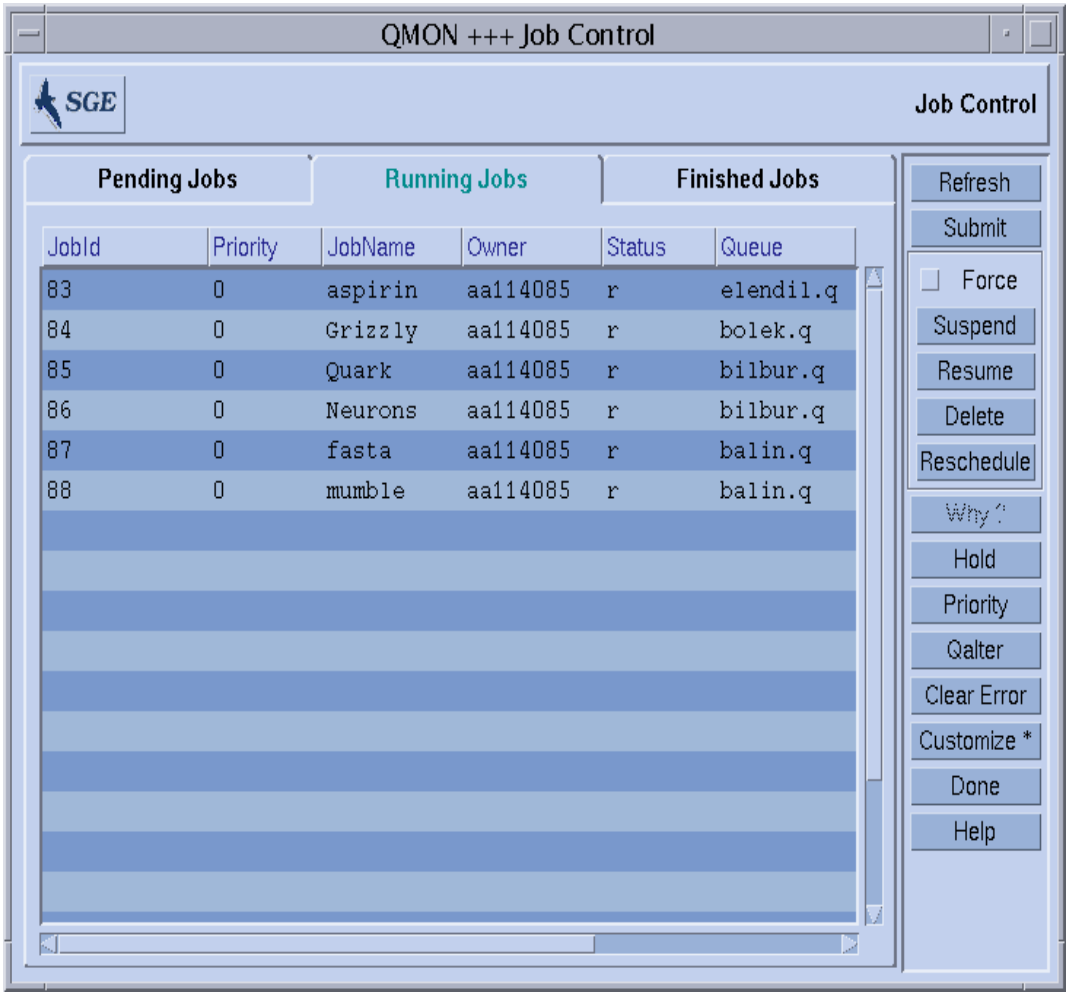


FIGURE 5-6 Job Control Dialogue Box—After Filtering

Note – The Save button displayed in the Customization dialogue box in FIGURE 5-3, for example, stores the customizations into the file `.qmon_preferences` in the user's home directory and thus redefines the default appearance of the Job Control dialogue box.

The Job Control dialogue box in FIGURE 5-6 is also an example of how array jobs are displayed in `QMON`.

Jobs can be selected (for later operation) with the following mouse/key combinations:

- Clicking on a job with the left mouse button while the Control key is pressed starts a selection of multiple jobs.
- Clicking on another job with the left mouse button while the Shift key is pressed selects all jobs in between and including the job at the selection start and the current job.
- Clicking on a job with the left mouse button while the Control key is pressed toggles the selection state of a single job.

The selected jobs can be suspended, resumed (unsuspended), deleted, held back (and released), re-prioritized and modified (`Qalter`) through the corresponding buttons at the right side of the screen.

The actions suspend, unsuspend, delete, hold, modify priority and modify job may only be applied to a job by the job owner or by Sun Grid Engine managers and operators (see “Managers, Operators and Owners” on page 67). Only running jobs can be suspended/resumed and only pending jobs can be held back and modified (in priority as well as in other attributes).

Suspending a job means the equivalent to sending the signal, `SIGSTOP`, to the process group of the job with the UNIX `kill` command, which halts the job and no longer consumes CPU time. Unsuspending the job sends the signal, SIGCONT, thereby resuming the job (see the `kill` manual page of your system for more information on signalling processes).

Note – Suspension, unsuspension and deletion can be forced; i.e., registered with `sge_qmaster` without notification of the `sge_execd` controlling the job(s), in case the corresponding `sge_execd` is unreachable—for example, due to network problems. Use the `Force` flag for this purpose.

If using the `Hold` button on a selected pending job, the Set Hold sub-dialogue box is opened (see FIGURE 5-7).

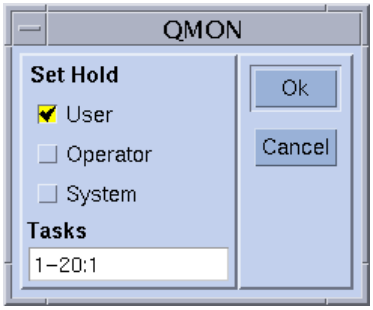


FIGURE 5-7 Job Control Holds

The Set Hold sub-dialogue box enables setting and resetting user, system, and operator holds. User holds can be set/reset by the job owner as well as Sun Grid Engine operators and managers. Operator holds can be set/reset by managers and operator and manager holds can be set/reset by managers only. As long as any hold is assigned to a job it is not eligible for execution. Alternate ways to set/reset holds are the `qalter, qhold` and `qrls` commands (see the corresponding entries in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual*).

If the Priority button is pressed, another sub-dialogue box is opened (FIGURE 5-8), which enables entering the new priority of the selected pending jobs. In Sun Grid Engine, the priority determines the order of the jobs in the pending jobs list and the order in which the pending jobs are displayed by the `Job Control` dialogue. Users can only set the priority in the range between 0 and -1024. Sun Grid Engine operators and managers can also increase the priority level up to the maximum of 1023 (see the section, “Job Priorities” on page 102 for details about job priorities).

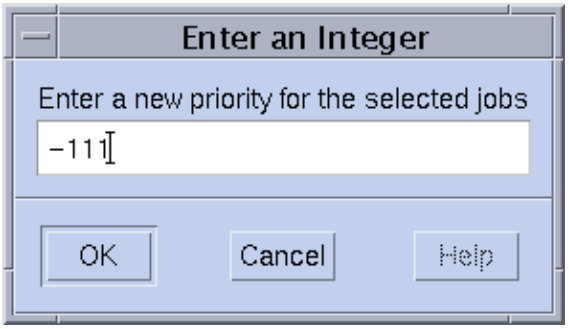


FIGURE 5-8 Job Control Priority Definition

The Qalter button, when pressed for a pending job, opens the Job Submission screen described in “How To Submit Jobs From the Graphical User Interface, QMON” on page 71 with all the entries of the dialogue box set corresponding to the attributes of the job as defined during submission. Those entries, which cannot be changed, are set insensitive. The others may be edited and the changes are registered with Sun Grid Engine by pushing the Qalter button (a replacement for the Submit button) in the Job Submission dialogue box.

The Verify flag in the Job Submission screen has a special meaning when used in the Qalter mode. You can check pending jobs for their consistency and investigate why they have not been scheduled yet. You just have to select the desired consistency checking mode for the Verify flag and push the Qalter button. The system will display warnings on inconsistencies depending on the selected checking mode. Refer to the section, “Advanced Example” on page 81 and the `-w` option in the `qalter` manual page for further information.

Another method for checking why jobs are still pending is to select a job and click on the Why? button of the Job Control dialogue box. This will open the Object Browser dialogue box and display a list of reasons which prevented the Sun Grid Engine scheduler from dispatching the job in its most recent pass. An example browser screen displaying such a message is shown in FIGURE 5-9.

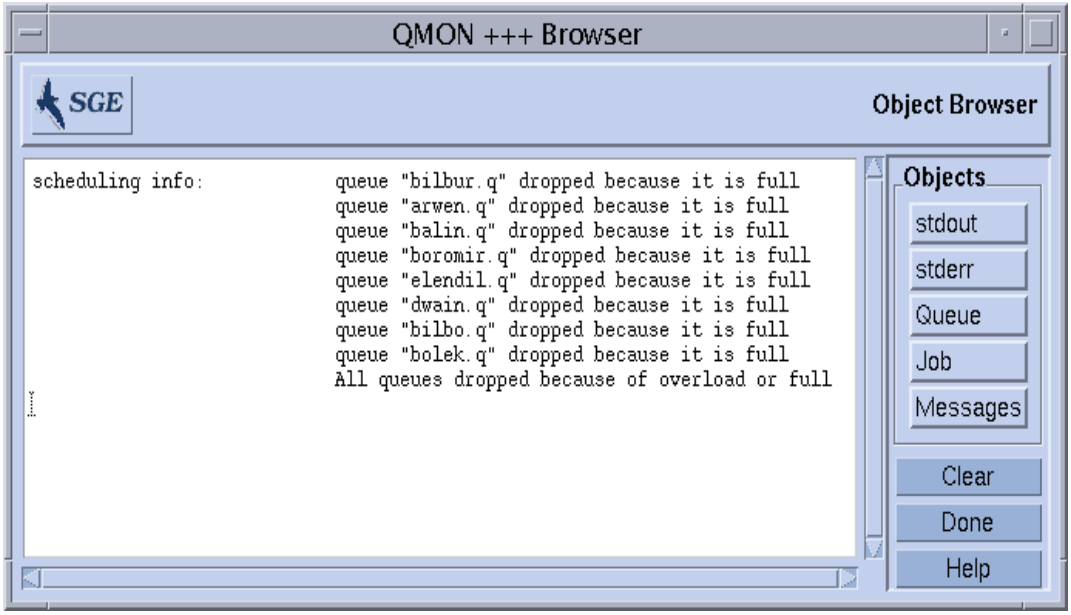


FIGURE 5-9 Browser Displaying Scheduling Information

Note – The Why? button only delivers meaningful output if the scheduler configuration parameter `schedd_job_info` is set to `true` (see `sched_conf` in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual*). The displayed scheduler information relates to the last scheduling interval. It may not be accurate anymore by the time you investigate for reasons why your job has not been scheduled.

The Clear Error button can be used to remove an error state from a selected pending job, which had been started in an earlier attempt, but failed due to a job dependent problem (e.g., insufficient permissions to write to the specified job output file).

Note – Error states are displayed using a red font in the pending jobs list and should only be removed after correcting the error condition; e.g., via `qalter`. Such error conditions are automatically reported via electronic mail, if the job requests to send e-mail in case it is aborted (e.g., via the `qsub -m a` option).

To keep the information being displayed up-to-date, QMON uses a polling scheme to retrieve the status of the jobs from sge_qmaster. An update can be forced by pressing the Refresh button.

Finally, the button provides a link to the QMON Job Submission dialogue box (see FIGURE 5-10, for example).

Additional Information with the QMON Object Browser

The QMON Object Browser can be used to quickly retrieve additional information on Sun Grid Engine jobs without a need to customize the Job Control dialogue box, as explained in section “How To Monitor and Control Jobs with QMON” on page 117.

The Object Browser is opened upon pushing the Browser icon button in the QMON main menu. The browser displays information about Sun Grid Engine jobs if the Job button in the browser is selected and if the mouse pointer is moved over a job‘s line in the Job Control dialogue box (see FIGURE 5-2 for example).

The browser screen in FIGURE 5-10 gives an example of the information displayed in such a situation.

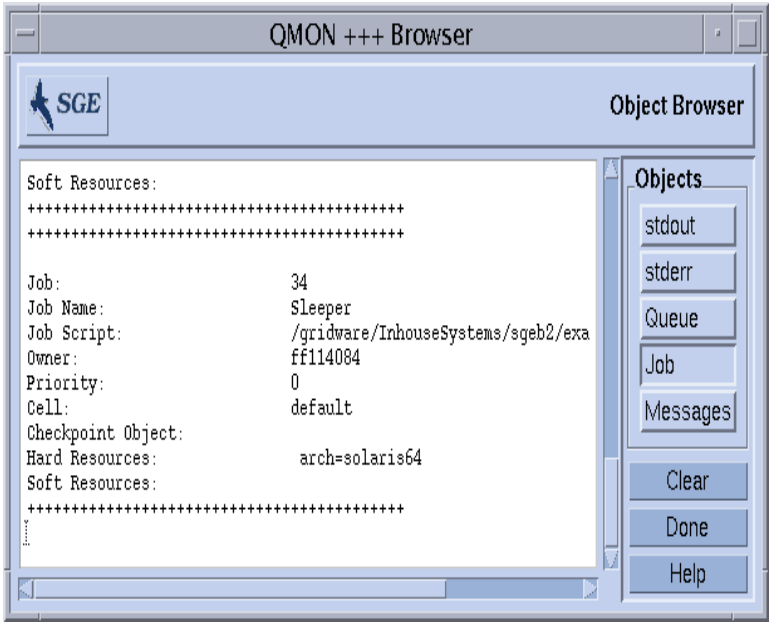


FIGURE 5-10 Object Browser—Job

▼ How To Monitor Jobs with qstat

- **From the command line, use one of the following commands, guided by information detailed in the following sections.**

```
% qstat
% qstat -f
```

The first form provides an overview of the submitted jobs only (see TABLE 5-1). The second form includes information on the currently configured queues in addition (see TABLE 5-2)).

In the first form, a header line indicates the meaning of the columns. The purpose of most of the columns should be self-explanatory. The `state` column, however, contains single character codes with the following meaning: `r` for running, `s` for suspended, `q` for queued and `w` for waiting (see the `qstat` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for a detailed explanation of the `qstat` output format).

The second form is divided into two sections, the first displaying the status of all available queues, the second (entitled with the `- PENDING JOBS - ...` separator) shows the status of the `sge_qmaster` job spool area. The first line of the queue section defines the meaning of the columns with respect to the enlisted queues. The queues are separated by horizontal rules. If jobs run in a queue they are printed below the associated queue in the same format as in the `qstat` command in its first form. The pending jobs in the second output section are also printed as in qstat's first form.

The following columns of the queue description require some more explanation.

- `qtype` – This is the queue type—one of `B`(atch), `I`(nteractive), `P`(arallel) and `C`(heckpointing) or combinations thereof.
- `used/free` – This is the count of used / free job slots in the queue.
- `states` – This is the state of the queue—one of `u`(nknown), `a`(laram), `s`(uspended), `d`(isabled), `E`(rror), or combinations thereof.

Again, the `qstat` manual page contains a more detailed description of the `qstat` output format.

Various additional options to the `qstat` command enhance the functionality in both versions. The `-r` option can be used to display the resource requirements of submitted jobs. Furthermore the output may be restricted to a certain user, to a specific queue and the `-l` option may be used to specify resource requirements as described in the section, "Resource Requirement Definition" on page 86 for the `qsub`

command. If resource requirements are used, only those queues (and the jobs running in these queues) are displayed that match the resource requirement specification in the `qstat` command line.

TABLE 5-1 Example of `qstat` Output

job-ID	prior	name	user	state	submit/start at	queue	function
231	0	hydra	craig	r	07/13/96 20:27:15	durin.q	MASTER
232	0	compile	penny	r	07/13/96 20:30:40	durin.q	MASTER
230	0	blackhole	don	r	07/13/96 20:26:10	dwain.q	MASTER
233	0	mac	elaine	r	07/13/96 20:30:40	dwain.q	MASTER
234	0	golf	shannon	r	07/13/96 20:31:44	dwain.q	MASTER
236	5	word	elaine	qw	07/13/96 20:32:07		
235	0	andrun	penny	qw	07/13/96 20:31:43		

TABLE 5-2 Example of `qstat -f` Output

queuename			qtype	used/free	load_avg	arch	states	
dq			BIP	0/1	99.99	sun4	au	
durin.q			BIP	2/2	0.36	sun4		
231	0	hydra		craig	r	07/13/96	20:27:15	MASTER
232	0	compile		penny	r	07/13/96	20:30:40	MASTER
dwain.q			BIP	3/3	0.36	sun4		
230	0	blackhole		don	r	07/13/96	20:26:10	MASTER
233	0	mac		elaine	r	07/13/96	20:30:40	MASTER
234	0	golf		shannon	r	07/13/96	20:31:44	MASTER
fq			BIP	0/3	0.36	sun4		
###								
- PENDING JOBS - PENDING JOBS - PENDING JOBS - PENDING JOBS - PENDING JOBS -								
###								
236	5	word		elaine	qw	07/13/96	20:32:07	
235	0	andrun		penny	qw	07/13/96	20:31:43	

▼ How To Monitor Jobs by Electronic Mail

- **From the command line, enter the following command with appropriate arguments, guided by information detailed in the following sections.**

```
% qsub arguments
```

The `qsub -m` switch requests electronic mail to be sent to the user submitting a job or to the email address(es) specified by the `-M` flag if certain events occur (see the `qsub` manual page for a description of the flags). An argument to the `-m` option specifies the events. The following selections are available:

- `b` – Mail is sent at the beginning of the job.
- `e` – Mail is sent at the end of the job.
- `a` – Mail is sent when the job is aborted (e.g. by a `qdel` command).
- `s` – Mail is sent when the job is suspended.
- `n` – No mail is sent (the default).

Multiple of these options may be selected with a single `-m` option in a comma-separated list.

The same mail events can be configured by help of the QMON Job Submission dialogue box. See the section, “Advanced Example” on page 81.

Controlling Sun Grid Engine Jobs from the Command Line

The section “How To Monitor and Control Jobs with `QMON`” on page 117 explains how Sun Grid Engine jobs can be deleted, suspended and resumed with the Sun Grid Engine graphical user‘s interface, `QMON`.

Equivalent functionality is also available from the command line, described in this section.

▼ How To Control Jobs from the Command Line

- **From the command line, enter one of the following commands and appropriate arguments, guided by information detailed in the following sections.**

```
% qdel arguments
% qmod arguments
```

You use the `qdel` command to cancel Sun Grid Engine jobs, regardless whether they are running or spooled. The `qmod` command provides the means to suspend and unsuspend (resume) jobs already running.

For both commands, you will need to know the job identification number, which is displayed in response to a successful `qsub` command. If you forget the number it can be retrieved via `qstat` (see the section, "How To Monitor Jobs with `qstat`" on page 127).

Included below are several examples for both commands:

```
% qdel job_id
% qdel -f job_id1, job_id2
% qmod -s job_id
% qmod -us -f job_id1, job_id2
% qmod -s job_id.task_id_range
```

In order to delete, suspend or unsuspend a job you must be either the owner of the job, a Sun Grid Engine manager or operator (see "Managers, Operators and Owners" on page 67).

For both commands, the `-f` force option can be used to register a status change for the job(s) at `sge_qmaster` without contacting `sge_execd` in case `sge_execd` is unreachable, e.g. due to network problems. The `-f` option is intended for usage by the administrator. In case of `qdel`, however, users can be enabled to force deletion of their own jobs if the flag `ENABLE_FORCED_QDEL` in the cluster configuration `qmaster_params` entry is set (see the `sge_conf` manual page in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for more information).

Job Dependencies

The most convenient way to build a complex task often is to split the task into sub-tasks. In these cases sub-tasks depend on the successful completion of other sub-tasks before they can get started. An example is that a predecessor task produces an output file which has to be read and processed by a successor task.

Sun Grid Engine supports interdependent tasks with its job dependency facility. Jobs can be configured to depend on the successful completion of one or multiple other jobs. The facility is enforced by the `qsub -hold_jid` option. A list of jobs can be specified upon which the submitted job depends. The list of jobs can also contain subsets of array jobs. The submitted job will not be eligible for execution unless all jobs in the dependency list have completed successfully.

Controlling Queues

As described in the section, “Queues and Queue Properties” on page 56, the owners of queues have permission to suspend / unsuspend or disable / enable queues. This is desirable, if these users need certain machines from time to time for important work and if they are affected strongly by Sun Grid Engine jobs running in the background.

There are two ways to suspend or enable queues.

- The `QMON` Queue Control dialogue box
- The `qmod` command

▼ How To Control Queues with `QMON`

- **In the `QMON` Main menu, click the Queue Control button.**

The Queue Control dialogue box, similar to that shown in FIGURE 5-11, is displayed.

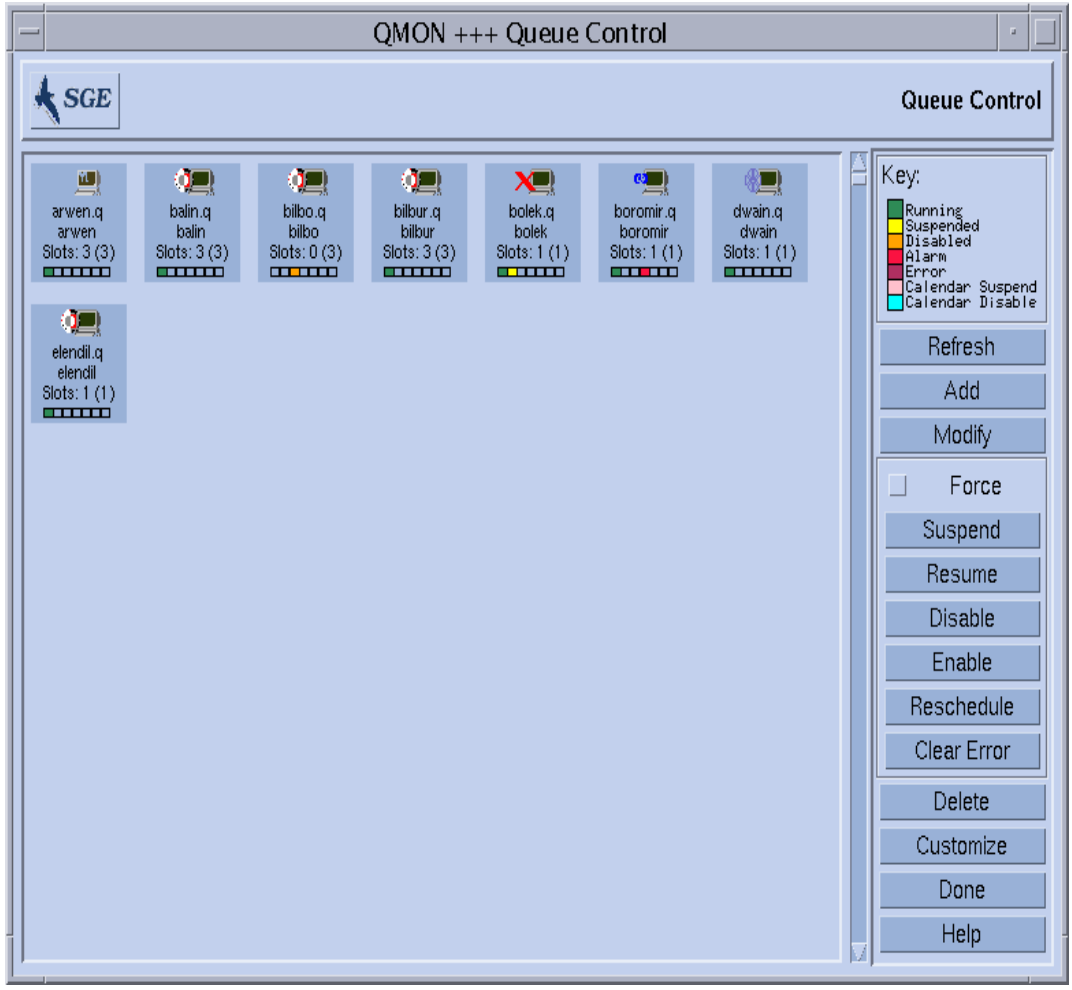


FIGURE 5-11 Queue Control Dialogue Box

The purpose of the Queue Control dialogue box is to provide a quick overview on the resources being available and on the activity in the cluster. It also provides the means to suspend / unsuspend and to disable / enable queues as well as to configure queues. Each icon being displayed represents a queue. If the main display area is empty, no queues are configured. Each queue icon is labelled with the queue name, the name of the host on which the queue resides and the number of job slots being occupied. If a `sge_execd` is running on the queue host and has already registered with `sge_qmaster` a picture on the queue icon indicates the queue host's operating

system architecture and a color bar at the bottom of the icon informs about the status of the queue. A legend on the right side of the Queue Control dialogue box displays the meaning of the colors.

For those queues, the user can retrieve the current attribute, load and resource consumption information for the queue and implicitly of the machine which hosts a queue by clicking to the queue icon with the left mouse button while the Shift key on the keyboard is pressed. This will pop-up an information screen similar to the one displayed in FIGURE 5-12.

Queues are selected by clicking with the left mouse on the button or into a rectangular area surrounding the queue icon buttons. The Delete, Suspend / Unsuspend or Disable / Enable buttons can be used to execute the corresponding operation on the selected queues. The suspend / unsuspend and disable / enable operation require notification of the corresponding `sge_execd`. If this is not possible (e.g., because the host is down) a `sge_qmaster` internal status change can be forced if the Force toggle button is switched on.

If a queue is suspended, the queue is closed for further jobs and the jobs already executing in the queue are suspended as explained in the section, “How To Monitor and Control Jobs with `QMON`” on page 117. The queue and its jobs are resumed as soon as the queue is unsuspended.

Note – If a job in a suspended queue has been suspended explicitly in addition, it will not be resumed if the queue is unsuspended. It needs to be unsuspended explicitly again.

Queues which are disabled are closed, however, the jobs executing in those queues are allowed to continue. To disable a queue is commonly used to “drain“ a queue. After the queue is enabled, it is eligible for job execution again. No action on still executing jobs is performed.

The suspend / unsuspend and disable / enable operations require queue owner or Sun Grid Engine manager or operator permission (see the section, “Managers, Operators and Owners” on page 67).

The information displayed in the Queue Control dialogue box is update periodically. An update can be forced by pressing the Refresh button. The Done button closes the dialogue box.

The Customize button enables you to select the queues to be displayed via a filter operation. The sample screen in FIGURE 5-13 shows the selection of only those queues that run on hosts belonging to architecture `osf4` (i.e, Compaq UNIX version 4). The Save button in the Customization dialogue box allows you to store your settings in the file, `.qmon_preferences` in your home directory for standard reactivation on later invocations of `QMON`.

For the purpose of configuring queues, a sub-dialogue box is opened when you press the Add or Modify button on the right side of the Queue Control screen (see the section, “How To Configure Queues with QMON” on page 164 for details).

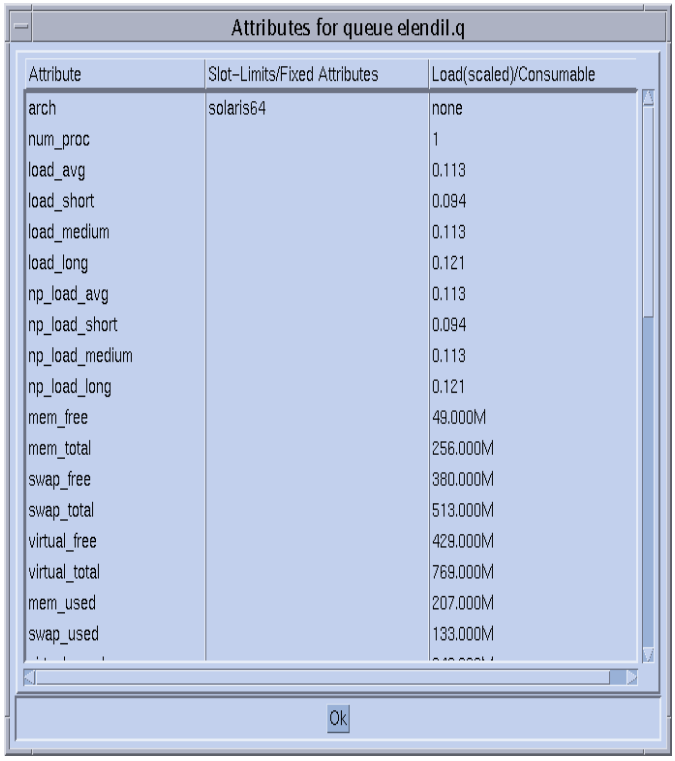


FIGURE 5-12 Queue Attribute Display

All attributes attached to the queue (including those being inherited from the host or cluster) are listed in the Attribute column. The Slot-Limits/Fixed Attributes column shows values for those attributes being defined as per queue slot limits or as fixed complex attributes. The Load(scaled)/Consumable column informs about the reported (and if configured scaled) load parameters (see the section, “Load Parameters” on page 206) and about available resource capacities based on the Sun Grid Engine consumable resources facility (see the section, “Consumable Resources” on page 194).

Note – Load reports and consumable capacities may overwrite each other, if a load attribute is configured as a consumable resource. The minimum value of both, which is used in the job dispatching algorithm, is displayed.

Note – The displayed load and consumable values currently do not take into account load adjustment corrections as described in the section, “Execution Hosts” on page 29.

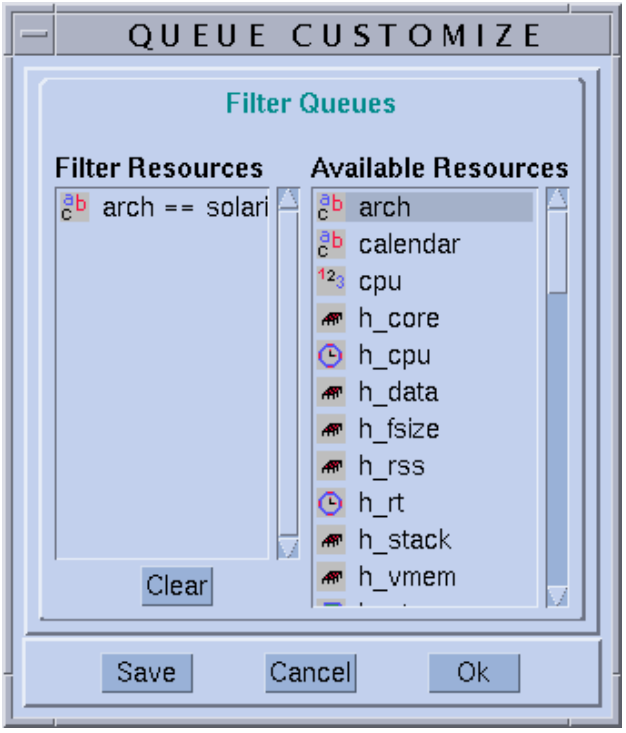


FIGURE 5-13 Queue Control Customization

▼ How To Control Queues with qmod

The section, “How To Control Jobs from the Command Line” on page 130 explained how the Sun Grid Engine command, qmod, can be used to suspend/unsuspend Sun Grid Engine jobs. However, the qmod command additionally provides the user with the means to suspend/unsuspend or disable/enable queues.

- **Enter the following command with appropriate arguments, guided by information detailed in the following sections.**

```
% qmod arguments
```

The following commands are examples how qmod is to be used for this purpose:

```
% qmod -s q_name
% qmod -us -f  q_name1, q_name2
% qmod -d q_name
% qmod -e  q_name1, q_name2, q_name3
```

The first two commands suspend or unsuspend queues, while the third and fourth command disable and enable queues. The second command uses the qmod -f option in addition to force registration of the status change in sge_qmaster in case the corresponding sge_execd is not reachable, e.g. due to network problems.

Note – Suspending / unsuspending as well as disabling / enabling queue requires queue owner, Sun Grid Engine manager or operator permission (see the section, “Managers, Operators and Owners” on page 67).

Note – You can use qmod commands with crontab or at jobs.

Customizing QMON

The look and feel of QMON is largely defined by a specifically designed resource file. Reasonable defaults are compiled in and a sample resource file is available under <sge_*root*>/qmon/Qmon.

The cluster administration may install site specific defaults in standard locations such as /usr/lib/X11/app-defaults/Qmon, by including QMON specific resource definitions into the standard .Xdefaults or .Xresources files or by putting a site specific Qmon file to a location referenced by standard search paths such as XAPPLRESDIR. Ask your administrator if any of the above is relevant in your case,

In addition, the user can configure personal preferences by either copying and modifying the Qmon file into the home directory (or to another location pointed to by the private XAPPLRESDIR search path) or by including the necessary resource definitions into the user‘s private .Xdefaults or .Xresources files. A private Qmon resource file may also by installed via the xrdb command during operation or at start-up of the X11 environment, e.g. in a .xinitrc resource file.

Refer to the comment lines in the sample Qmon file for detailed information on the possible customizations.

Another means of customizing QMON has been explained for the Job Control and Queue Control Customization dialogue boxes shown in FIGURE 5-2 and in FIGURE 5-13. In both dialogue boxes, you can use the Save button to store the filtering and display definitions configured with the customization dialogue boxes to the file, `.qmon_preferences`, in the user's home directory. Upon being restarted, QMON reads this file and reactivates the previously defined behavior.

PART IV Administration

Intended for the administrator, this part of the *Sun Grid Engine 5.3 Administration and User's Guide* includes six chapters.

- **Chapter 6** – "Host and Cluster Configuration" on page 141

  This chapter provides general background about, and detailed instructions for, configuring Sun Grid Engine 5.3 hosts and clusters.

- **Chapter 7** – "Configuring Queues and Queue Calendars" on page 163

  This chapter includes a description of the important concept of *queues*—which serve as "containers" for different categories of Sun Grid Engine 5.3 jobs. Complete instructions for configuring queues are included in this chapter.

- **Chapter 8** – "The Complexes Concept" on page 183

  This chapter explains how the Sun Grid Engine 5.3 system uses *complexes* to define all the pertinent information concerning the resource attributes a user may request for a job. The administrator configures various complexes to match the requirements of the environment, and this chapter provides detailed instructions for doing so.

- **Chapter 9** – "Managing User Access and Policies" on page 213

  This chapter provides full background information about the types of user policies that are available through the Sun Grid Engine 5.3 system, and provides instructions on how to match these policies to the computing environment.

- **Chapter 10** – "Managing Parallel Environments" on page 245

  In addition to describing how the Sun Grid Engine 5.3 systems fits in with *parallel environments*, this chapter provides full configuration instructions to address them.

- **Chapter 11** – "Error Messaging" on page 257

  This chapter explains the Sun Grid Engine 5.3 procedure for error message retrieval and describes how to run the software in debug mode.

CHAPTER 6

Host and Cluster Configuration

This chapter provides background information about, and instructions for, configuring various aspects of the Sun Grid Engine 5.3 system. You will find instructions in this chapter for the following tasks.

About Master and Shadow Master Configuration

The shadow master host name file, *<sge_root>/<cell>*/common/shadow_masters contains the name of the primary master host (the machine the Sun Grid Engine master daemon sge_qmaster is initially running on) and the *shadow master* hosts. The format of the master host name file is as follows.

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

The order of appearance of the (shadow) master hosts is significant. If the primary master host (the first line in the file) fails to proceed, the shadow master defined in the second line will take over. If this one fails also, the one defined in the third line is on duty and so forth.

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

- A shadow master host needs to run sge_shadowd.
- The shadow master hosts need to share sge_qmaster's status information, job and queue configuration logged to disk. In particular the (shadow) master hosts need read/write root access to the master's spool directory and to the directory *<sge_root>/<cell>*/common.
- The shadow master hostname file has to contain a line defining the host as shadow master host.

As soon as these requirement are met, the shadow master host facility is activated for this host. No restart of Sun Grid Engine daemons is necessary to activate the feature.

The automatic failover start of a sge_qmaster on a shadow master host will take some time (in the order of one minute). Meanwhile you will get a corresponding error message whenever a Sun Grid Engine command is executed.

Note – The file *<sge_root>/<cell>*/common/act_qmaster contains the name of the host actually running the sge_qmaster daemon.

In order to be able to start a shadow sge_qmaster Sun Grid Engine must be sure that either the *old* sge_qmaster has terminated or that it will terminate without performing actions interfering with the just started shadow sge_qmaster. Under very rare circumstances this is impossible. In these cases, a corresponding error message will be logged to the messages logfile of the sge_shadowds on the shadow master hosts (see Chapter 11, "Error Messaging" on page 257) and any attempts to

open a tcp connection to a sge_qmaster daemon will permanently fail. If this occurs, make sure that no master daemon is running and restart sge_qmaster manually on any of the shadow master machines (see the section, “How To Kill Daemons from the Command Line” on page 156).

About Daemons and Hosts

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

- **Master host** – The master host is central for the overall cluster activity. It runs the master daemon sge_qmaster. sge_qmaster controls all Sun Grid Engine components such as queues and jobs and maintains tables about the status of the components, about user access permissions and the like. The section, “How To Install the Master Host” on page 32 describes how to initially set up the master host, and the section, “About Master and Shadow Master Configuration” on page 142 shows how dynamic master host changes can be configured. The master host usually runs the Sun Grid Engine scheduler sge_schedd. The master host requires no further configuration other than performed by the installation procedure.
- **Execution hosts** – Execution hosts are nodes having permission to execute Sun Grid Engine jobs. Therefore, they are hosting Sun Grid Engine queues and run the Sun Grid Engine execution daemon sge_execd. An execution host is initially set up by the execution host installation procedure as described in the section, “How To Install Execution Hosts” on page 33).
- **Administration hosts** – Permission can be given to other hosts than the master host to carry out any kind of administrative activity in Sun Grid Engine. Administrative hosts are set up with the following command:

  qconf -ah *hostname*

  See the qconf manual page for details.
- **Submit hosts** – Submit hosts allow for submitting and controlling batch jobs only. In particular a user being logged into a submit host can submit jobs via qsub, can control the job status via qstat or run Sun Grid Engine's OSF/1 Motif graphical user's interface, QMON. Submit hosts are set up with the following command:

  qconf -as *hostname*

  See the qconf manual page for details.

Note – A host may belong to more than one of the above described classes. The master host is an administrative and submit host by default.

About Configuring Hosts

Sun Grid Engine maintains object lists for all types of hosts except for the master host. In the case of the administrative and submit hosts these lists simply provide the information whether or not a host has administrative or submit permission. In the case of the execution host object, further parameters, such as the load information as reported by the `sge_execd` running on the host is stored there as well as load parameter scaling factors to be provided by the Sun Grid Engine administrator.

The following sections explain how to configure the different host objects with the help of the Sun Grid Engine graphical user interface, `QMON` and from the command line.

The GUI administration is provided by a set of host configuration dialogue boxes which are invoked by pushing the Host Config icon button in the `QMON` Main menu. The available dialogue boxes are the Administration Host Configuration (see FIGURE 6-1), the Submit Host Configuration (see FIGURE 6-2), and the Execution Host Configuration (see FIGURE 6-3). The dialogue boxes can be switched by using the selection list button at the top of the screen.

The `qconf` command provides the command line interface for the host object management.

▼ How To Configure Administration Hosts with QMON

1. **Click the Administration Host tab at the top of the `QMON` Main menu.**

   The Administration Host Configuration dialogue box, which is similar to the following figure, is opened.

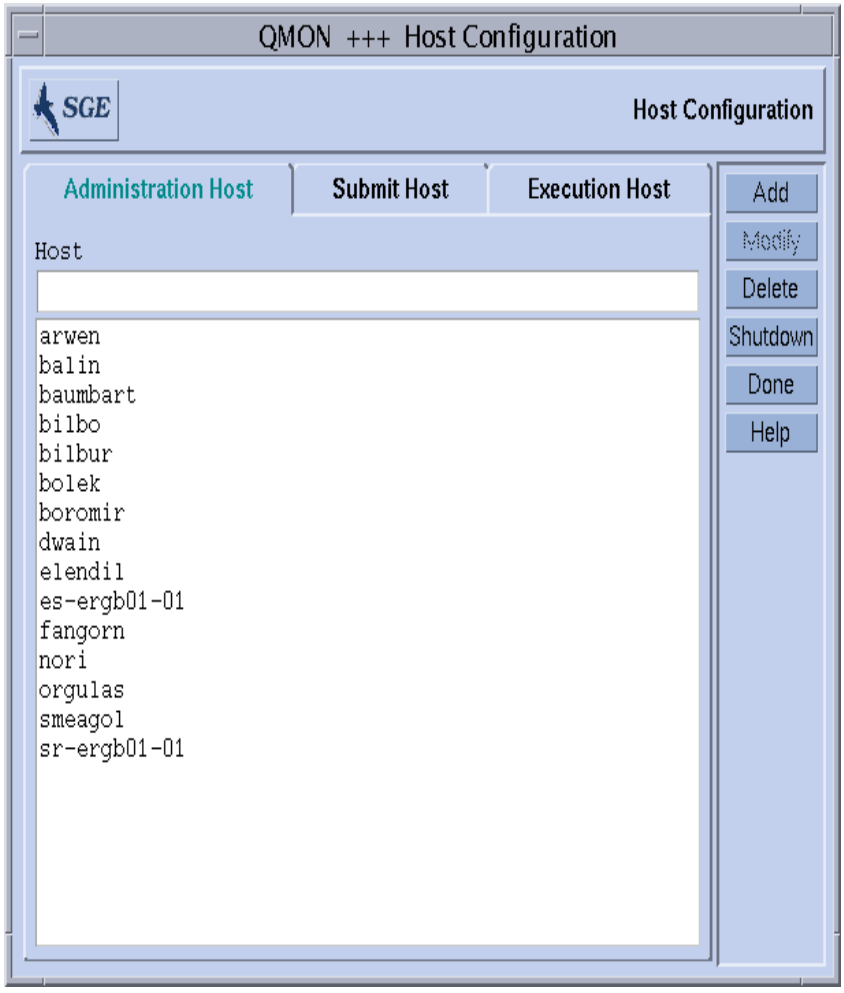


FIGURE 6-1 Administration Host Configuration Dialogue Box

Note – The Administration Host Configuration dialogue box is opened by default when the Host Config button is pressed for the first time.

---

2. **Depending on how you want to configure the host, proceed according to the guidance in the following sections.**

   With this dialogue box, you can configure hosts from which administrative Sun Grid Engine commands are allowed. The selection list in the center of the screen displays the hosts already declared to provide administrative permission.

▼ How To Delete an Administration Host

- **Delete an existing host from this list by clicking on its name with the left mouse button and then pressing the Delete button at the bottom of the dialogue box.**

▼ How To Add an Administration Host

- **Add a new host by entering its name into the Hostname input window and then pressing the Add button or the Return key.**

▼ How To Configure Administration Hosts From the Command Line

- **Enter the following command with appropriate arguments, depending on how you want to configure the host.**

```
% qconf arguments
```

Arguments to the `qconf` command and their consequences follow.

- `qconf -ah` *hostname*

  Add administrative host—adds the specified host to the list of administrative hosts.

- `qconf -dh` *hostname*

  Delete administrative host—deletes the specified host from the list of administrative hosts.

- `qconf -sh`

  Show administrative hosts—displays a list of all currently configured administrative hosts.

▼ How To Configure Submit Hosts with QMON

1. **Click the Submit Host tab at the top of the `QMON` Main menu.**

The Submit Host Configuration dialogue box, which is similar to the following figure, is opened.

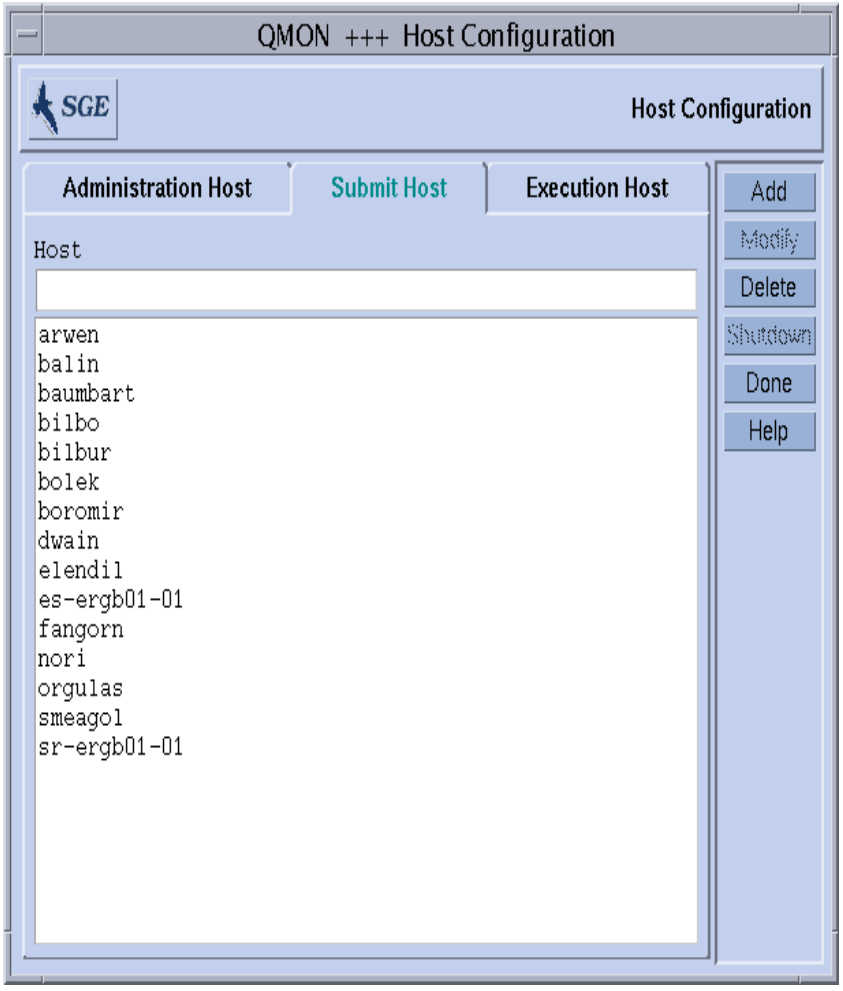


FIGURE 6-2 Submit Host Configuration

2. **Depending on how you want to configure the host, proceed according to the guidance in the following sections.**

   Using this dialog box, you can declare the hosts from which jobs can be submitted, monitored, and controlled. No administrative Sun Grid Engine commands are allowed from these hosts unless they are declared to be administrative hosts also (see “How To Configure Administration Hosts with QMON” on page 144). The selection list in the center of the screen displays the hosts already declared to provide submit permission.

▼ How To Delete a Submit Host

- **Delete an existing host by clicking on its name in the Submit Host dialogue box, and then pressing the Delete button at the bottom of the dialogue box.**

▼ How To Add a Submit Host

- **Add a host by entering its name into the Hostname input window in the Submit Host dialogue box, and then pressing the Add button or Return key.**

▼ How To Configure Submit Hosts from the Command Line

- **Enter the following command with appropriate arguments, depending on how you want to configure the host.**

```
% qconf arguments
```

Arguments to the `qconf` command and their consequences follow.

- `qconf -as` *hostname*

  Add submit host—adds the specified host to the list of submit hosts.

- `qconf -ds` *hostname*

  Delete submit host—deletes the specified host from the list of submit hosts.

- `qconf -ss`

  Show submit hosts—displays a list of the names of all hosts currently configured to provide submit permission.

▼ How To Configure Execution Hosts with QMON

1. **Click the Execution Host tab at the top of the `QMON` Main menu.**

   The Execution Host Configuration dialogue box, which is similar to the following figure, is opened.

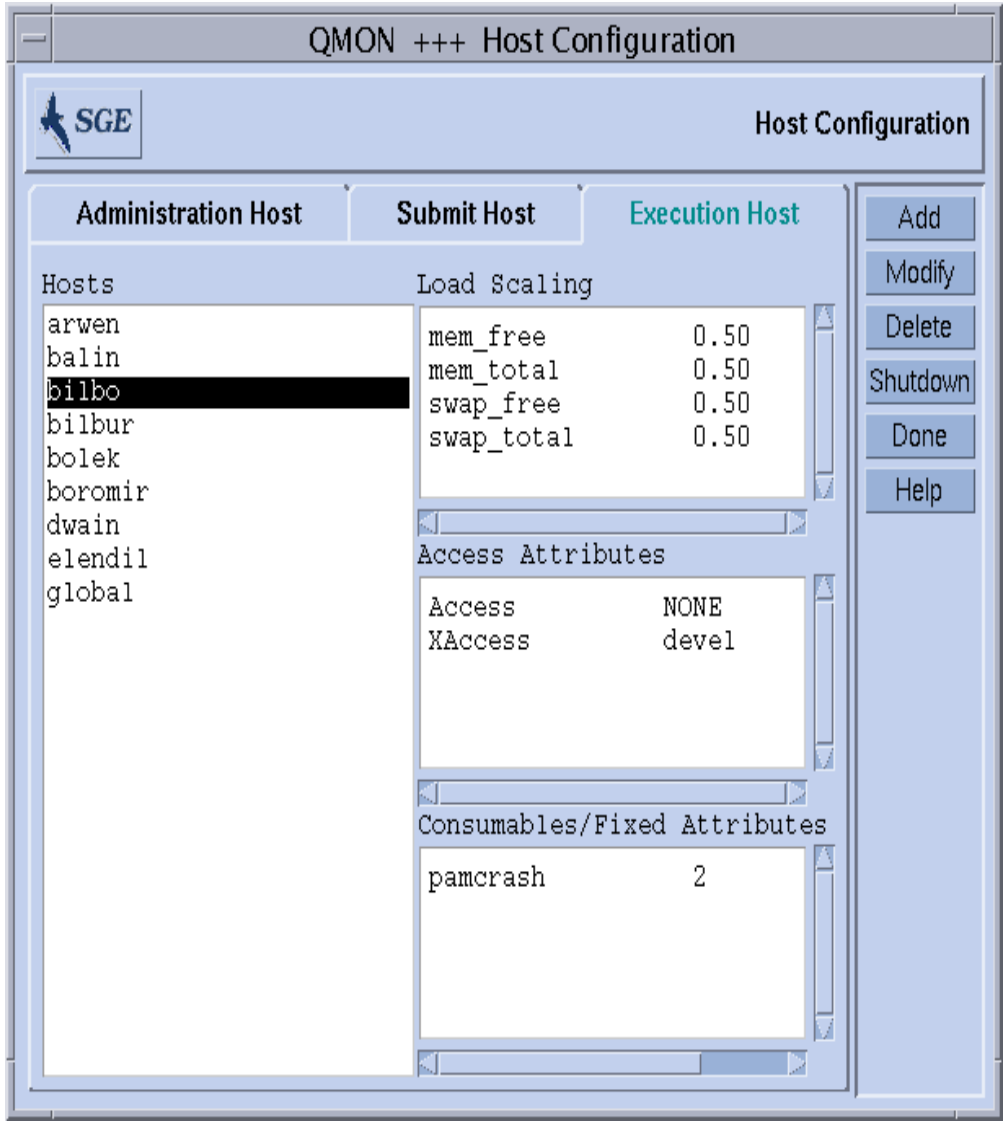


FIGURE 6-3 Execution Host Configuration

2. **Depending on how you want to configure the host, proceed according to the guidance in the following sections.**

Sun Grid Engine execution hosts can be configured from this dialogue box. No administrative or submit commands are automatically allowed from these hosts unless they are declared also to be administrative or submit hosts (see “How To Configure Administration Hosts with QMON” on page 144 and “How To Configure Submit Hosts with QMON” on page 146).

The Hosts selection list displays the execution hosts already defined. The currently configured load scaling factors, the access permissions and the resource availability for consumable and fixed complex attributes associated with the host are displayed in the Load Scaling, the Access Attributes and the Consumable/Fixed Attributes display windows for the selected execution host. Refer to the sections, “About Complexes” on page 183, “User Access Permissions” on page 66, and “Load Parameters” on page 206 for details on complex attributes, user access permissions, and load parameters.

▼ How To Delete an Execution Host

- **In the Execution Host dialogue box, click the name of the Execution host to be deleted and then press the Delete button at the button column on the right side of the dialogue box.**

▼ How To Shut Down the Execution Host Daemon

- **For any selected host, press the Shutdown button in the Execution Host dialogue box.**

▼ How To Add or Modify an Execution Host

1. **Press the Add or Modify button in the button column of the Execution Host dialogue box.**

   A dialogue box similar to the one displayed in FIGURE 6-4 appears.

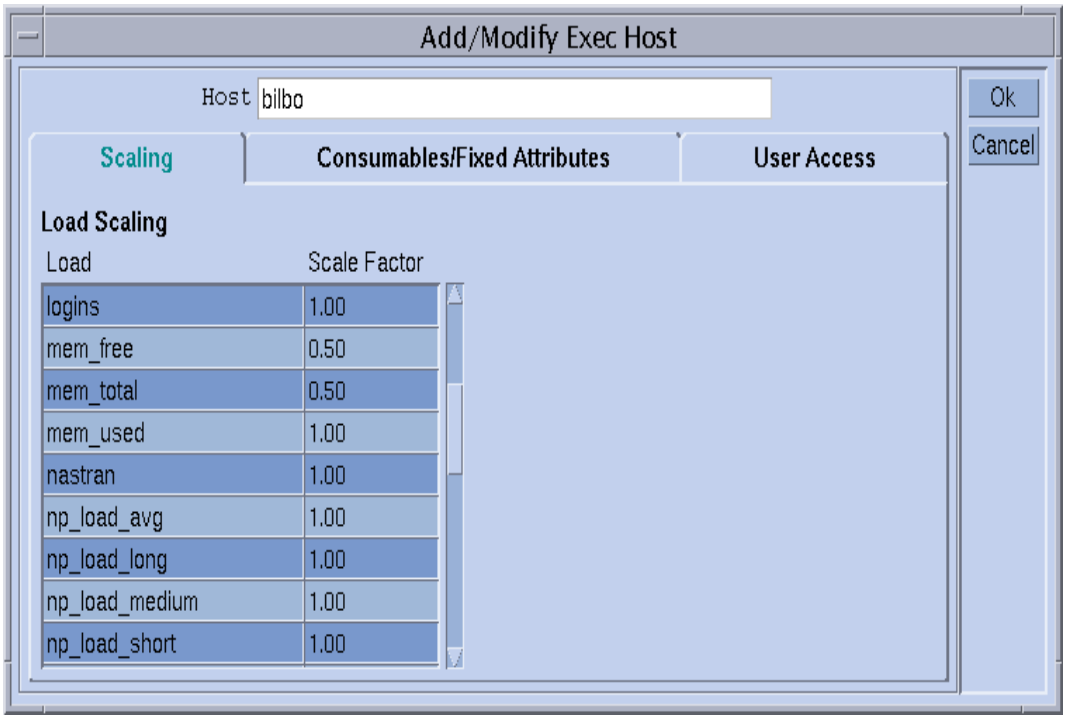


FIGURE 6-4 Modify Load Scaling

2. **Depending on how you want to modify the host, proceed according to the guidance in the following sections.**

   The dialogue box to add a new execution host or modify the configuration of an existing one allows for modification of all attributes associated with the host. The name of the execution host is displayed or can be added in the Host input window. You can define scaling factors by selecting the Scaling tab in the dialogue box (see FIGURE 6-4).

   All available load parameters are displayed in the Load column of the Load Scaling table and the corresponding definition of the scaling can be found in the Scale Factor column. The Scale Factor column can be edited. Valid scaling factors are positive floating point numbers in fixed point or scientific notation.

   When you select Consumables/Fixed Attributes is selected in the tab widget, the complex attributes associated with the host can be defined (see FIGURE 6-5). The complexes (see the section, “About Complexes” on page 183) associated with the host are the *global* and the *host complex* or the *administrator defined complexes* attached to the host via the Complex Selection area on the left bottom of the dialogue box. Available administrator defined complexes are displayed on the left and they can be

attached or detached via the red arrows. The Complex Configuration icon button opens the top level Complex Configuration dialogue box in case you need further information on the current complex configuration or if you want to modify it.

The Consumable/Fixed Attributes table in the right bottom area of the dialogue box enlists all complex attributes for which a value currently is defined. The list can be enhanced by clicking on the Name or Value button at the top. This will open a selection list with all attributes attached to the host (i.e., the union of all attributes configured in the global, the host and the administrator defined complexes attached to this host as described above). The Attribute Selection dialogue box is shown in FIGURE 6-6. Selecting one of the attributes and confirming the selection with the Ok button will add the attribute to the Name column of the Consumable/Fixed Attributes table and will put the pointer to the corresponding Value field. Modifying an existing value can be achieved by double-clicking with the left mouse button on the Value field. Deleting an attribute is performed by first selecting the corresponding table line with the left mouse button. The selected list entry can be deleted either by typing CTRL-D or by clicking the right mouse button to open a deletion box and confirming the deletion.

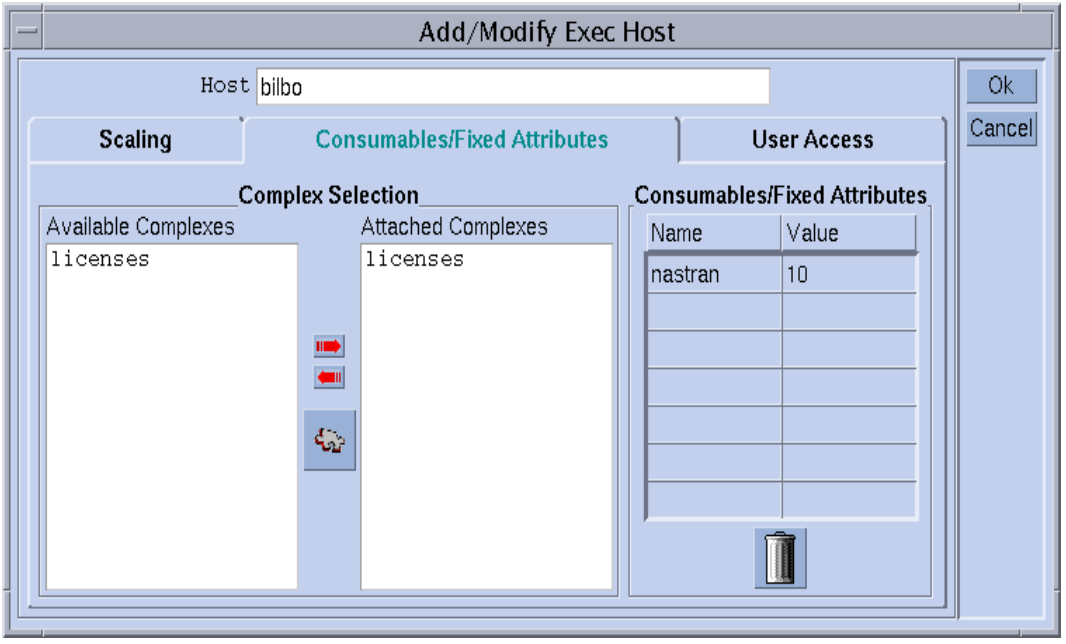


FIGURE 6-5 Modify Consumable/Fixed Attributes

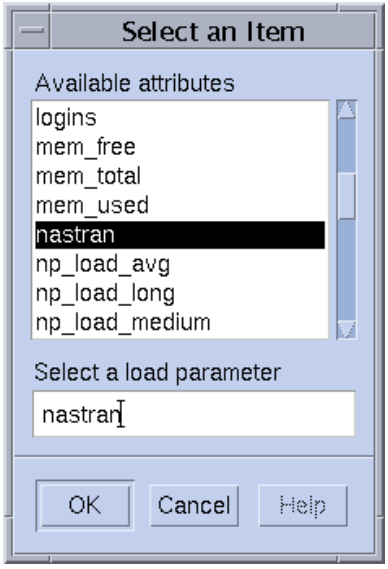


FIGURE 6-6 Available Complex Attributes

By selecting the User Access tab (FIGURE 6-7), you can define the access permissions to the execution host based on previously configured user access lists.

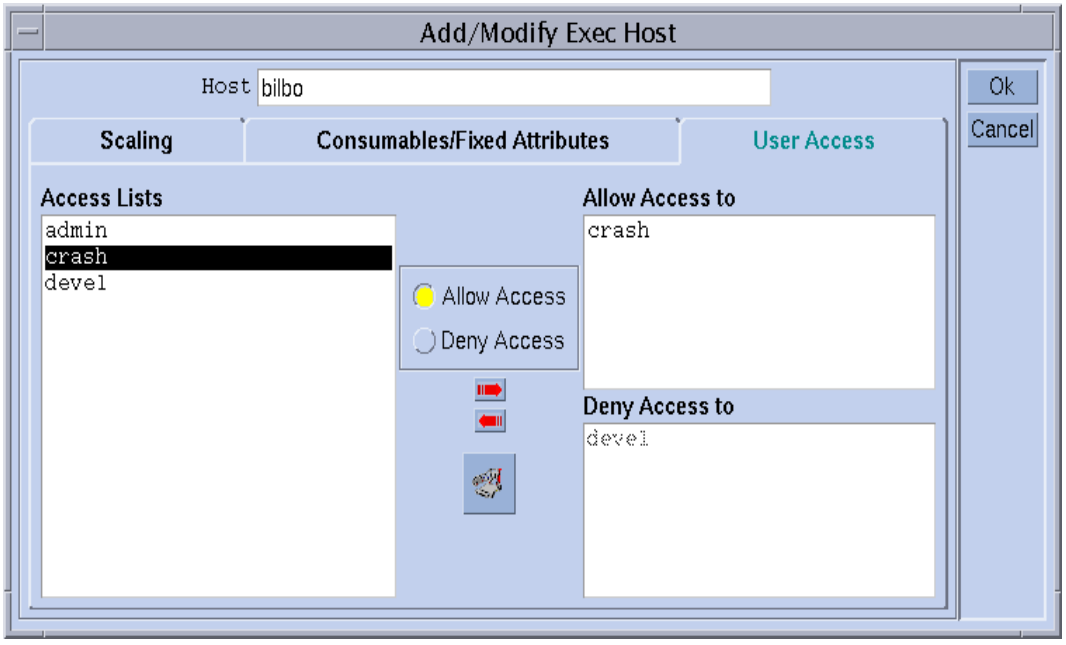


FIGURE 6-7 Modify User Access

▼ How To Configure Execution Hosts from the Command Line

- **Enter the following command with appropriate arguments, depending on how you want to configure the host.**

```
% qconf arguments
```

The command line interface for maintaining the list of execution hosts is provided by the following options to the `qconf` command.

- `qconf -ae` *[exec_host_template]*

Add execution host—brings up an editor (default `vi` or corresponding to the `$EDITOR` environment variable) with an execution host configuration template. If the optional parameter *exec_host_template* (the name of an already configured execution host) is present, the configuration of this execution host is used as

template. The execution host is configured by changing the template and saving to disk. See the `host_conf` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for a detailed description of the template entries to be changed.

- `qconf -de` *hostname*

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*

Modify execution host—brings up an editor (default `vi` or corresponding to the `$EDITOR` environment variable) with the configuration of the specified execution host as template. The execution host configuration is modified by changing the template and saving to disk. See the `host_conf` manual page in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for a detailed description of the template entries to be changed.

- `qconf -Me` *filename*

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 off-line execution host configuration changes; e.g., in `cron` jobs, as it requires no manual interaction.

- `qconf -se` *hostname*

Show execution host—show the configuration of the specified execution host as defined in `host_conf`.

- `qconf -sel`

Show execution host list.—display a list of host names that are configured to be execution hosts.

▼ How To Monitor Execution Hosts With `qhost`

The `qhost` command provides a convenient way to retrieve a quick overview of the execution host status.

- **Enter the following command.**

```
% qhost
```

The command produces output similar to the following.

TABLE 6-1 Sample `qhost` Output

HOSTNAME	ARCH	NPROC	LOAD	MEMTOT	MEMUSE	SWAPTO	SWAPUS
global	-	-	-	-	-	-	-
BALROG.genias.de	solaris6	2	0.38	1.0G	994.0M	900.0M	891.0M
BILBUR.genias.de	solaris	1	0.18	96.0M	70.0M	164.0M	9.0M
DWAIN.genias.de	irix6	1	1.13	149.0M	55.8M	40.0M	0.0
GLOIN.genias.de	osf4	2	0.05	768.0M	701.0M	1.9G	13.5M
SPEEDY.genias.de	alinux	1	0.08	248.8M	60.6M	125.7M	232.0K
SARUMAN.genias.de	solaris	1	0.11	96.0M	77.0M	192.0M	9.0M
FANGORN.genias.de	linux	1	2.01	124.8M	49.9M	127.7M	4.3M

Refer to the `qhost` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for a description of the output format and for further options.

▼ How To Kill Daemons from the Command Line

- **Use one of the following commands. Note that you need Sun Grid Engine manager or operator privileges for these operations (see Chapter 9, "Managing User Access and Policies" on page 213)**.

```
% qconf -kej
% qconf -ks
% qconf -km
```

- The first command will kill all currently active jobs and bring down all Sun Grid Engine execution daemons.

Note – If replacing that command by `qconf -ke`, the Sun Grid Engine execution daemons are aborted, but the active jobs are not cancelled. Jobs which finish while no `sge_execd` is running on that system are not reported to `sge_qmaster` until `sge_execd` is restarted again. The job reports are not lost, however.

- The second command will shut down the Sun Grid Engine scheduler `sge_schedd`.
- The third command will force the `sge_qmaster` process to terminate.

If you have running jobs and you want to wait with the shutdown procedure of Sun Grid Engine until the currently active jobs are finished, you can use the command below for each queue before executing the `qconf` sequence described above.

```
% qmod -d queue_name
```

The `qmod` disable command prevents new jobs from being scheduled to the disabled queues. You should then wait with the killing of the daemons until no jobs run in the queues any longer.

▼ How To Restart Daemons from the Command Line

1. **Log in as `root` to the machine on which you want Sun Grid Engine 5.3 daemons restarted.**
2. **Execute the following script.**

```
% <sge_root>/<cell>/common/rcsge
```

This script looks for the daemons normally running on this host, and subsequently starts the corresponding ones.

The Basic Cluster Configuration

The basic Sun Grid Engine cluster configuration is a set of information configured to reflect site dependencies like valid paths for programs such as `mail` or `xterm` and to influence the Sun Grid Engine behavior. There is a global configuration, which is provided by for the Sun Grid Engine master host as well as every host in the Sun Grid Engine pool. In addition, the Sun Grid Engine system may be configured to use a configuration local to every host to override particular entries in the global configuration.

The `sge_conf` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* contains a detailed description of the configuration entries. The Sun Grid Engine cluster administrator should adapt the global and local configurations to the site's needs directly after the installation and keep it up to date afterwards.

▼ How To Display the Basic Cluster Configurations from the Command Line

The Sun Grid Engine command to display the current configuration is the show configuration option of the `qconf` program. The following are a few examples (see the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for a detailed description).

- **Enter one of the following commands.**

```
% qconf -sconf
% qconf -sconf global
% qconf -sconf <host>
```

The first two commands are equivalent and display the global configuration. The third command displays the host's local configuration.

▼ How To Modify the Basic Cluster Configurations from the Command Line

---

Note – The Sun Grid Engine command—`qconf`—to change the cluster configurations may be used by Sun Grid Engine administrators only.

- **Enter one of the following commands.**

```
% qconf -mconf global
% qconf -mconf <host>
```

- The first command example modifies the global configuration.
- The second example operates on the local configuration of the specified execution or master host.

The two commands above are examples of the many available `qconf` commands. Refer to the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for others.

▼ How To Display a Cluster Configuration with QMON

1. **In the QMON Main menu, click the Cluster Configuration button.**

   The Cluster Configuration dialog box, similar to the example in FIGURE 6-8, is displayed.

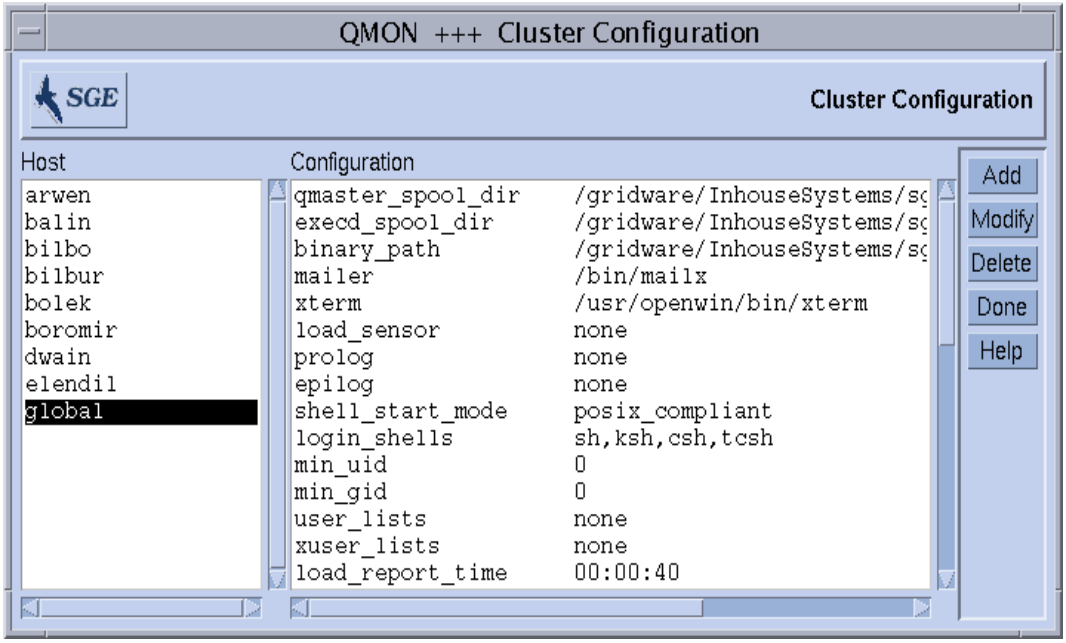


FIGURE 6-8 Cluster Configuration Dialog Box

2. **In the Host selection list on the left side of the screen, click the name of a host to display the current configuration for that host.**

▼ How To Delete a Cluster Configuration with QMON

1. **In the QMON Main menu, click the Cluster Configuration button.**
2. **In the Host selection list on the left side of the screen, click the name of a host whose configuration you want to delete.**
3. **Press the Delete button.**

▼ How To Display a Global Cluster Configuration with QMON

- **In the Host selection list, select the name, g**`lobal`**.**

  The configurations are displayed in the format which is described in the `sge_conf` manual page. Use the Modify button to modify the selected global or host local configuration. Use the Add button to add a new configuration for a specific host.

▼ How To Use QMON To Modify Global and Host Configurations

1. **In the Cluster Configuration dialogue box (described in the section, “How To Display a Cluster Configuration with QMON” on page 159), click either the Add button or the Modify button.**

   The Cluster Settings dialogue box, similar to the example in FIGURE 6-9, is opened.

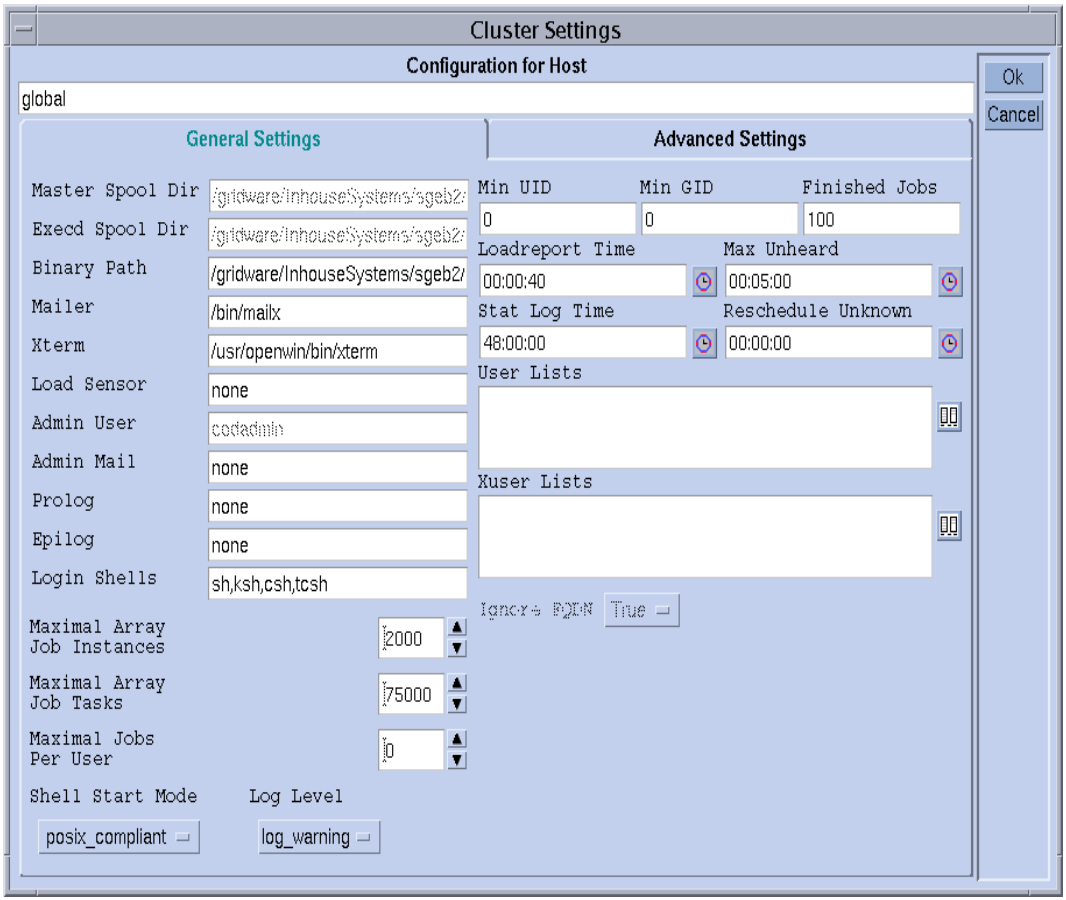


FIGURE 6-9 Cluster Settings Dialogue Box—General Settings

2. **Make any changes, guided by the information detailed in the following sections.**

   The Cluster Settings dialogue box provides the means for changing all parameters of a global or host local configuration. All entry fields are only accessible if the global configuration is changed; i.e., if you selected the host, global, and if you pressed Modify . If a regular host is modified, its actual configuration is reflected in the

dialogue box and only those parameters can be modified that are feasible for host local changes. If a new host local configuration is added, the dialogue box entries will be empty fields.

The Advanced Settings tab (FIGURE 6-10) shows a corresponding behavior depending on whether a global, host local or new configuration is changed. It provides access to more rarely used cluster configuration parameters.

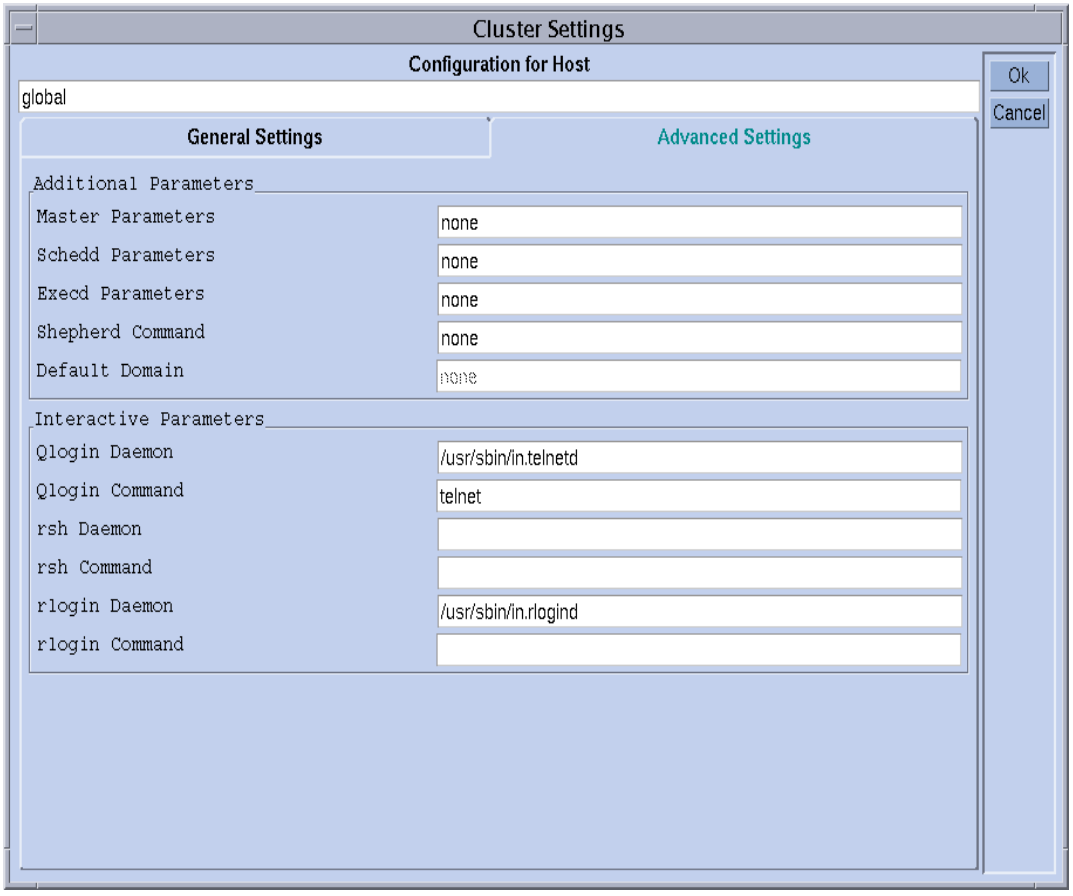


FIGURE 6-10 Cluster Settings Dialogue Box—Advanced Settings

After finishing the modifications, the Ok button on the right upper corner will register the modified configuration. Pressing Cancel discards any changes. The dialogue box is closed in both cases.

Refer to the `sge_conf` manual page for a complete description of all cluster configuration parameters.

CHAPTER 7

Configuring Queues and Queue Calendars

This chapter provides background information about, and instructions for, configuring Sun Grid Engine 5.3 queues.

A list of specific tasks for which instructions are included in this chapter follows.

About Configuring Queues

Sun Grid Engine *queues* are containers for different categories of jobs and provide the corresponding resources for concurrent execution of multiple jobs belonging to the same category. Jobs will not wait in Sun Grid Engine queues, but start running immediately as soon as they are dispatched. The Sun Grid Engine scheduler's job pending list is the only waiting area for Sun Grid Engine jobs.

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

▼ How To Configure Queues with QMON

1. **From the `QMON` Main menu, press the Queue Control button.**
2. **In the Queue Control dialogue box, press the Add or the Modify button.**

   The Queue Configuration dialogue box is opened. The Queue Control dialogue box and its facilities to monitor and manipulate the queue status are described in the section, “How To Control Queues with `QMON`” on page 132. If the Queue Configuration dialogue box is opened for the first time, it shows the general parameters form (see “How To Configure General Parameters” on page 165).

3. **Make configuration decisions guided by information detailed in the following sections.**

   The queue to be affected by the desired operation is displayed or defined in the Queue and Hostname windows in the upper screen region. If a queue is to be modified, an existing queue has to be selected in the Queue Control dialogue box before the Queue Configuration dialogue box is opened. A queue name and a host on which the queue resides must be defined if a new queue is going to be added.

   To increase the ease of use of the Queue Configuration dialogue box, three buttons are available directly below the Hostname window: The Clone button, which allows for the import of all parameters of an existing queue via a queue selection list, the Reset button, which loads the configuration of the template queue and the Refresh button, which loads the configuration of other objects which were modified while the Queue Configuration dialogue box was open (see the section, “How To Configure User Complexes” on page 172 and “How To Configure User Access” on page 175 for further details concerning the Refresh button).

   The Ok button on the right upper corner of the Queue Configuration dialogue box registers the changes with `sge_qmaster`, while the Cancel button below discards any changes. Both buttons close the dialogue box.

   Nine parameter sets are available to define a queue.

   - General (see “How To Configure General Parameters” on page 165)
   - Execution Method (see “How To Configure Execution Method Parameters” on page 167)
   - Checkpointing (see “How To Configure Checkpointing Parameters” on page 168)
   - Load/Suspend Thresholds (see “How To Configure Load and Suspend Thresholds” on page 169)

- Limits (see “How To Configure Limits” on page 170)
- Complexes (see “How To Configure User Complexes” on page 172)
- Subordinates (see “How To Configure Subordinate Queues” on page 174)
- User Access (see “How To Configure User Access” on page 175)
- Owners (see “How To Configure Owners” on page 176)

You select the desired parameter set via the Queue Parameter tab.

▼ How To Configure General Parameters

- **Select the General parameter set.**

  A screen similar to the example in FIGURE 7-1 is displayed.

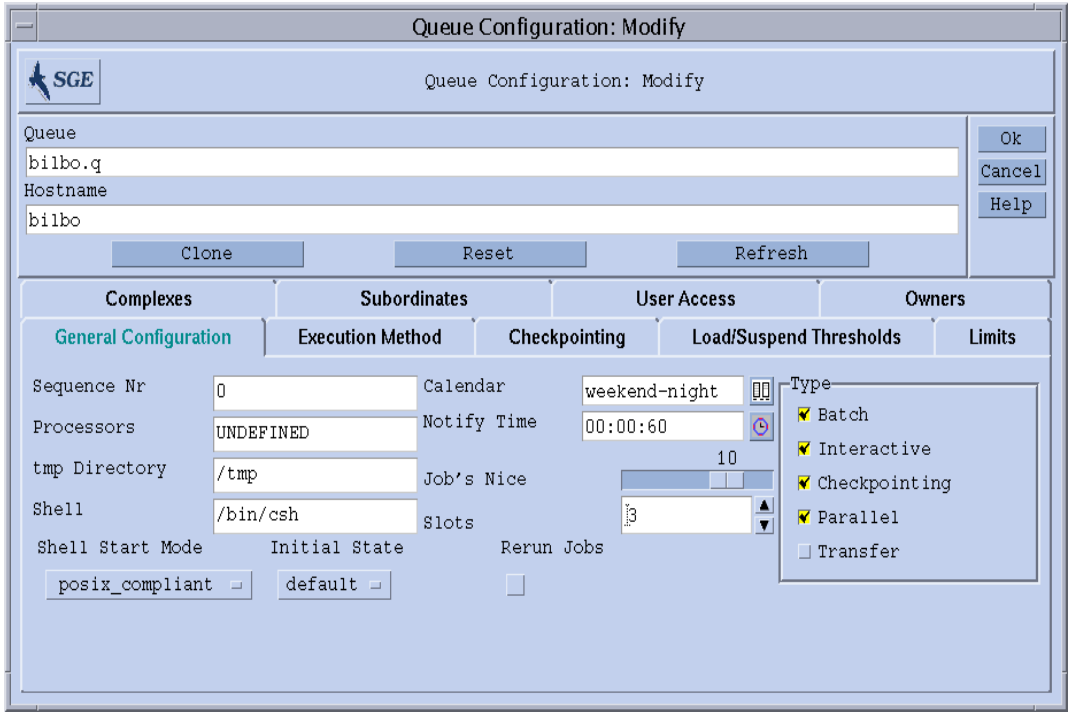


FIGURE 7-1 Queue Configuration—General Parameters

The fields offered allow for setting the following parameters:

- 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 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 Sun Grid Engine distribution for more information.
- Temporary directory path.
- Default command interpreter (`Shell`) to be used to execute the job scripts.
- A calendar attached to the queue defining *on-duty* and *off-duty* times for the queue.
- The time waited between delivery of `SIGUSR1/SIGUSR2` notification signals and suspend / kill signals (`Notify`).
- The `nice` value with which to start the jobs in this queue (`0` means use system default).
- The number of jobs to be allowed to execute concurrently in the queue (job slots).
- The type of the queue and of the jobs being allowed to execute in this queue. Multiple selections are feasible.
- The `Shell Start Mode`; i.e., the mode in which to start the job script.
- The `Initial State` in which a newly added queue comes up or in which the queue is restored if the `sge_execd` running on the queue host gets restarted.
- The queue's default `rerun` policy to be enforced on jobs which have been aborted; e.g., due to system crashes. The user may overwrite this policy by the `qsub -r` option or the Job Submission dialogue box (see FIGURE 4-7).

Refer to the `queue_conf` manual page for detailed information on these parameters.

▼ How To Configure Execution Method Parameters

- **Select the Execution Method parameter set.**

  A screen similar to the example in FIGURE 7-2 is displayed.

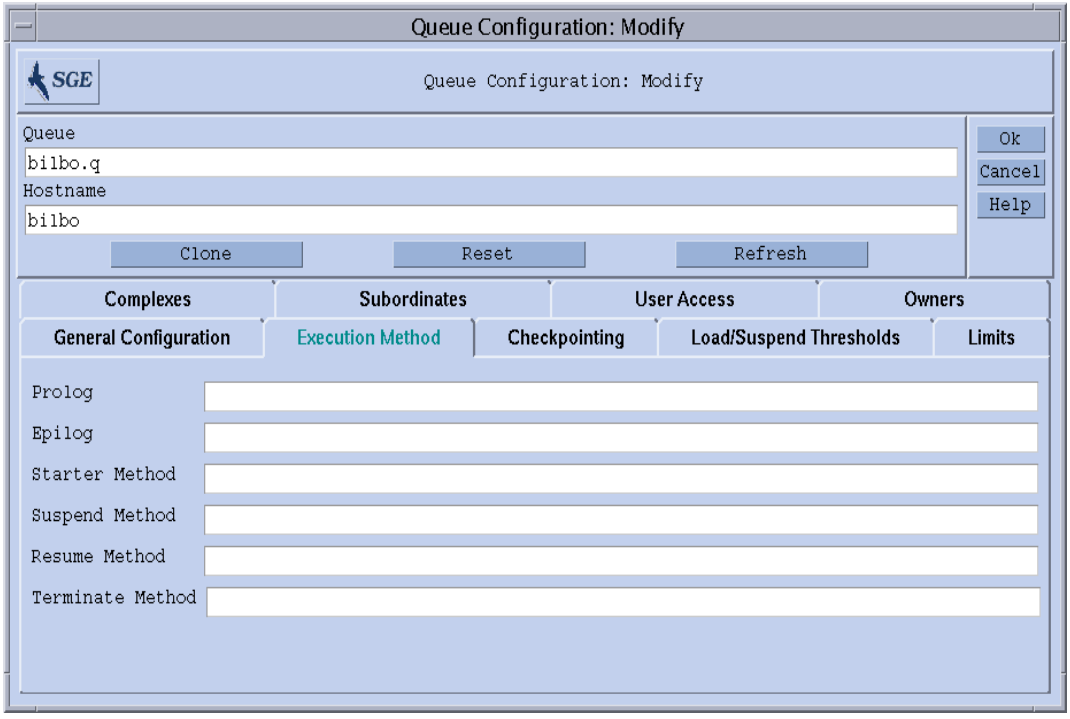


FIGURE 7-2 Queue Configuration—Execution Method parameters

The fields offered allow for setting the following parameters:

- A queue-specific prologue and epilogue script executed with the same environment as the job before the job script is started and after it is finished respectively.
- A start/suspend/resume/terminate method overwriting Sun Grid Engine's default methods for these applying these actions to jobs.

Refer to the `queue_conf` manual page for detailed information on these parameters.

▼ How To Configure Checkpointing Parameters

- **Select the Checkpointing parameter set.**

  A screen similar to the example in FIGURE 7-3 is displayed.

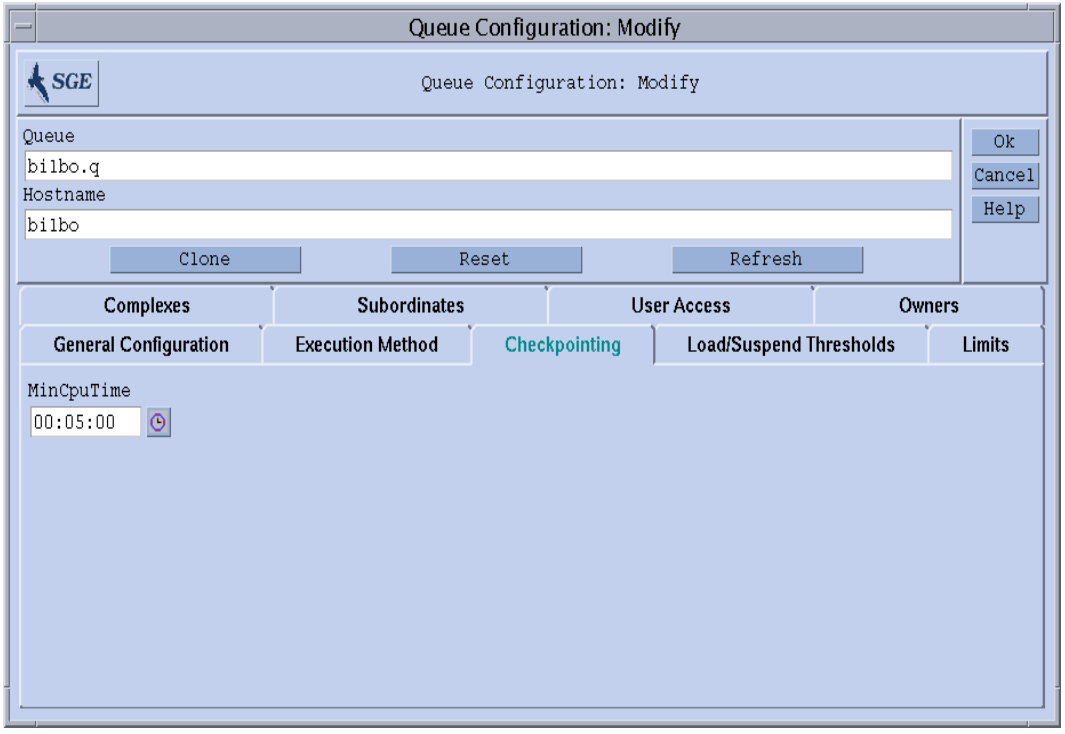


FIGURE 7-3 Queue Configuration—Checkpointing parameters

The field offered allows for setting the following parameter.

- The periodical checkpoint interval (`MinCpuTime`)

Refer to the `queue_conf` manual page for detailed information on these parameters.

▼ How To Configure Load and Suspend Thresholds

- **Select the Load/Suspend Thresholds parameter set.**

  A screen similar to the example in FIGURE 7-4 is displayed.

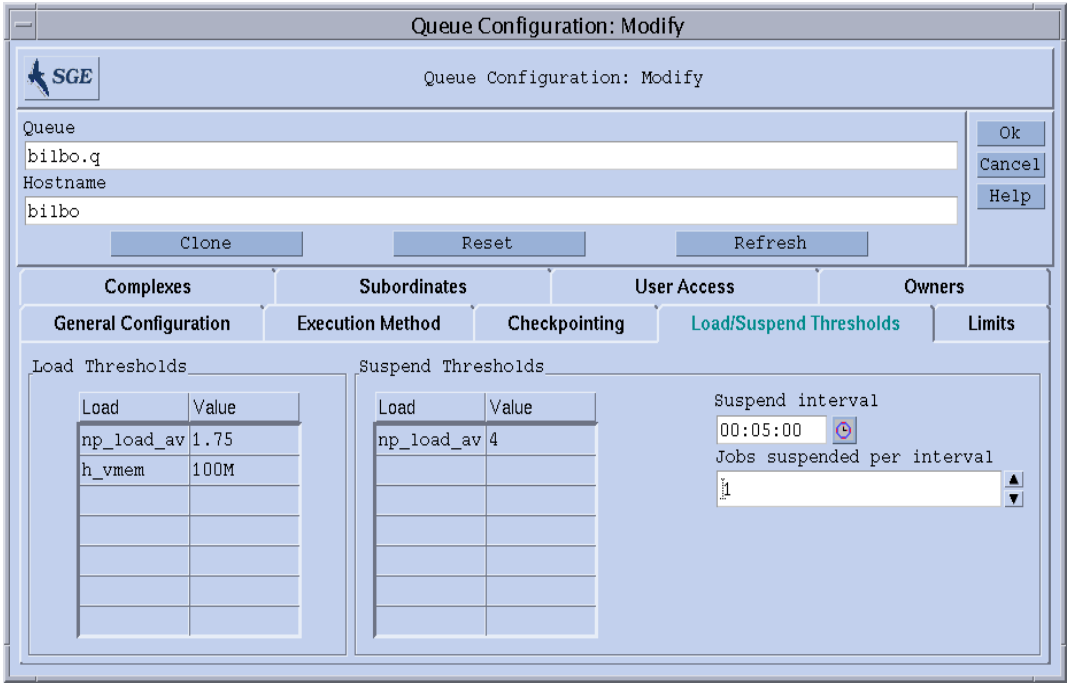


FIGURE 7-4 Queue Configuration—Load Thresholds

The fields offered allow for setting the following parameters.

- The Load Thresholds and the Suspend Thresholds tables, which define overload thresholds for load parameters and consumable complex attributes (see “About Complexes” on page 183).

  Overload in the case of load thresholds results in preventing the queue from receiving further jobs by Sun Grid Engine. Exceeding one or more suspend thresholds causes suspension of jobs in the queue to reduce the load. The currently configured thresholds are displayed in the tables. An existing threshold can be selected and changed by double-clicking with the left mouse button to the corresponding Value field. To add new thresholds click to the Name or Value button at the top. This will open a selection list with all valid attributes attached to the queue. The Attributes Selection dialogue box is shown in FIGURE 6-6. Selecting one of the attributes and confirming the selection with the Ok button will add the attribute to the Name column of the corresponding threshold table

and will put the pointer to its Value field. A selected list entry can be deleted either by typing CTRL-D or by clicking the right mouse button to open a deletion box and confirming the deletion.

- The number of jobs which are suspended per time interval to reduce the load on the system which hosts the configured queue.
- The time interval between suspension of further jobs in case suspend thresholds are still exceeded.

Refer to the `queue_conf` manual page for detailed information on these parameters.

▼ How To Configure Limits

- **Select the Limits parameter set.**

  A screen similar to the example in FIGURE 7-5 is displayed.

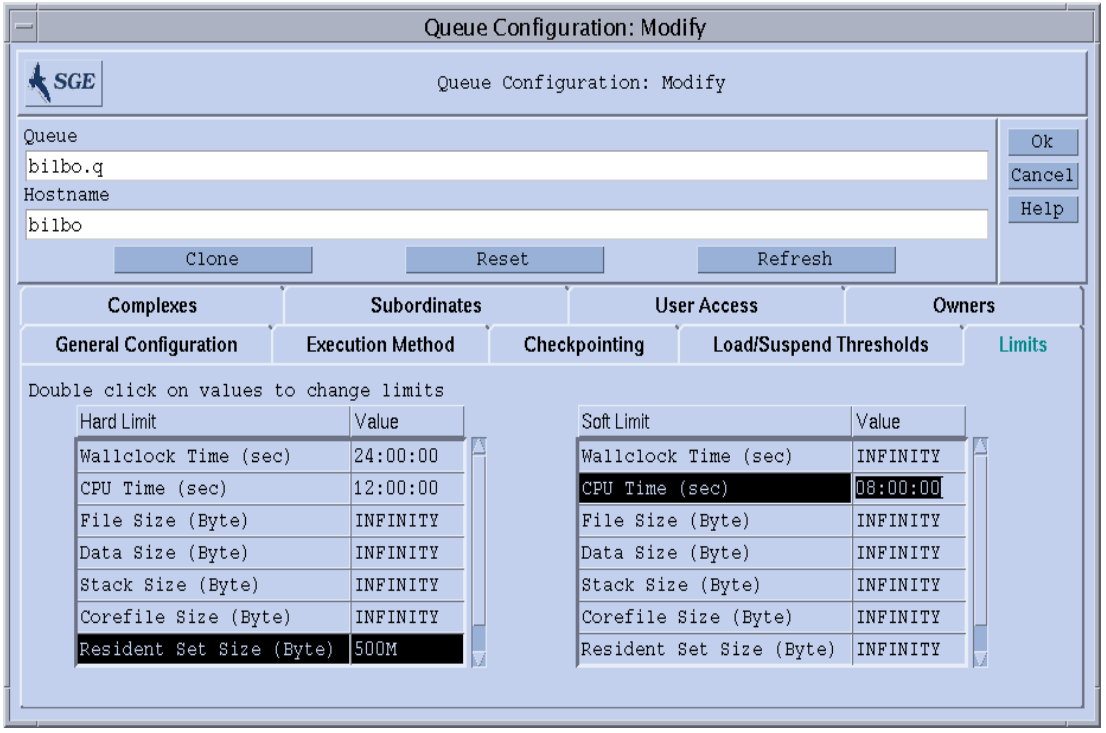


FIGURE 7-5 Queue Configuration—Limits

The fields offered allow for setting the following parameters.

- The *hard* and *soft* limits which are to be imposed on the jobs running in the queue.

To change a value of a limit double-click the Value field of the limit entry. Double clicking a Value field twice opens convenient input dialogue boxes for either Memory or Time limit values (see FIGURE 7-6 and FIGURE 7-7).

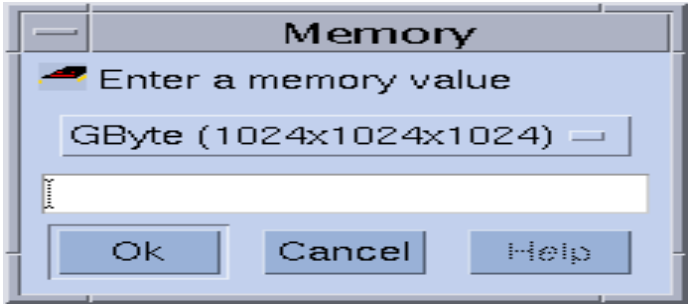


FIGURE 7-6 Memory Input Dialogue Box

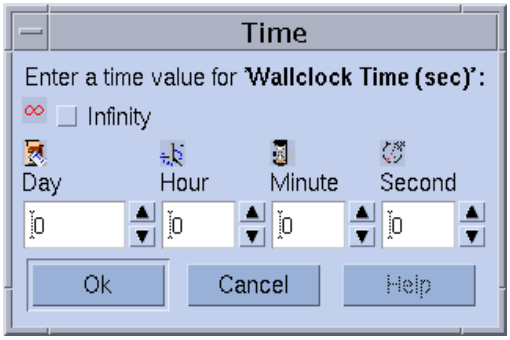


FIGURE 7-7 Time Input Dialogue Box

Refer to the `queue_conf` and `setrlimit` manual page for detailed information on the individual limit parameters and their interpretation for different operating system architectures.

▼ How To Configure User Complexes

- **Select the User Complexes parameter set.**

  A screen similar to the example in FIGURE 7-8 is displayed.

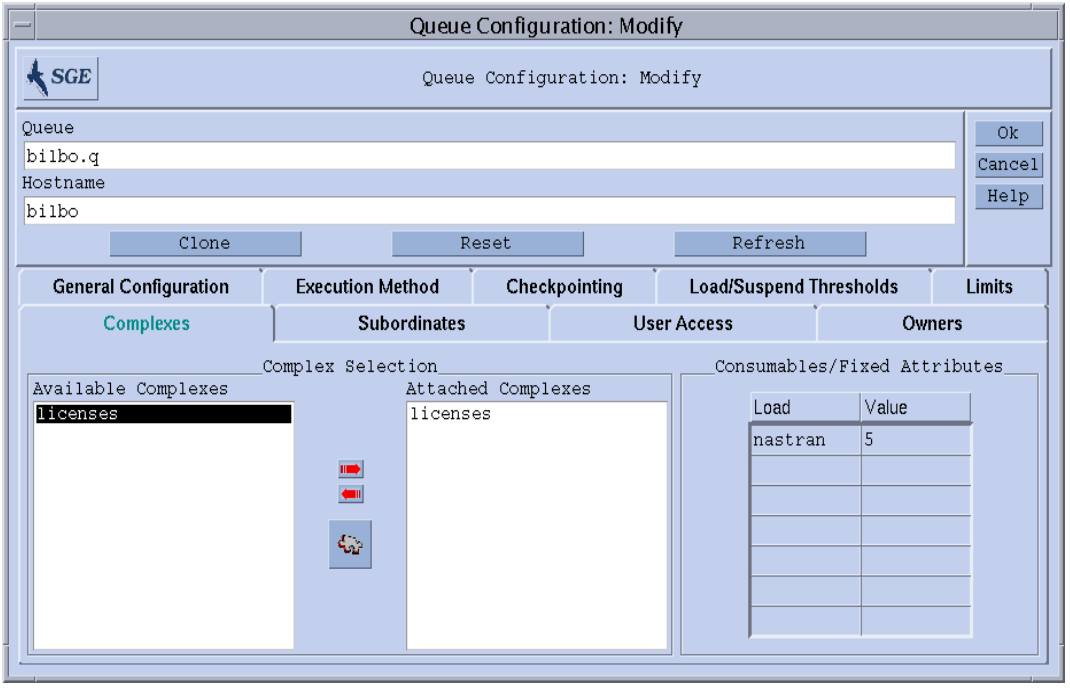


FIGURE 7-8 Queue Configuration—User Complexes

The fields offered allow for setting the following parameters.

- The set of user defined complexes (see “User-Defined Complexes” on page 190) being attached to the queue

  The red arrows in the center of the Complex Selection box allow to attach and detach a user defined complex from/to the queue.

- A value definition for selected attributes from the set of complexes parameters available for this queue

  The available complex parameters are assembled per default from the global complex, the host complex and from the attached user defined complexes. Attributes are either consumable or fixed parameters. The definition of a queue value defines a capacity managed by the queue in the case of a consumable attribute or simply a fixed, queue specific value in the case of fixed attributes (see “About Complexes” on page 183 for further details). The attributes, for which values are explicitly defined, are displayed in the Consumable/Fixed Attributes table. An existing attribute can be selected and changed by double-clicking the

corresponding Value field. To add new attribute definitions click to the Name or Value button at the top. This will open a selection list with all valid attributes attached to the queue. The Attribute Selection dialogue box is shown in FIGURE 6-6. Selecting one of the attributes and confirming the selection with the Ok button will add the attribute to the Name column of the attribute table and will put the pointer to its Value field. A selected list entry can be deleted either by typing CTRL-D or by clicking the right mouse button to open a deletion box and confirming the deletion.

Refer to the `queue_conf` manual page for detailed information on these parameters.

The Complex Configuration dialogue box (see FIGURE 8-5 in Chapter 8, “The Complexes Concept” on page 183 for an example) is opened upon clicking on the Complex Config icon button. You can check or modify the current complexes configuration before user-defined complexes are attached or detached to a queue.

▼ How To Configure Subordinate Queues

- **Select the Subordinates parameter set.**

  A screen similar to the example in FIGURE 7-9 is displayed.

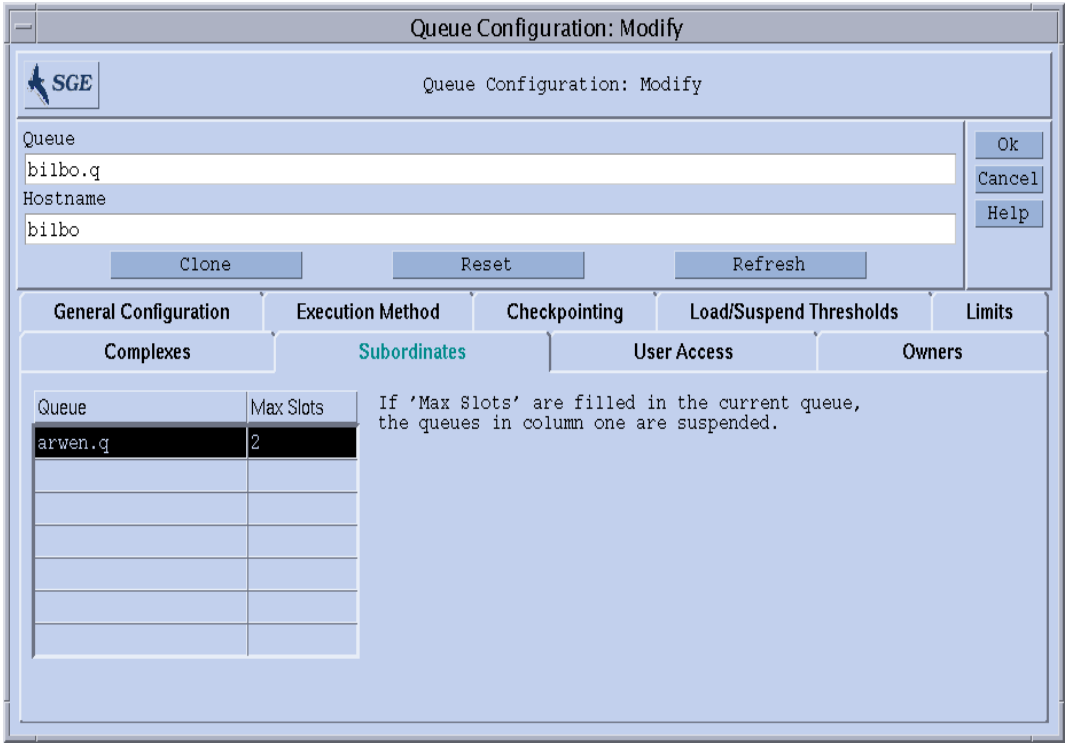


FIGURE 7-9 Queue Configuration—Subordinates

The fields offered allow for setting the following parameters.

- The queues that are *subordinated* to the configured queue

  Subordinated queues are suspend if the configured queue becomes *busy* and are unsuspended if the configured queue is no longer busy. For any subordinated queue the number of job slots can be configured which at least has to be occupied in the configured queue to trigger a suspension. If no job slot value is specified, all slots need to be filled to trigger suspension of the corresponding queue.

Rrefer to the `queue_conf` manual page for detailed information on these parameters.

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

▼ How To Configure User Access

- **Select the User Access parameter set.**

  A screen similar to the example in FIGURE 7-10 is displayed.

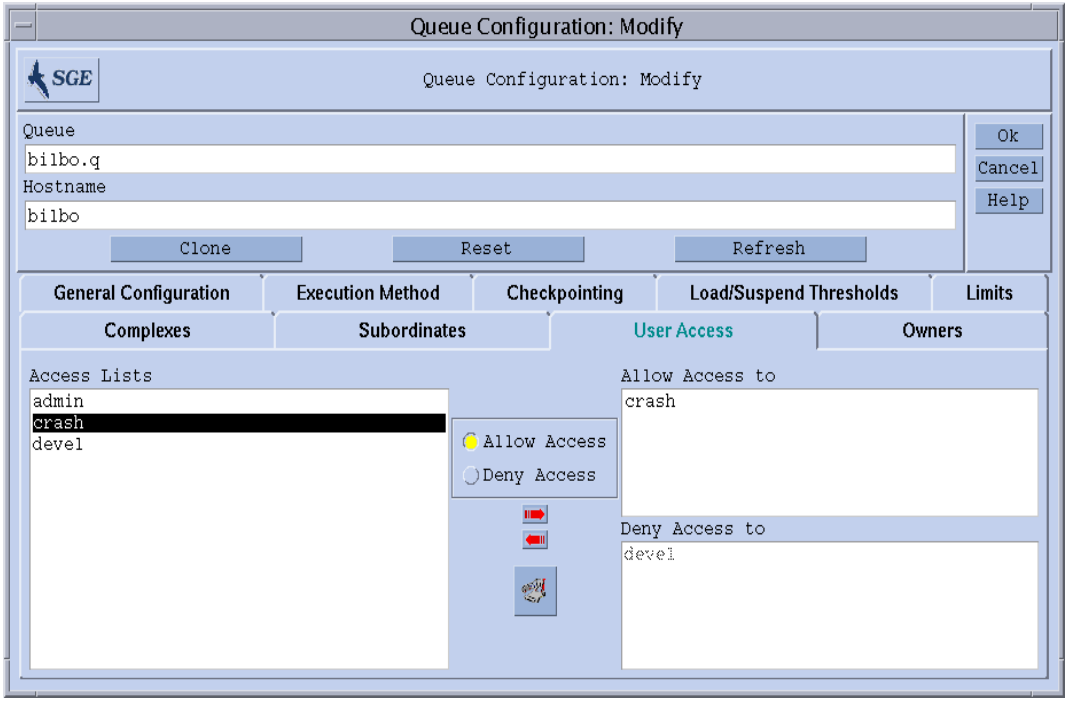


FIGURE 7-10 Queue Configuration—User Access

The fields offered allow for setting the following parameters.

- The user access lists being attached to the allow or deny lists of the queue

  Users or user groups belonging to access lists which are included in the allow list have access to the queue. Those being associated with the deny list may not access the queue. If the allow list is empty access is unrestricted unless explicitly stated otherwise in the deny list.

Refer to the `queue_conf` manual page for detailed information on these parameters.

Open the Access List Configuration dialogue box (see “User Access Permissions” on page 66) by clicking the button in the middle bottom of the screen.

▼ How To Configure Owners

- **Select the Owners parameter set.**

  A screen similar to the example in FIGURE 7-11 is displayed.

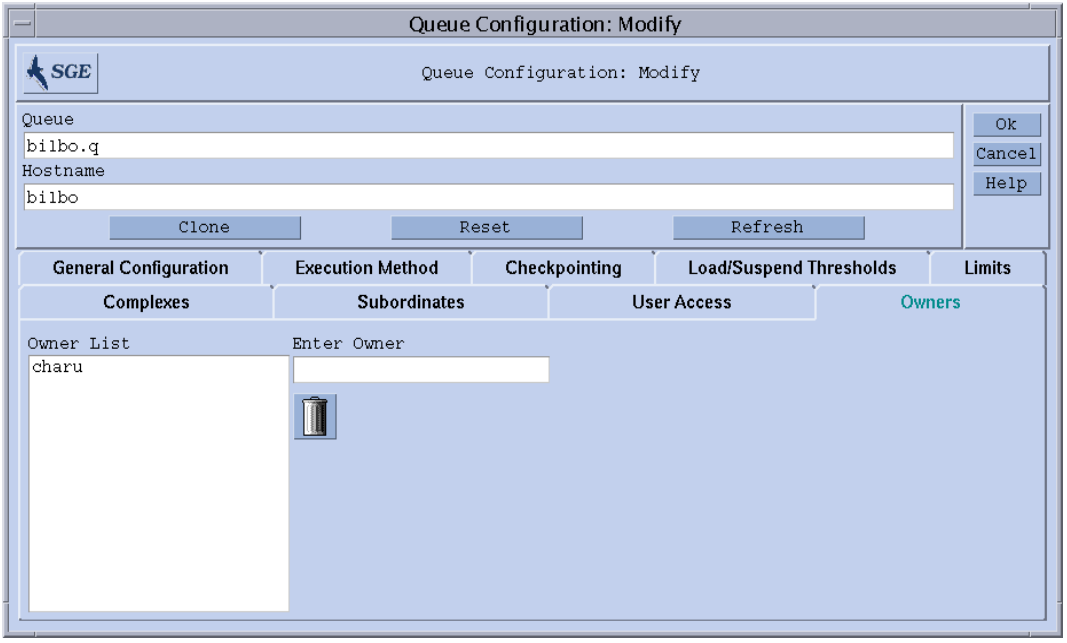


FIGURE 7-11 Queue Configuration—Owners

The fields offered allow for setting the following parameters:

- The list of queue owners

  An owner of a queue is given permission to suspend/unsuspend or disable/enable the queue. All feasible user accounts are valid values to be added to the queue owner list. To delete an user account from the queue owner list select it in the Owner List window and click on the garbage bin icon in the right lower corner of the dialogue box.

Refer to the `queue_conf` manual page for detailed information on these parameters.

▼ How To Configure Queues from the Command Line

- **Enter the following command and appropriate options, depending on how you want to configure the queues.**

```
# qconf options
```

The qconf command has the following options.

- `qconf -aq` *[queue_name]*

  Add queue—brings up an editor (default `vi` or corresponding to the `$EDITOR` environment variable) with a queue configuration template. If the optional parameter *queue_name* is present, the configuration of this queue is used as template. The queue is configured by changing the template and saving to disk. See the `queue_conf` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for a detailed description of the template entries to be changed.

- `qconf -Aq` *file_name*

  Add queue—uses the file *file_name* to define a queue. The definition file might have been produced by `qconf -sq` *queue_name* (see below).

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

  Clean queue—cleans the status of the specified queue(s) to be idle and free from running jobs. The status is reset without respect to the current status. The option is useful for eliminating error conditions, but should not be used in normal operation mode.

- `qconf -dq` *queue_name[,...]*

  Delete queue—deletes the queue(s) specified in the argument list from the list of available queues.

- `qconf -mq` *queue_name*

  Modify queue—modifies the specified queue. Brings up an editor (default `vi` or corresponding to the `$EDITOR` environment variable) with the configuration of the queue to be changed. The queue is modified by changing the configuration and saving to disk.

- `qconf -Mq` *file_name*

  Modify queue—uses the file *file_name* to define the modified queue configuration. The definition file might have been produced by `qconf -sq` *queue_name* (see below) and subsequent modification.

- `qconf -sq` *[queue_name[,...]]*

Show queue—either displays the default template queue configuration (if no arguments are present) or the current configuration of the queues enlisted in the comma separated argument list.

- `qconf -sql`

  Show queue list—displays a list of all currently configured queues.

About Queue Calendars

Queue calendars define the availability of Sun Grid Engine queues dependent on the day of the year, the day of the week and/or the day time. Queues can be configured to change their status at arbitrary points in time. The queue status can be changed to disabled, enabled, suspended and resumed (unsuspended).

Sun Grid Engine provides the ability to define a site specific set of calendars, each of which contains arbitrary status changes and the time events at which they occur. These calendars can be referred to by queues, i.e. each queue may (or may not) attach a single calendar thereby adopting the availability profile defined in the attached calendar.

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

▼ How To Configure Queue Calendars With QMON

1. **In the `QMON` Main menu, click Calendar Config.**

The Queue Calendar Configuration dialogue box, similar to FIGURE 7-12, is displayed.

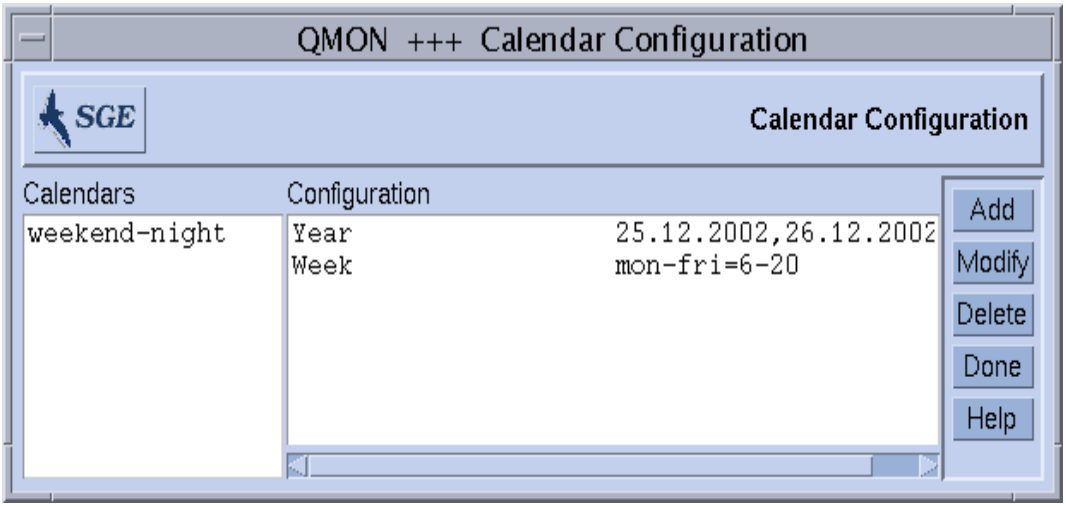


FIGURE 7-12 Calendar Configuration

Available access lists are displayed in the Calendars selection list on the left side of the screen.

2. **In the Calendars selection list, click the calendar configuration that you want to modify or delete.**
3. **Depending on how you want to change the configuration, do one of the following.**
   a. **Delete the selected calendar by pressing the Delete button on the right side of the screen.**
   b. **Modify the selected calendar by pressing the Modify button.**
   c. **Add access lists by pressing the Add button.**

   In all cases, the Calendar Definition dialogue box, similar to the example in FIGURE 7-13, is opened and provides the means to delete, modify, or add.

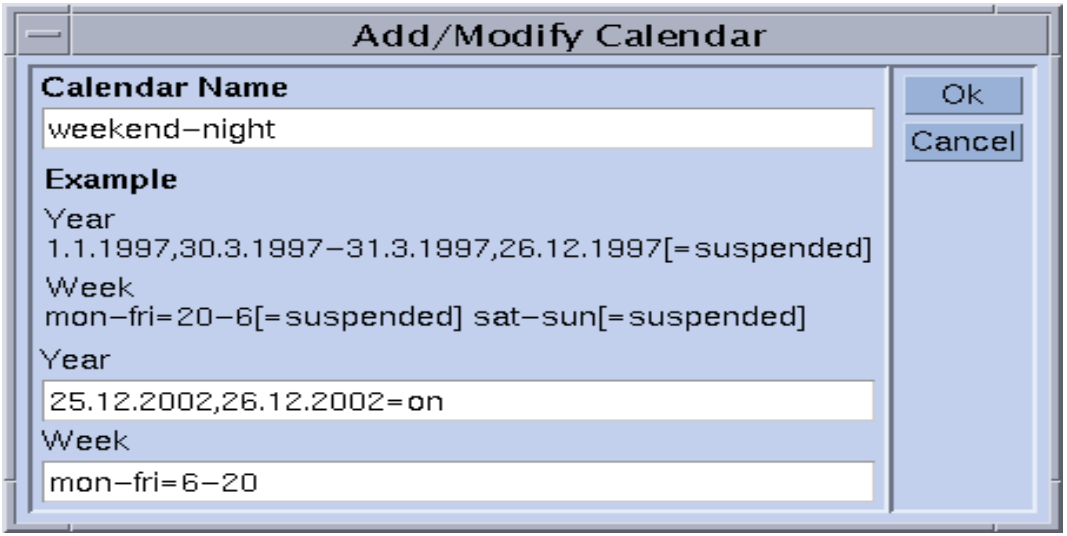


FIGURE 7-13 Add, Delete, or Modify Calendar

4. **Proceed according to the guidance in the following sections.**

   The Calendar Name input window either displays the name of the selected calendar in the case of a modify operation, or you can use it to enter the name of the calendar to be declared. The Year and Week input fields enable you to define the calendar events, using the syntax described in the `calendar_conf` man page.

   The example of the calendar configuration above is appropriate for queues that should be available outside office hours and on weekends. In addition, the Christmas holidays have been defined to be handled like weekends.

   See the `calendar_conf` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for a detailed description of the syntax and for further examples.

   By attaching a calendar configuration for a queue the availability profile defined by the calendar is set for the queue. Calendars are attached in the general parameter queue configuration menu as displayed in FIGURE 7-14. The `Calendar` input field

contains the calendar name to be attached and the icon button next to the input field opens a selection dialogue with the list of currently configured calendars. See section “About Configuring Queues” on page 163 for further details on configuring queues.

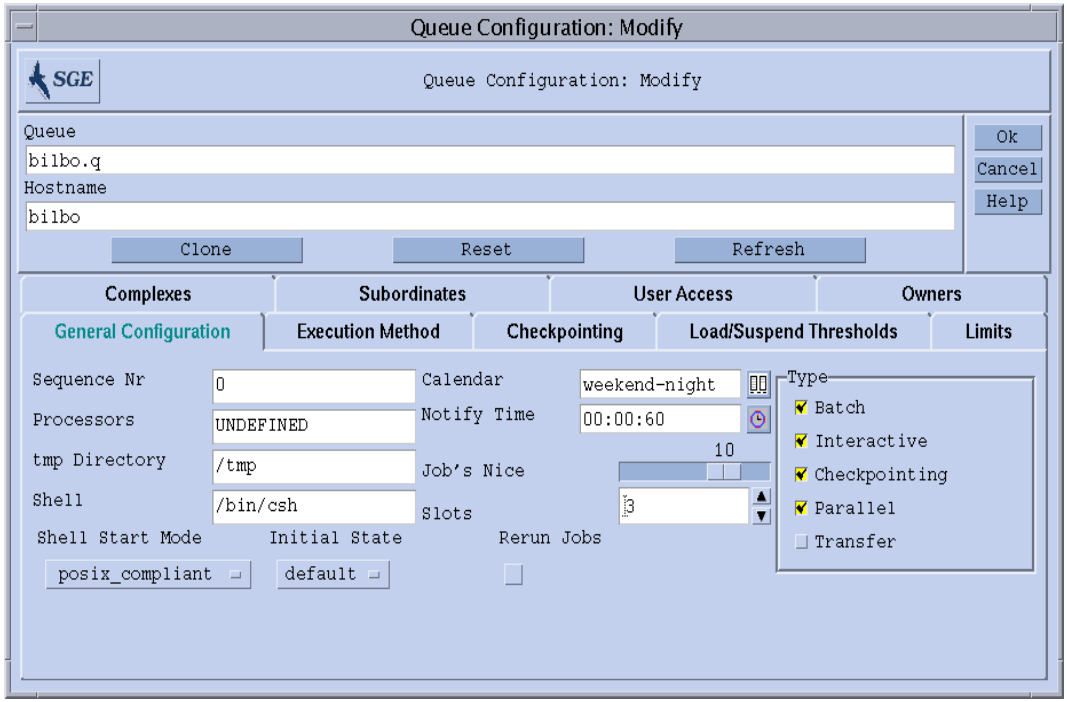


FIGURE 7-14 Calendar Configuration in General Parameters Queue Configuration Menu

▼ How To Configure Calendars From the Command Line

- **Enter the following command, with appropriate switches.**

```
% qconf switches
```

The four available switches are the following.

- `qconf -Acal, -acal`

  Add calendar – This command adds a new calendar configuration to the Sun Grid Engine cluster. The calendar to be added is either read from file or an editor with a template configuration is opened, enabling you to enter the calendar.

- `qconf -dcal`

  Delete Calendar.

- `qconf -Mcal, -mcal`

  Modify calendar – This command modifies an existing calendar configuration. The calendar to be modified is either read from file (`-Mcal`) or an editor with the previous configuration is opened, enabling you to enter the new definition (`-mcal`).

- `qconf -scal, -scall`

  Show calendar – This command displays an existing calendar configuration (`-scal`) or prints a list of all configured calendars (`-scall`).

CHAPTER 8

The Complexes Concept

This chapter explains the important Sun Grid Engine 5.3 concept known as *complexes*. In addition to background information relating to complexes and associated concepts, this chapter provides detailed instructions on how to accomplish the following tasks.

About Complexes

The definition of complexes provides all pertinent information concerning the resource attributes a user may request for a Sun Grid Engine job via the `qsub` or `qalter -l` option and for the interpretation of these parameters within the Sun Grid Engine system.

Complexes also build the framework for Sun Grid Engine system's *Consumable Resources* facility, a feature allowing for the definition of cluster global, host specific or queue related attributes which identify a resource with an associated capacity. Availability of resources in combination with the requirements of Sun Grid Engine jobs will be taken into account during the scheduling process. Sun Grid Engine will also perform the bookkeeping and capacity planning required to prevent from oversubscription of consumable resources. Examples for typical consumable attributes are available free memory, unoccupied licenses of a software package, free disk space or available bandwidth on a network connection.

In a more general sense, Sun Grid Engine complexes are used as a means for describing the intended interpretation of queue, host and cluster attributes. The description includes the attribute name, a shortcut which can be used to reference it, the value type (e.g., `STRING` or `TIME`) of an attribute, a pre-defined value being

assigned to the complex attribute, a relation operator used by the Sun Grid Engine scheduler `sge_schedd`, a requestable flag which determines whether the attribute may be requested for a job by a user or not, a consumable flag which identifies the attribute as consumable attribute if set and a default request value taken into account for consumable attributes if jobs do not explicitly specify their request for such an attribute.

The `QMON` Complex Configuration dialogue box shown in FIGURE 8-1 illustrates how complex attributes can be defined.

▼ How To Add Or Modify a Complex Configuration

1. **In the `QMON` Main menu, press the Complex Configuration button.**

   The Complex Configuration dialogue box, similar to the example in FIGURE 8-1, is displayed.

2. **Add or modify Complex configurations, guided by the information detailed in the following sections.**
   - “The Queue Complex” on page 186
   - “The Host Complex” on page 187
   - “The Global Complex” on page 189
   - “User-Defined Complexes” on page 190

The Complex Configuration dialogue box provides the means for changing the definition of the existing complexes and for defining new user complexes.

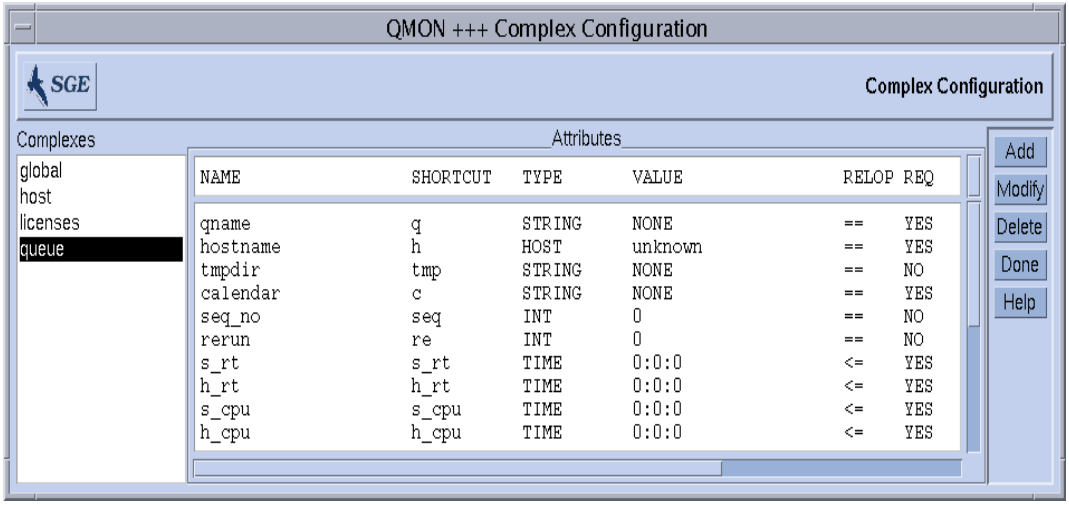


FIGURE 8-1 Complex Configuration Dialogue Box—Queue

On the left side of the screen, a selection list for all complexes known to the system is displayed. It can be used if a complex is to be modified or deleted. The desired operation (Add, Modify or Delete) can be selected with the corresponding buttons on the right side of the screen. If a new complex is to be created or an existing complex is modified, a dialogue box similar to the example in FIGURE 8-2 is opened.

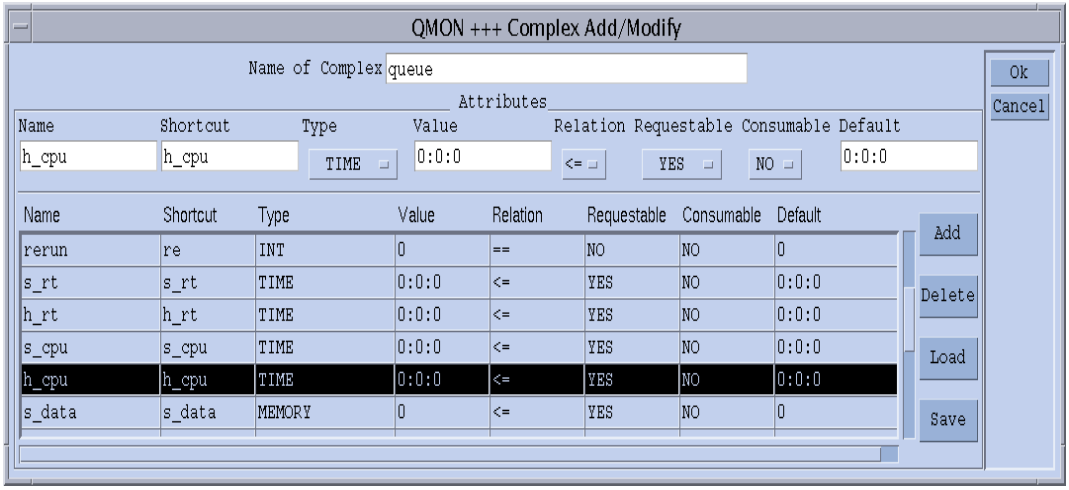


FIGURE 8-2 Complex Add/Modify Dialogue Box

You must enter the name of the complex or, if it is displayed in the Name of Complex input window at the top, select it. You can modify the complex attributes in the Complex Definition table by selecting a line with the left mouse button. The selected entry will be displayed in the definition windows and selectors at the top of the Attributes box. Changing the definition and pressing the Add button will update the changes in the definition table.

A new entry can be added by filling out the definition windows and using the selectors and then pressing the Add button. No line in the attributes table should be selected when adding new items.

The Load and Save buttons can be used to load and save complex configurations from and to regular files. A file selection box is opened to select the files. The Delete button can be used to delete selected lines in a complex configuration.

Please refer to the complex manual page for details on the meaning of the rows and columns in the table. The Ok button in the upper right corner of the screen will finally register the new/changed complex with `sge_qmaster`.

Complex Types

The Sun Grid Engine complexes object integrates four different types of complexes.

- Queue complex
- Host complex
- Global complex
- User-defined complex

The following sections describe each type in detail.

The Queue Complex

The Queue complex is referenced by the special name, `queue`.

In its default form, it contains a selection of parameters in the queue configuration as defined in `queue_conf`. The main purpose of the queue complex is to define how these parameters are to be interpreted and to provide a container for further attributes which are intended to be available for all queues. The queue complex thus can be extended by user-defined attributes.

If the queue complex is referenced in context with a particular queue, the corresponding configuration values of the queue replace the attribute values (they *overwrite* the `value` column) in the queue complex.

If, for example, the queue complex is setup for a queue called *big*, the value column for the queue complex attribute `qname`, which carries the default value `unknown` (see FIGURE 8-1), is set to big.

This implicit value setting can be overwritten by using the `complex_values` parameter in the queue configuration (see "About Configuring Queues" on page 163). This is usually done for *Consumable Resources* (see the section, "Consumable Resources" on page 194). For the virtual memory size limit, for example, the queue configuration value `h_vmem` would be used to limit the amount of total occupied memory per job, while a corresponding entry in the `complex_values` list would define the total available amount of virtual memory on a host or assigned to a queue.

If the administrator adds attributes to the queue complex, their value in association with a particular queue is either defined via the `complex_values` parameter of that queue or the `value` column in the queue complex configuration is used by default.

The Host Complex

The Host complex is referenced by the special name, `host`, and contains the characteristics definition of all attributes which are intended to be managed on a host basis (see FIGURE 8-3). The standard set of host-related attributes consists of two categories, but it may be enhanced likewise the queue complex described above. The first category is built by several queue configuration attributes which are particularly suitable to be managed on a host basis. These attributes are:

- slots
- seven
- `h_vmem`
- `s_fsize`
- `h_fsize`

(Refer to the `queue_conf` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for details).

Note – Defining these attributes in the host complex is no contradiction to having them also in the queue configuration. It allows maintaining the corresponding resources on a host level and at the same time on a queue level. Total virtual free memory (`h_vmem`) can be managed for a host, for example, and a subset of the total amount can be associated with a queue on that host.

The second attribute category in the standard host complex are the default load values. Every `sge_execd` periodically reports load to `sge_qmaster`. The reported load values are either the standard Sun Grid Engine load values such as the CPU load average or load values defined by the Sun Grid Engine administration (see the

section, “Load Parameters” on page 206*)*. The characteristics definition for the standard load values is part of the default host complex, while administrator defined load values require extension of the host complex.

The host complex commonly is not only extended to include non-standard load parameters, but also to manage host related resources such as the number of software licenses being assigned to a host or the available disk space on a host local filesystem.

If the host complex is associated with a host or a queue on that host, a concrete value for a particular host complex attribute is determined by one of the following.

- The queue configuration in the case of the queue configuration derived attributes
- A reported load value
- The explicit definition of a value in the `complex_values` entry of the corresponding host configuration (see the section, “About Configuring Hosts” on page 144)

If none of the above is available (e.g., the value is supposed to be a load parameter, but `sge_execd` does not report a load value for it), the `value` field in the host complex configuration is used.

The total free virtual memory attribute `h_vmem`, for example, 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 and attached to a queue on that host may be defined in the `complex_values` lists of that host and that queue configuration. Together with defining `h_vmem` as a *consumable resource* (see “Consumable Resources” on page 194), this allows to efficiently exploit memory of a machine without risking memory oversubscription often resulting in reduced system performance caused by *swapping*.

Note – Only the `Shortcut`, `Value`, `Relation`, `Requestable`, `Consumable` and `Default` columns may be changed for the system default load attributes. No default attributes should be deleted.

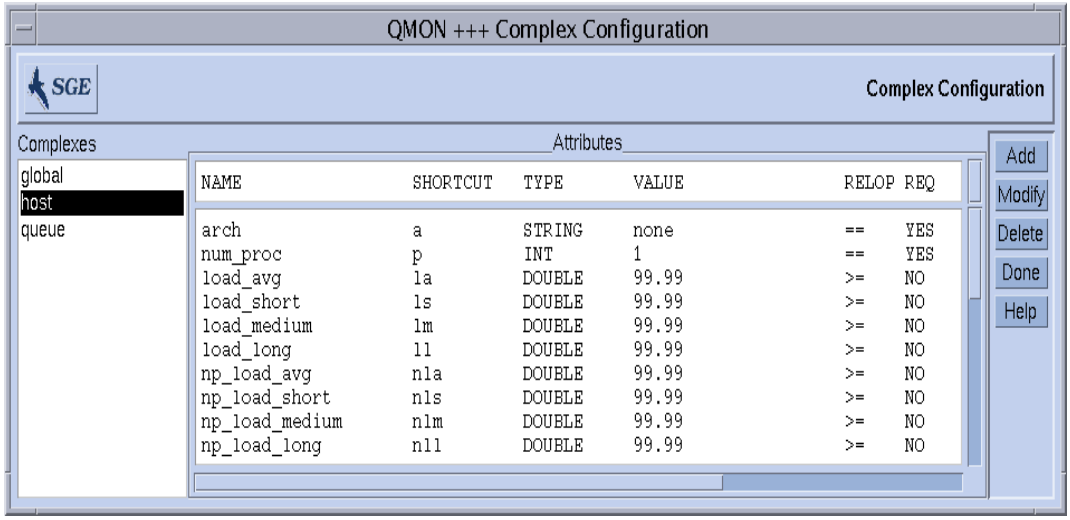


FIGURE 8-3 Complex Configuration Dialogue Box—Host

The Global Complex

The Global complex is referenced by the special complex name, `global`.

The entries configured in the global complex refer to cluster wide resource attributes, such as available network bandwidth of a file server or the free disk space on a network wide available filesystem (see FIGURE 8-4). Global resource attributes can also be associated with load reports, if the corresponding load report contains the `GLOBAL` identifier (see the section, “Load Parameters” on page 206). Global load values can be reported from any host in the cluster. There are no global load values reported by Sun Grid Engine by default and hence there is no default global complex configuration.

Concrete values for global complex attributes are either determined by global load reports, by explicit definition in the `complex_values` parameter of the `global` host configuration (see the section, “About Configuring Hosts” on page 144) or in association with a particular host or queue and an explicit definition the

corresponding `complex_values` lists. If none of the above is the case (e.g., a load value has not yet been reported), the `value` field in the global complex configuration is used.

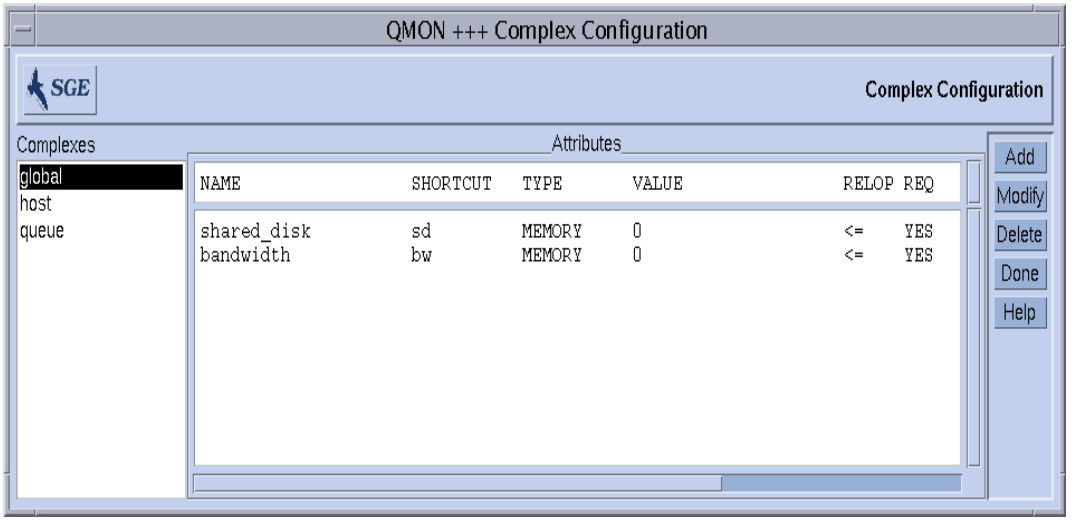


FIGURE 8-4 Complex Configuration Dialogue Box—Global

User-Defined Complexes

By setting up user-defined complexes, the Sun Grid Engine administration has the ability to extend the set of attributes managed by Sun Grid Engine while restricting the influence of those attributes to particular queues and / or hosts. A user complex is just a named collection of attributes and the corresponding definition as to how these attributes are to be handled by Sun Grid Engine. One or more of these user-defined complexes can be attached to a queue and / or host via the `complex_list` queue and host configuration parameter (see the sections, “About Configuring Queues” on page 163 and “About Configuring Hosts” on page 144). The attributes defined in all assigned complexes become available to the queue and the host respectively in addition to the default complex attributes.

Concrete values for user-defined complexes in association with queues and hosts have to be set by the `complex_values` parameter in the queue and host configuration or otherwise the `value` field of the user complex configuration is used.

As an example, let the following user-defined complex `licenses` be defined.

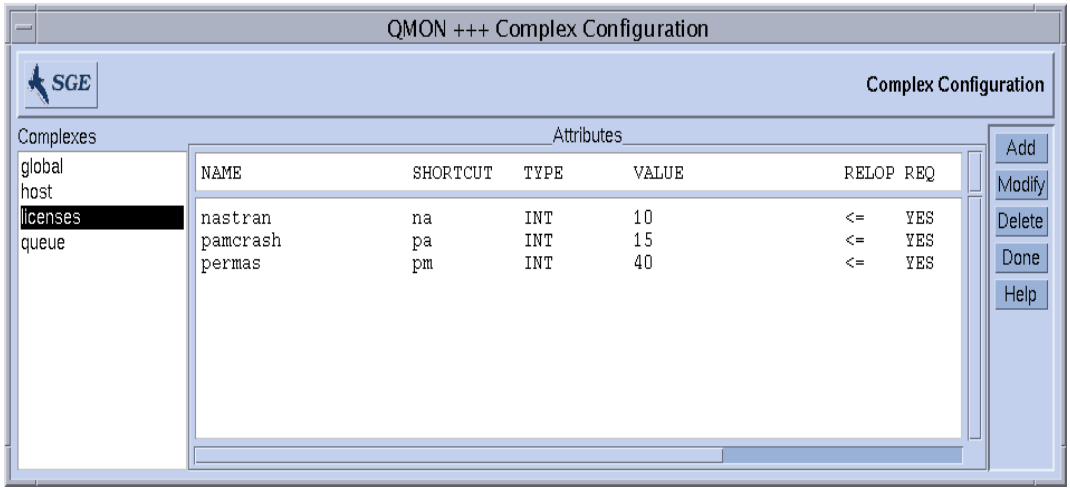


FIGURE 8-5 Complex Configuration Dialogue Box—Licenses

And let, for at least one or multiple queues, the `licenses` complex be added to the list of associated user-defined complexes as shown in the queue configuration `User Complexes` sub-dialogue box displayed in FIGURE 8-6 (see “About Configuring Queues” on page 163 and its releated sections for details on how to configure queues).

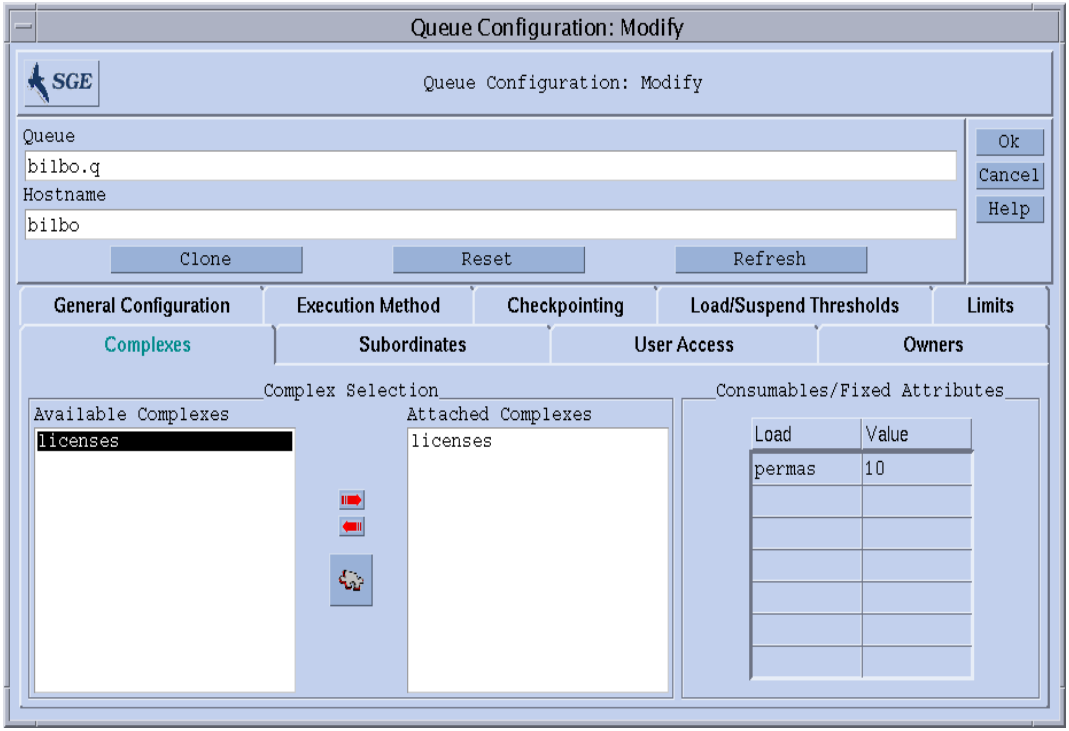


FIGURE 8-6 User-Defined Complexes Queue Configuration

Then the displayed queue is configured to manage up to 10 licenses of the software package `permas`. Furthermore, the `licenses` complex attribute `permas` becomes requestable for Sun Grid Engine jobs as expressed in the Available Resources list in the Requested Resources sub-dialogue box of the Submit dialogue box shown in FIGURE 8-7 (see Chapter 4, the section, “Submitting Jobs” on page 69 for details on how to submit jobs).

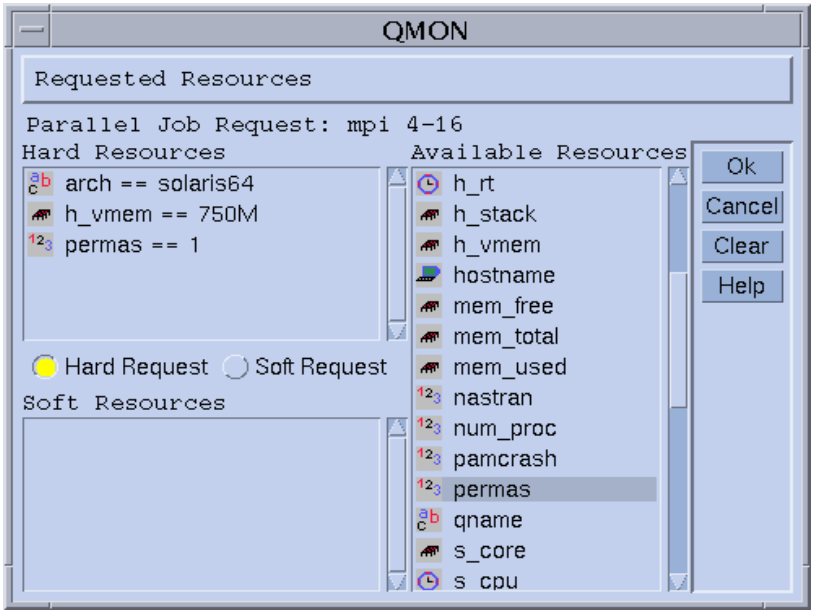


FIGURE 8-7 Requested Resources Submit Sub-Dialogue Box

Alternatively, the user could submit jobs from the command line and request `licenses` attributes as follows.

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

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

As a consequence of such a configuration and similar job requests, the only queues being eligible for these jobs would be the ones which are associated with the user-defined `licenses` complex, which have `permas` licenses configured and available.

Invalid User-Defined Complex Names

The following is a list of complex names that are reserved and thus not allowed to be designated as user-defined complex names.

- `global`
- `host`
- `queue`

Consumable Resources

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

Consumables can be combined with default or user-defined load parameters (see "Load Parameters" on page 206); i.e, load values can be reported for consumable attributes or conversely the Consumable flag can be set for load attributes. The Sun Grid Engine consumable resource management takes both the load (measuring availability of the resource) and the internal bookkeeping into account in this case, and makes sure that neither of both exceeds a given limit.

To enable consumable resource management, you must define the total capacity of a resource. This can be done on a cluster global, per host, and per queue basis while these categories may supersede each other in the given order (i.e., a host can restrict availability of a cluster resource and a queue can restrict host and cluster resources). The definition of resource capacities is performed with the `complex_values` entry in the queue and host configuration (see the `host_conf` and `queue_conf` entries in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual*, as well as "About Configuring Queues" on page 163 and "About Configuring Hosts" on page 144). The `complex_values` definition of the `global` host specifies cluster global consumable settings. To each consumable complex attribute in a `complex_values` list a value is assigned which denotes the maximum available amount for that resource. The internal bookkeeping will subtract from this total the assumed resource consumption by all running jobs as expressed through the jobs' resource requests.

▼ How To Set Up Consumable Resources

Only numeric complex attributes (those with type `INT`, `MEMORY`, and `TIME`) can be configured as consumables.

1. **In the `QMON` Main menu, press the Complex Configuration button.**

   The Complex Configuration dialogue box, similar to the example in FIGURE 8-1, is displayed.

2. **To switch on the Sun Grid Engine consumable management for an attribute, set the `CONSUMABLE` flag for it in the complex configuration as depicted in FIGURE 8-8 for the `virtual_free` memory resource, for example.**

3. **Set up other consumable resources, guided by the examples detailed in the following sections.**
   - "Example 1: Floating Software License Management" on page 196
   - "Example 2: Space Sharing for Virtual Memory" on page 200
   - "Example 3: Managing Available Disk Space" on page 203

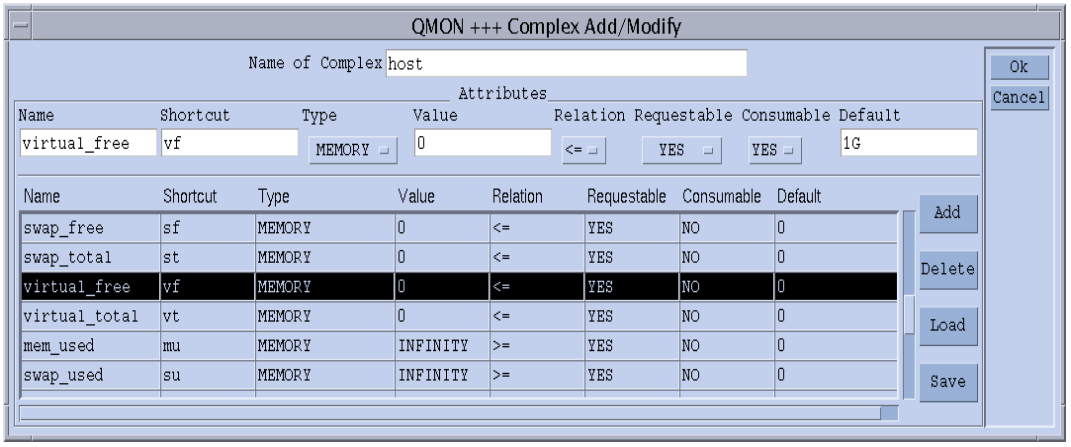


FIGURE 8-8 Complex Configuration Dialogue Box—`virtual_free`

Then, for each queue or for each host you want Sun Grid Engine to do the required capacity planning, you have to define the capacity in a `complex_values` list. An example is shown in figure FIGURE 8-9 where 1 Gigabyte of virtual memory is defined as capacity value of the current host.

The virtual memory requirements of all jobs running concurrently on that host (in any queue) will be accumulated and subtracted from the capacity of 1 Gigabyte to determine available virtual memory. If a job request for `virtual_free` exceeds the available amount, the job will not be dispatched to a queue on that host.

Note – Jobs can be forced to request a resource and thus to specify their assumed consumption via the *force* value of the Requestable parameter (see FIGURE 8-8).

Note – A default resource consumption value can be pre-defined by the administrator for consumable attributes not explicitly requested by the job (see FIGURE 8-8—200 Megabytes are set as default). This is meaningful only if requesting the attribute is not enforced, as explained above.

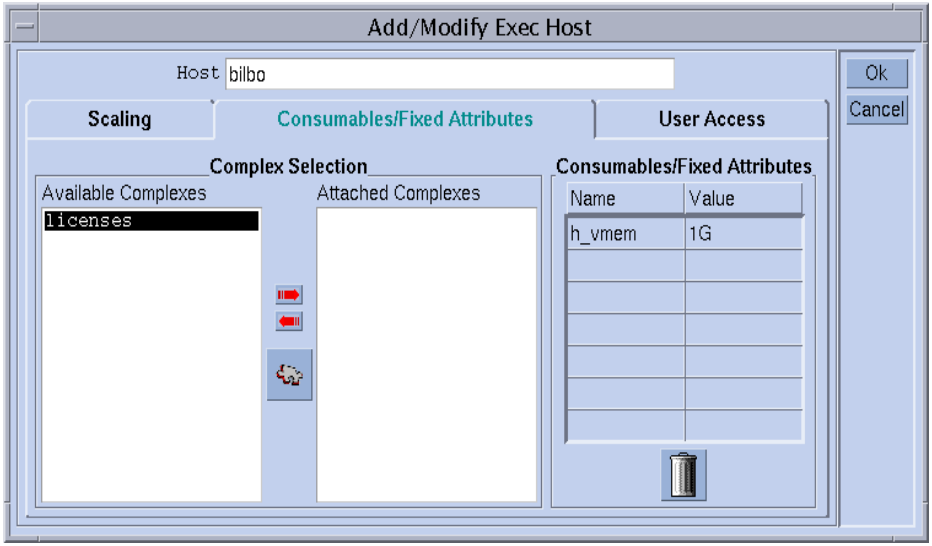


FIGURE 8-9 Execution Host Configuration—`virtual_free`

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 have the software package pam-crash in use in your cluster and you have access to 10 floating licenses; i.e., you can use `pam-crash` on every system as long as the total active invocations of the software do not exceed the number 10. The goal is to configure Sun Grid Engine in a way that prevents scheduling pam-crash jobs as long as all 10 licenses are occupied by other running pam-crash jobs.

With Sun Grid Engine consumable resources, this can be achieved easily. First, you need to add the number of available `pam-crash` licenses as a consumable resource to the Global complex configuration, as shown in FIGURE 8-10.

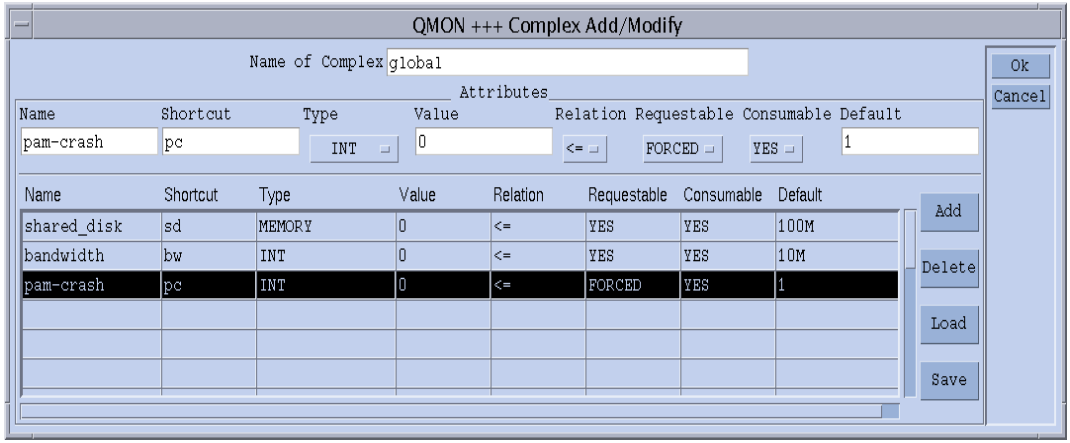


FIGURE 8-10 Complex Configuration dialogue—`pam-crash`

The name of the consumable attribute is set to `pam-crash` and `pc` can be used as a shortcut in the `qalter`, `qselect`, `qsh`, `qstat` or `qsub -l` option instead. The attribute type is defined to be an integer counter. The setting of the Value field is irrelevant for consumable resources as they receive their value from the global, host or queue configurations via the `complex_values` lists (see below). The Requestable flag is set to `FORCED` to indicate that users have to request how much pam-crash licenses their job will occupy when submitting it. The Consumable flag finally defines the attribute to be a consumable resource while the setting of `Default` is irrelevant since `Requestable` is set to `FORCED` and thus a request value will be received for this attribute with any job.

To activate resource planning for this attribute and for the cluster the number of available `pam-crash` licenses has to be defined in the global host configuration as displayed in FIGURE 8-11. The value for the attribute `pam-crash` is set to 10 corresponding to 10 floating licenses.

Note – The table `Consumable/Fixed Attributes` corresponds to the `complex_values` entry described in the host configuration file format, `host_conf`.

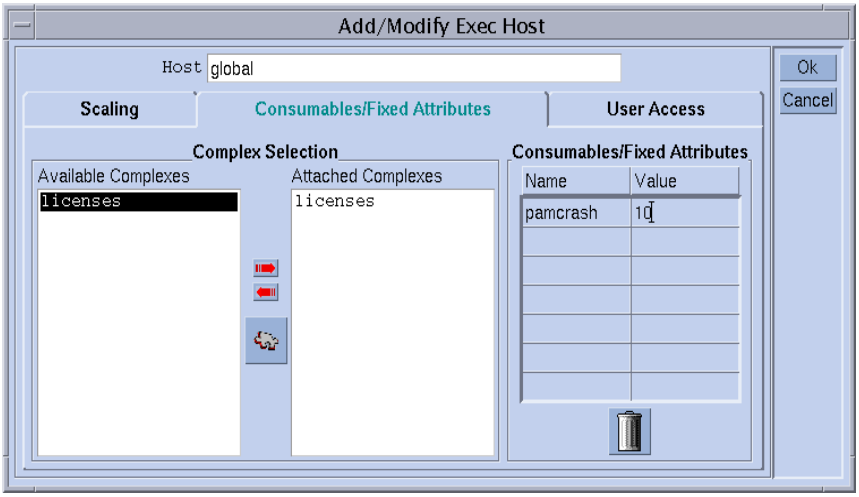


FIGURE 8-11 Global Host Configuration—`pam-crash`

Assume that a user submits the following job.

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

The job will get started only if fewer than 10 `pam-crash` licenses are currently occupied. The job may run anywhere in the cluster, however, and it will occupy one `pam-crash` license for itself throughout its run time.

If one of your hosts in the cluster cannot be included in the floating license—e.g., because you do not have `pam-crash` binaries for it—you can exclude it from the `pam-crash` license management by setting the capacity related to this host for the consumable attribute `pam-crash` to 0. This has to be done in the Execution Host Configuration Dialogue Box, as shown for host `bilbo` in FIGURE 8-12.

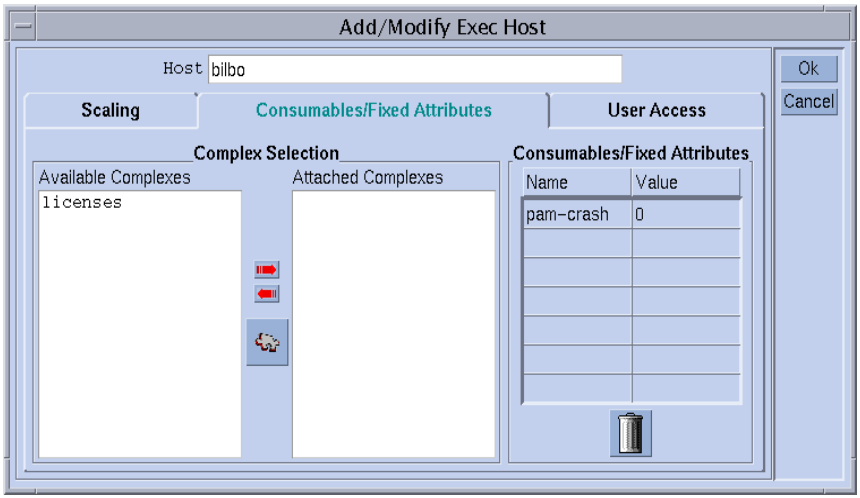


FIGURE 8-12 Execution Host Configuration—`pam-crash`

Note – The `pam-crash` attribute is implicitly available to the execution host, because the attributes of the `global` complex are inherited to all execution hosts. Likewise, by setting the capacity to 0, you could also restrict the number of licenses to be managed by a particular host as part of all licenses of the cluster to a certain non-zero value, such as 2. In this case, a maximum of 2 `pam-crash` jobs could co-exist on that host.

Similarly, you could want to prevent a certain queue from executing `pam-crash` jobs; e.g., because it is an express queue with memory and CPU-time limits not suitable for `pam-crash`. In this case, you just would have to set the corresponding capacity to 0 in the queue configuration as shown in FIGURE 8-13.

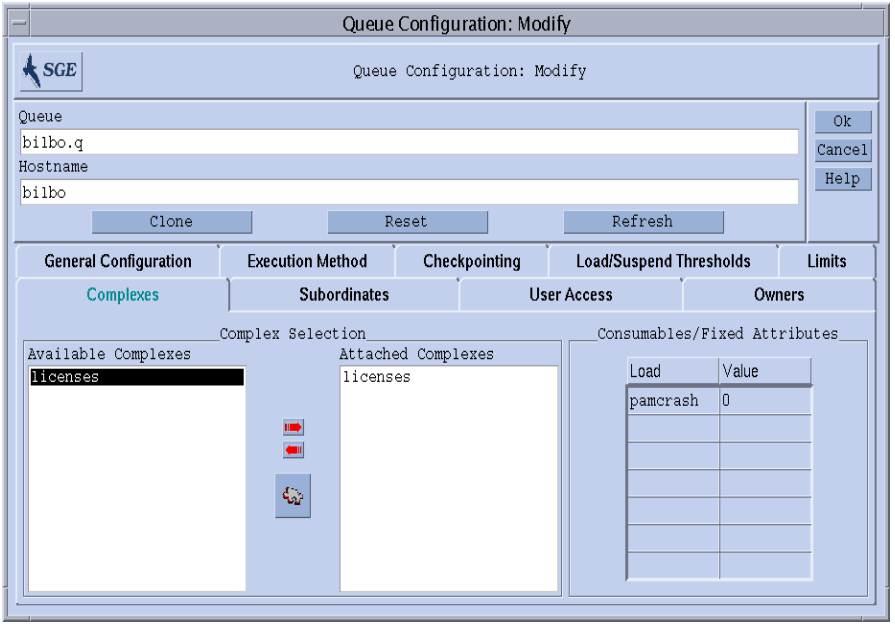


FIGURE 8-13 Queue Configuration—`pam-crash`

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

Example 2: Space Sharing for Virtual Memory

A common task for system administrators is to tune a system in a way that performance degradation caused by memory oversubscription, and consequently swapping of a machine, is avoided. Sun Grid Engine software can support you in this task via the Consumable Resources facility.

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

Sun Grid Engine software can guarantee this for all jobs started by way of it, 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 requested is not exceeded during run time.

An example for a possible host complex configuration is shown in FIGURE 8-8 and a corresponding execution host configuration for a host with 1 Gigabyte of main memory is depicted in FIGURE 8-9.

Note – The `Requestable` flag is set to `YES` in the host configuration example as opposed to `FORCED` in the previous example of a global complex configuration. This means, that users do not have to indicate the memory requirements of their jobs, but that the value in the `Default` field is used if an explicit memory request is missing. The value of 1 Gigabyte as default request in this case means, that a job without request is assumed to occupy all the available physical memory.

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

If you run a mix of different job classes with typical different memory requirements on a single machine you might wish to partition the memory of the machine for use through these job classes. This functionality, frequently called *space sharing*, can be accomplished by configuring a queue for each job class and by assigning to it a portion of the total memory on that host.

In the example, the queue configuration shown in FIGURE 8-14 would attach half of the total memory available to host `bilbo`—500 Megabytes, to the queue `bilbo.q`. Hence the accumulated memory consumption of all jobs executing in queue `bilbo.q` may not exceed 500 Megabytes. Jobs in other queues are not taken into account, but the total memory consumption of all running jobs on host `bilbo` may still not exceed 1 Gigabyte.

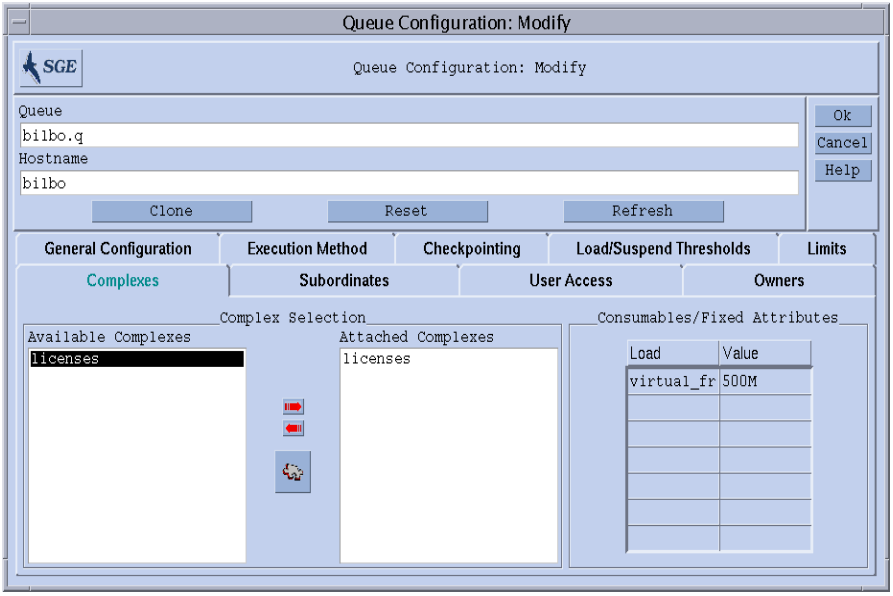


FIGURE 8-14 Queue Configuration—`virtual_free`

Note – The attribute `virtual_free` is available to all queues via inheritance from the host complex.

Users might submit jobs to a system configured similarly to the example case 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 Megabytes of memory are available and this amount will be taken into account in the capacity planning for the `virtual_free` consumable resource. The second job will only run if no other job is on the system as it implicitly request all the available memory. In addition, it will not be able to run in queue `bilbo.q` because it exceeds the queue's memory capacity.

Example 3: Managing Available Disk Space

Some applications need to manipulate huge data sets stored in files and hence depend on 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 Sun Grid Engine does not provide free disk space as one of its standard load parameters. This is due to the fact that disks are usually partitioned into file systems in a site specific way, which does not allow to identify the file system *of interest* automatically.

Nevertheless, available disk space can be managed efficiently by Sun Grid Engine via the consumables resources facility. It is recommended to use the host complex attribute `h_fsize` for this purpose for reasons explained later in this section. First, the attribute has to be configured as a onsumable resource, as shown, for example, in FIGURE 8-15.

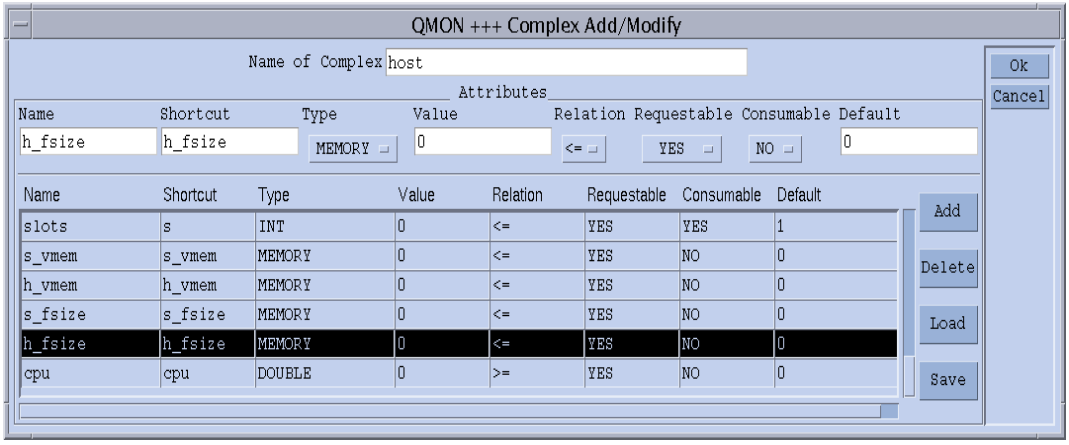


FIGURE 8-15 Complex Configuration—`h_fsize`

Assuming host local file systems, it is reasonable to put the capacity definition for the disk space consumable to the host configuration as shown in FIGURE 8-16.

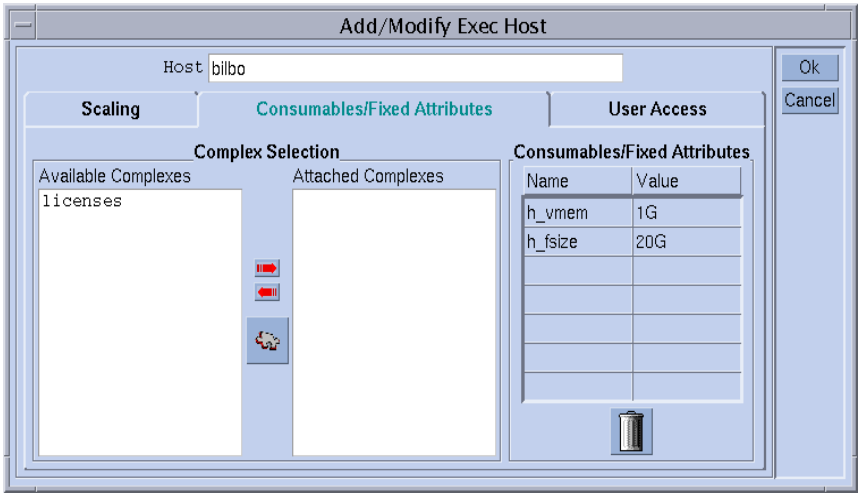


FIGURE 8-16 Execution Host Configuration—`h_fsize`

Submitting jobs to a Sun Grid Engine system configured in such a way works analogously to the previous examples:.

```
% qsub -l hf=5G big_sort.sh
```

The reason why the `h_fsize` attribute has been recommended in this example lies in the fact that `h_fsize` also is used as the *hard file size limit* in the queue configuration. The file size limit is used to restrict the ability of the jobs to create files larger than specified during job submission (20 Gigabytes in the example above) or the corresponding value from the queue configuration if the job does not request the attribute. The `Requestable` flag for `h_fsize` has been set to `FORCED` in our example, so a request will always be present.

By using the queue limit as the consumable resource, we automatically gain control on the requests as specified by the user versus the real resource consumption by the job scripts. Any violation of the limit will be sanctioned and the job eventually will be aborted (see the `queue_conf` and the `setrlimit` manual pages for details). This way it can be ensured that the resource requests, on which the Sun Grid Engine internal capacity planning is based, are reliable.

Note – Some operating systems only provide per process file size limits. In this case a job might create multiple files with a size up to the limit. On systems which support per job file size limitation, Sun Grid Engine however uses this functionality with the `h_fsize` attribute (see the `queue_conf` manual pages for further details).

If you expect applications not being submitted to Sun Grid Engine to occupy disk space concurrently, the Sun Grid Engine internal bookkeeping might not be sufficient to prevent from application failure due to lack of disk space. To avoid this problem it would be helpful to receive disk space usage statistics in a periodical fashion, which would indicate total disk space consumption including the one occurring outside Sun Grid Engine.

The Sun Grid Engine load sensor interface (see “Adding Site-Specific Load Parameters” on page 207) allows you to enhance the set of standard Sun Grid Engine load parameters with site-specific information, such as the available disk space on a particular filesystem.

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

Configuring Complexes

Sun Grid Engine complexes can either be defined and maintained graphically via the `QMON` Complex Configuration dialogue box shown and explained in the section, “How To Add Or Modify a Complex Configuration” on page 184 and following, or can be performed from the command line.

▼ How To Modify Complex Configurations From the Command Line

Enter the following command and appropriate options.

```
% qconf options
```

Refer either to the complex entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual*, or to the complex man page, for a detailed definition of the `qconf` command format and the valid value field syntax.

Useful options include the following.

- `-ac`
- `-mc`
- `-Ac`
- `-Mc`

While the `qconf -Ac` and `-Mc` options take a complexes configuration file as an argument, the `-ac` and `-mc` options bring up an editor filled in with a template complex configuration or the configuration of an existing complex for modification.

The meanings of the options follow.

- `qconf -Ac, -ac`

  Add a new complex to the list of available complexes.

- `qconf -Mc, -mc`

  Modify an existing complex.

Example of the `qconf` Command

The following command:

```
% qconf -sc licenses
```

prints the `nastran` complex (as defined in FIGURE 8-5) to the standard output stream in the file format as defined in the `complex (5)` manual page. A sample output is shown in TABLE 8-1 for the `licenses` complex.

```
#name        shortcut    type     valu     relo     requestable    consumabl     default
                                  e        p                       e
#-------------------------------------------------------------------------------
nastran      na          INT      10       <=       YES            NO            0
pam-crash    pc          INT      15       <=       YES            YES           1
permas       pm          INT      40       <=       FORCED         YES           1
#---- # start a comment but comments are not saved across edits ----------------------
```

TABLE 8-1 `qconf -sc` Sample Output

Load Parameters

This section explains the Sun Grid Engine 5.3 concept of load parameters, and includes instructions for writing your own load sensors.

The Default Load Parameters

By default, `sge_execd` periodically reports several load parameters and the corresponding values to `sge_qmaster`. They are stored in the `sge_qmaster` internal host object (see the section, “About Daemons and Hosts” on page 143). However, they are used internally only if a complex attribute with a corresponding name is defined. Such complex attributes contain the definition as to how load values have to be interpreted (see the section, “Complex Types” on page 186 for details).

After the primary installation a standard set of load parameters is reported. All attributes required for the standard load parameters are defined in the host complex. Subsequent releases of Sun Grid Engine may provide extended sets of default load parameters. Therefore, the set of load parameters being reported by default is documented in the file <*sge_root*>`/doc/load_parameters.asc`.

Note – The complex in which load attributes are defined decides about their accessibility. Defining load parameters in the global complex makes them available for the entire cluster and all hosts. Defining them in the host complex provides the attributes for all hosts but not cluster globally. Defining them in a user-defined complex allows to control visibility of the load parameter by attaching or detaching a user complex to a host.

Note – Load attributes should not be defined in queue complexes as they would be neither available to any host nor to the cluster.

Adding Site-Specific Load Parameters

The set of default load parameters may not be adequate to completely describe the load situation in a cluster, especial with respect to site specific policies, applications and configurations. Therefore, Sun Grid Engine software provides the means to extend the set of load parameters in an arbitrary fashion. For this purpose, `sge_execd` offers an interface to feed load parameters together with the current load values into `sge_execd`. Afterwards, these parameters are treated exactly like the default load parameters. Likewise for the default load parameters (see the section, “The Default Load Parameters” on page 207) corresponding attributes need to be defined in a load complex for the load parameters to become effective.

▼ How to Write Your Own Load Sensors

To feed `sge_execd` with additional load information, you must supply a *load sensor*. The load sensor may be a script or a binary executable. In either case, its handling of the standard input and output stream and its control flow must comply to the following rules:

The load sensor has to be written as infinite loop waiting 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 load data retrieval cycle 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`.

Rules

The format is as follows:

- A load value report starts with a line containing nothing but the word, `begin`.
- Individual load values are separated by new lines.
- Each load value information consists of three parts separated by colons (`:`) and containing no blanks.
- The first part of a load value information is either the name of the host for which load is reported, or the special name, `global`.
- The second part is the symbolic name of the load value, as defined in the host or global complex list (see the `complex(5)`entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for details). If a load value is reported for which no entry in the host or global complex list exists, the reported load value is not used.
- The third part is the measured load value.
- A load value report ends with a line with the word, `end`.

Example of a Script

CODE EXAMPLE 8-1 is an example of a Bourne shell script load sensor.

```
#!/bin/sh
myhost=‘uname -n‘
while [ 1 ]; do
   # wait for input
   read input
   result=$?
   if [ $result != 0 ]; then
      exit 1
   fi
   if [ $input = quit ]; then
      exit 0
   fi
   #send users logged in
   logins=‘who | cut -f1 -d" " | sort | uniq | wc -l‘ | sed "s/^ *//"
   echo begin
   echo "$myhost:logins:$logins"
   echo end
done
# we never get here
exit 0
```

CODE EXAMPLE 8-1 Bourne Shell Script Load Sensor

If this example is saved into the file `load.sh` and executable permission is assigned to it via `chmod`, you can test it interactively from the command line by invoking `load.sh` and pressing, repeatedly, the Return key of the keyboard.

As soon as the procedure works, you can install it for any execution host by configuring the path of the load sensor as the `load_sensor` parameter for the cluster, global, or the execution hos- specific configuration (see the section, “The Basic Cluster Configuration” on page 157 or the `sge_conf` manual page).

The corresponding QMON screen might look like the example in FIGURE 8-17.

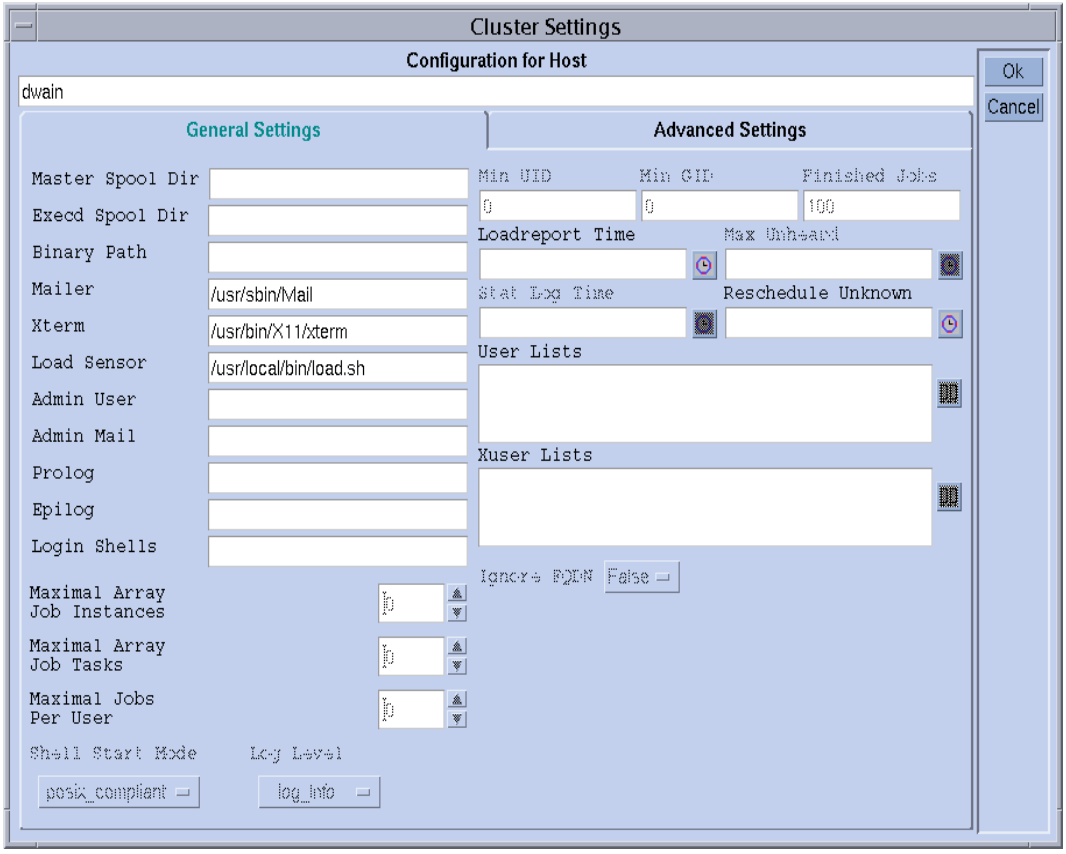


FIGURE 8-17 Local Configuration With Load Sensor

The reported load parameter, logins, will be usable as soon as a corresponding attribute is added to the hose complex. The required definition might look similar to the last table entry in FIGURE 8-18, an example of a QMON Complex Configuration screen.

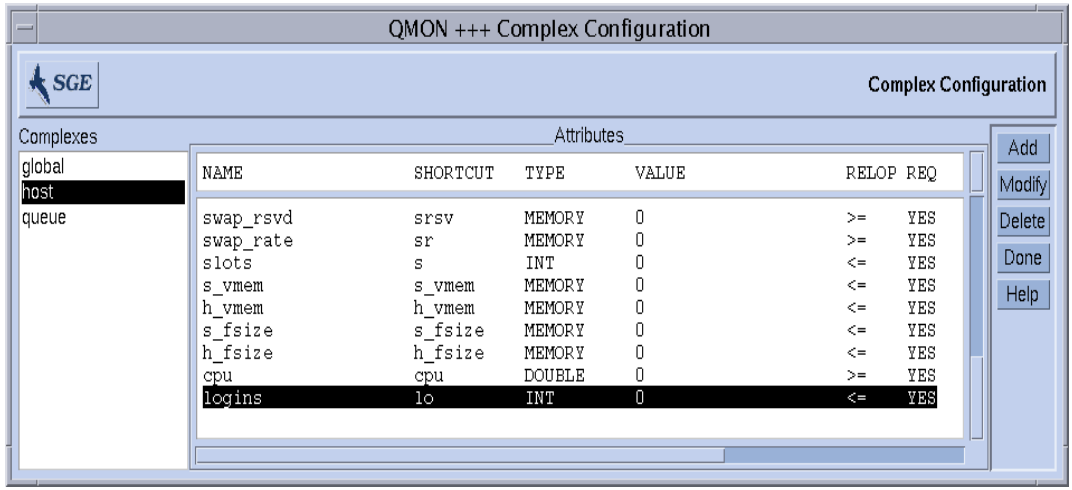


FIGURE 8-18 Complex Configuration Dialogue Box—logins

CHAPTER 9

Managing User Access and Policies

This chapter contains important information that pertains to the management of user, and related, accounts and policies in the Sun Grid Engine system. Topics in this chapter include user access, scheduling, path aliasing, default requests, accounting and utilization statistics, and support for checkpointing.

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

About Setting Up a User

The following list describes the necessary / available tasks in order to set up a user for Sun Grid Engine:

- Required Logins

  In order to submit a job from host *A* for execution on host *B*, the user has to have identical accounts (i.e., identical user names) on the hosts *A* and *B*. No login is required on the machine where `sge_qmaster` runs.

- Setting Sun Grid Engine Access Permissions

Sun Grid Engine software offers the ability to restrict user access to the entire cluster, to queues and parallel environments. See the section, “About User Access Permissions” on page 219 for a detailed description.

In addition, a Sun Grid Engine system user may get the permission to suspend or enable certain queues (see “How To Configure Owners” on page 176 for more information).

- File Access Restrictions

  Sun Grid Engine users need to have read access to the directory *<sge_root>*`/cell/common`.

  Before a Sun Grid Engine job is started, the Sun Grid Engine execution daemon (running as *root*) creates a temporary working directory for the job and changes the ownership of the directory to the job owner (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` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for more information).

  Make sure that temporary directories may be created under the `tmpdir` location, set to Sun Grid Engine user ownership and that the users may write to the temporary directories afterwards.

- Site Dependencies

  By definition, batch jobs do not have a terminal connection. Thus, UNIX commands like `stty` in the command interpreters start-up resource file (e.g. `.cshrc` for `csh`) may lead to errors. Check for occurrence and avoid such commands as described in “How To Verify the Installation” on page 48.

  As Sun Grid Engine batch jobs usually are executed off-line, there are only two methods to notify a job owner about error events and the like. One way is to log the error messages to file the other is to send electronic mail (e-mail). Under some rare circumstances (e.g., if the error log file can’t be opened) e-mail is the only way to directly notify the user (error messages like these are logged to the Sun Grid Engine system logfile anyway, but usually the user would not look into the system logfile). Therefore, it is advantageous if the electronic mail system is properly installed for Sun Grid Engine users.

- Sun Grid Engine Definition Files

  You can set up the following definition files for Sun Grid Engine users.

  - `qmon` (the resource file for the Sun Grid Engine GUI; see the section, “Customizing `QMON`” on page 9)
  - `sge_aliases` (current working directory path aliases; see the section, “About Path Aliasing” on page 233)
  - `sge_request` (default request definition file; see the section, “About Configuring Default Requests” on page 235).

About User Access

Three user categories exist in the Sun Grid Engine 5.3 system.

- **Managers** – Managers have full capabilities to manipulate Sun Grid Engine. By default, the superusers of the master host and any machine hosting a queue have manager privileges.
- **Operators** – The operators can perform many of the same commands as the manager except that they cannot add, delete, or modify queues.
- **Owners** – The queue owners are restricted to suspending/unsuspending or disabling/enabling the owned queues. These privileges are necessary for successful usage of `qidle`. Users are commonly declared to be owner of the queues residing on their desktop workstation.

Each category is described in more detail by the subsequent sections.

▼ How To Configure Accounts with QMON

1. **In the QMON Main menu, press the User Configuration button.**
2. **Depending on what you want to do, press one of the following tab selectors.**
   - Manager Account Configuration (see FIGURE 9-1)
   - Operator Account Configuration (see FIGURE 9-2)
   - User Access List Configuration (see FIGURE 9-3)
3. **Proceed according to the guidance in the following sections.**

Note – The Manager Account Configuration dialogue box is opened by default when the User Configuration button is pressed for the first time.

▼ How To Configure Manager Accounts with QMON

When you select the Manager tab, the Manager Configuration dialogue box (see FIGURE 9-1) is presented and, from there, you can declare which accounts are allowed to execute any administrative Sun Grid Engine command. The selection list in the lower half of the screen displays the accounts already declared to have administrative permission.

- **Deletion** – Delete an existing manager account from this list by clicking on its name and then by pressing the Delete button at the right side of the dialogue box.

- **Addition** – Add a new manager account by entering its name in the input window above the selection list and pressing the Add button afterwards or pressing the Return key on the keyboard.

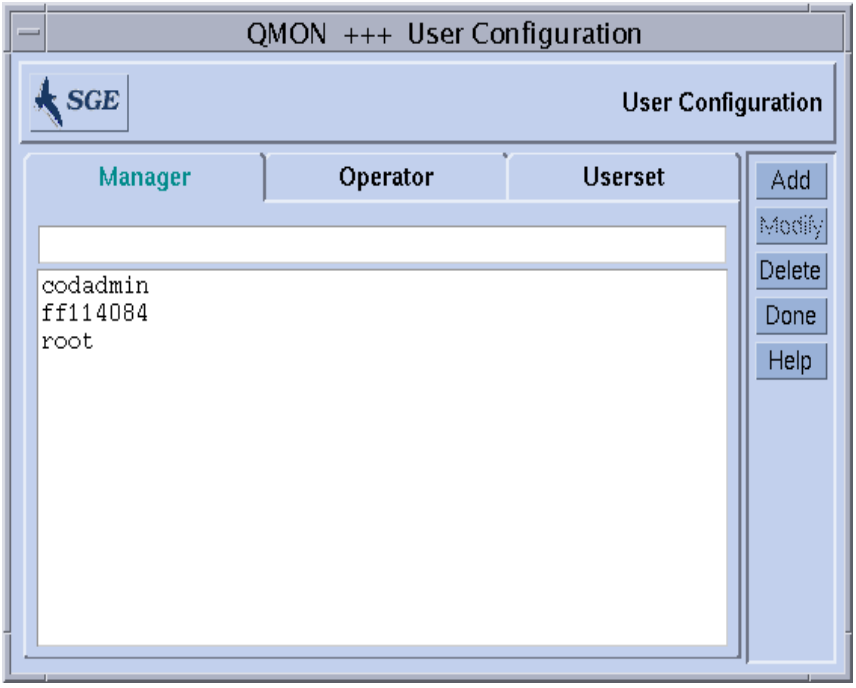


FIGURE 9-1 Manager Configuration Dialogue Box

▼ How To Configure Manager Accounts from the Command Line

- **Enter the following command with appropriate switches.**

```
# qconf switches
```

Available Switches

- `qconf -am` *user_name[,...]*

  Add manager – This command adds one or multiple users to the list of Sun Grid Engine managers. By default the root accounts of all Sun Grid Engine trusted hosts (see the section, “About Daemons and Hosts” on page 143) are Sun Grid Engine managers.

- `qconf -dm` *user_name[,...]*

  Delete manager – This command deletes the specified users from the list of Sun Grid Engine managers.

- `qconf -sm`

  Show managers – This command shows the list of all Sun Grid Engine managers.

▼ How To Configure Operator Accounts with QMON

Whe you select the Operator tab, the Operator Configuration dialogue box is presented (see FIGURE 9-2) and, from there, you can declare which accounts are allowed to have *restricted* administrative Sun Grid Engine command permission (unless they are also declared to be manager accounts—see "How To Configure Manager Accounts with QMON" on page 215). The selection list in the lower half of the screen displays the accounts already declared to provide operator permission.

- **Deletion** – Delete an existing operator account from this list by clicking on its name and then by pressing the Delete button at the right side of the dialogue box.
- **Addition** – Add a new operator account by entering its name in the input window above the selection list and pressing the Add button afterwards or pressing the Return key on the keyboard.

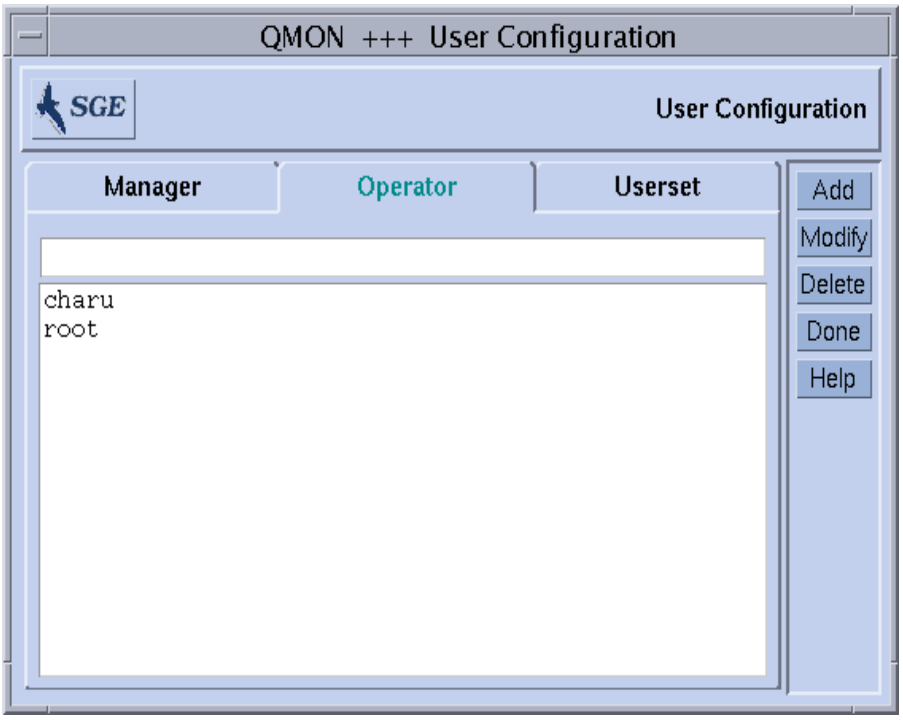


FIGURE 9-2 Operator Configuration Dialogue Box

▼ How To Configure Operator Accounts from the Command Line

- **Enter the following command with appropriate switches.**

```
# qconf switches
```

Available Switches

- `qconf -ao` *user_name[,...]*

  Add Operator – This command adds one or multiple users to the list of Sun Grid Engine operators.

- `qconf -do` *user_name[,...]*

Delete operator – This command deletes the specified users from the list of Sun Grid Engine operators.

- `qconf -so`

  Show operators – This command shows the list of all Sun Grid Engine operators.

About Queue Owner Accounts

Queue owners are defined during configuration or modifications of a Sun Grid Engine queue. Refer to sections, “How To Configure Queues with `QMON`” on page 164 and “How To Configure Queues from the Command Line” on page 177. The owner of a queue is able to do the following.

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

Note – Jobs that have been suspended explicitly while a queue was suspended will not resume execution when the queue is unsuspended. They need to be unsuspended explicitly.

Typically, users are set up to be owners of certain queues, if these users need certain machines from time to time for important work and if they are affected strongly by Sun Grid Engine jobs running in the background.

About User Access Permissions

Any user having a valid login on at least one Submit host and an Execution host has the ability to use the Sun Grid Engine system. However, Sun Grid Engine managers can inhibit access for certain users to certain or all queues. Furthermore, the usage of facilities such as specific parallel environments (see the section, “About Parallel Environments” on page 245) can be restricted as well.

For the purpose of defining access permissions, *user access lists*—which constitute named arbitrary overlapping or non-overlapping sets of users—have to be defined. User names and UNIX group names can be used to define those user access lists. The user access lists are then used in the cluster configuration (see the section, “The Basic Cluster Configuration” on page 157), in the queue configuration (see the section, “How To Configure Subordinate Queues” on page 174) or in the process of configuring parallel environment interfaces (see the section, “How To Configure PEs with `QMON`” on page 246) to either deny or allow access to a specific resource.

▼ How To Configure User Access Lists with QMON

When you select the Userset tab, the Userset Configuration dialogue box, which is similar to the example in FIGURE 9-3, is presented.

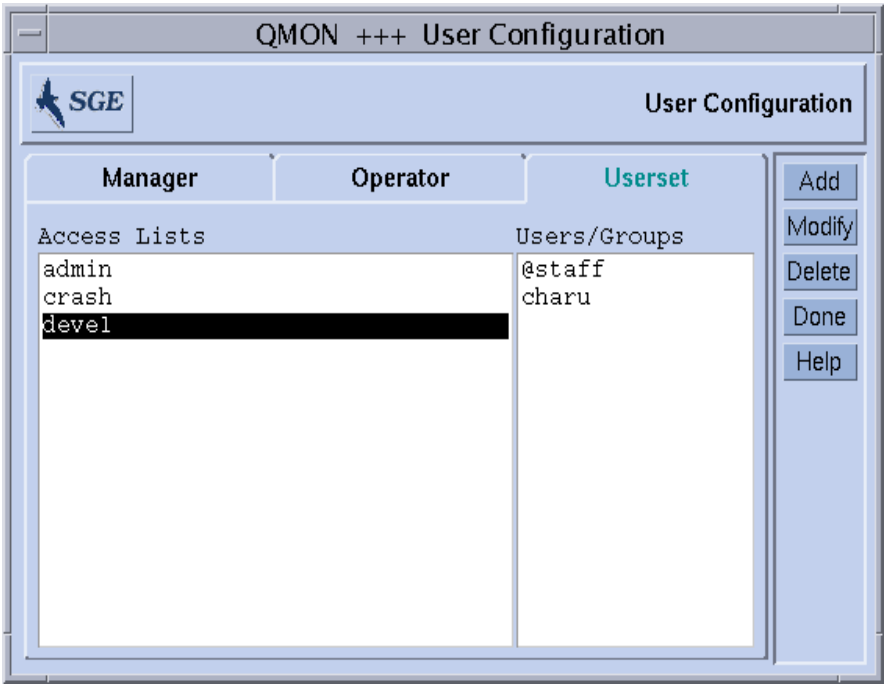


FIGURE 9-3 Userset Configuration Dialogue Box

The available access lists are displayed in the Usersets selection list on the left side of the screen. To display the content of an access list in the Users/Groups display region, click it in the Access Lists selection list.

Note – Groups are differentiated from users by a prefixed @ sign.

You use the Userset Configuration dialogue box to perform the following tasks.

- **Deletion** – Delete an existing access list from the Userset selection list by clicking on its name and then by pressing the Delete button at the right side of the dialogue box.
- **Addition** – Add a new access list by entering its name in the input window above the selection list and pressing the Add button afterwards or pressing the Return key on the keyboard.
- **Modification** – Modify a selected access list by pressing the Modify button.

In all cases, the Access List Definition dialogue box, similar to the one displayed in FIGURE 9-4, is opened and provides the corresponding means.

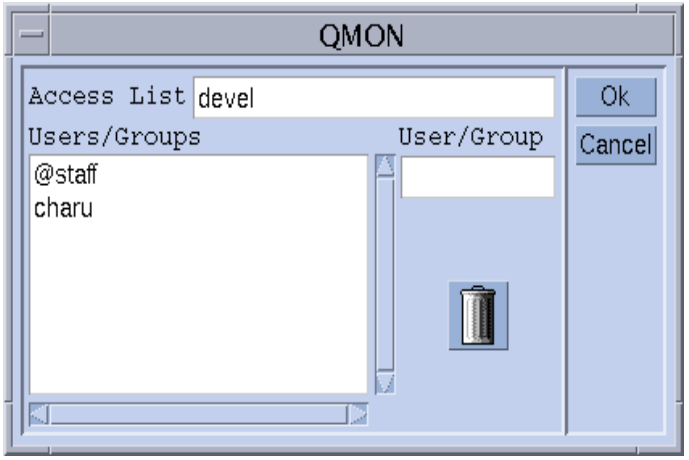


FIGURE 9-4 Access List Definition Dialogue Box

Explanation of the Access List Definition Dialogue Box Windows

- **Userset Name input window** – Displays either the name of the selected access list in the case of a Modify operation, or you can use it to enter the name of the access list to be declared.
- **Users/Groups display region** – Contains the access list entries as defined so far.
- **User/Group input window** – Must be used to add new entries to the access list.

The entered user or group names (groups are prefixed by the @ sign) are appended to the Users/Groups display region after you press the Return key on the keyboard. You can delete entries by selecting them in the display region and then pressing the garbage bin icon button.

The modified or newly defined access lists are registered as soon as you press the Ok button, or they are discarded if you press the Cancel button. In both cases, the Access List Definition dialogue box is closed.

▼ How To Configure User Access Lists from the Command Line

- **Enter the following command with appropriate options.**

```
# qconf switches
```

Available Options

- `qconf -au` *user_name[,...] access_list_name[,...]*

  Add user—This command adds one or more users to the specified access list(s).

- `qconf -du` *user_name[,...] access_list_name[,...]*

  Delete user—This command deletes one or more users from the specified access list(s).

- `qconf -su` *access_list_name[,...]*

  Show user access list—This command displays the specified access lists.

- `qconf -sul`

  Show user access lists—This command prints a listing of all access lists currently defined.

About Scheduling

The Sun Grid Engine system's job-scheduling activities comprise the following.

- Pre-dispatching decisions—These are activities such as eliminating execution queues because they are full or overloaded and spooling jobs currently not eligible for execution in a waiting area.
- Dispatching—These activities involve deciding a job's importance with respect to all other pending and running jobs, sensing the load on all the machines in the cluster, and sending the job to an execution queue on a machine selected according to the configured selection criteria.

Sun Grid Engine software schedules jobs across a heterogeneous cluster of computers based on the following.

- The cluster's current load
- The jobs' resource requirements (e.g., CPU, memory, and I/O bandwidth)

Scheduling decisions are based on the strategy for the site and the instantaneous load characteristics of each computer in the cluster. A site's scheduling strategy is expressed through the Sun 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 Sun Grid Engine scheduling tasks.

- **Queue sorting**—The software ranks the queues in the cluster according to the order in which the queues should be filled up.
- **Job sorting**—This determines the order in which the Sun Grid Engine system attempts to schedule jobs.

Queue Sorting

The following means are provided to determine the order in which Sun Grid Engine attempts to fill up queues.

- **Load reporting**—Sun Grid Engine administrators can select which load parameters are used to compare the load status of hosts and their queues. The wide variety of standard load parameters being available and an interface for extending this set with site-specific load sensors are described in the section, "Load Parameters" on page 206.
- **Load scaling**—Load reports from different hosts can be normalized to reflect a comparable situation (see the section, "How To Configure Execution Hosts with QMON" on page 148).
- **Load adjustment**—Sun Grid Engine software can be configured to automatically correct the last reported load as jobs are dispatched to hosts. The corrected load will represent 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 shows effect.
- **Sequence number**—Queues can be sorted following a strict sequence.

Job Sorting

Before Sun Grid Engine starts dispatching, jobs are brought into an order of highest priority first. Sun Grid Engine will then attempt 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 of control over the job order.

- **User sort** – If this scheduling alternative is in effect jobs of different users are interleaved. I.e., the first jobs all users have submitted are treated equally, then the second, and so on.
- **Job priority –** Administrators can assign a priority number to a job,thereby directly determining the sorting order. A user can lower the priority assigned to his or her own jobs.
- **Maximum number of user/group jobs** – The maximum number of jobs a user or a UNIX user group can have running in the Sun Grid Engine system concurrently can be restricted. This will influence the pending job list sorting order, because jobs of users not exceeding their limit will be given preference.

What Happens in a Scheduler Interval

The Scheduler schedules work in intervals. Between scheduling actions Sun Grid Engine keeps information about significant events such as job submittal, job completion, job cancellation, an update of the cluster configuration, or registration of a new machine in the cluster. When scheduling occurs, the scheduler does the following.

- Takes into account all significant events.
- Sorts jobs and queues corresponding to the administrator specifications.
- Takes into account all jobs' resource requirements.

Then, as needed, the Sun Grid Engine system does the following.

- Dispatches new jobs.
- Suspends executing jobs.
- Maintains the status quo.

Scheduler Monitoring

If a job does not get started and if the reasons are unclear to you, you can execute `qalter` for the job together with the `-w v` option. Sun Grid Engine software will assume an empty cluster and will check whether there is any queue available which is suitable for the job.

Further information can be obtained by executing `qstat -j` *job_id*. It will print a summary of the job's request profile containing also the reasons why the job was not scheduled in the last scheduling run. Executing `qstat -j` without a job ID will summarize the reasons for all jobs not having been scheduled in the last scheduling interval.

Note – Collection of scheduling reason information has to be switched on in the scheduler configuration `sched_conf`. Refer to either the `schedd_job_info` parameter in the corresponding *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* or the section, "How To Change the Scheduler Configuration with `QMON`" on page 229.

To retrieve even further detail about the decisions of the Sun Grid Engine scheduler `sge_schedd`, you can use the `-tsm` option of the `qconf` command. This command will force `sge_schedd` to write trace output to the file.

Default Scheduling

The default Sun Grid Engine scheduling is a *first-in-first-out* policy; i.e., the first job submitted is the first the scheduler examines in order to dispatch it to a queue. If the first job in the list of pending jobs finds a suitable and idle queue it will be started first in a scheduler run. Only if the first job fails to find a suitable free resource the second job or a job ranked behind may be started before the first in the pending jobs list.

As far as the queue selection for jobs is concerned, the default Sun Grid Engine strategy is to select queues on the least loaded host as long as they deliver suitable service for the job's resource requirements. If multiple suitable queues share the same load the queue being selected is unpredictable.

Scheduling Alternatives

There are various ways to modify the job scheduling and queue selection strategy.

- 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

Following sections explore these alternatives in detail.

Changing the Scheduling Algorithm

The scheduler configuration parameter `algorithm` (see the `sched_conf` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for further information) is designed to provide a selection for the scheduling algorithm in use. Currently, `default` is the only allowed setting.

Job Priorities

The Sun Grid Engine administration may assign an integer number called job priority to a job spooled in the pending jobs list. The job priority defines the job's position in the pending jobs list. The job with the highest priority number will be examined first by the scheduler. The value range for job priorities is between -1024 and 1023 with 0 being the priority for new jobs just submitted. If a negative priority value is assigned to a job, the job is sorted even behind new jobs just submitted. If multiple jobs with the same priority number exist the default first-in-first-out rule applies within this priority class.

Job priorities are assigned to a job via the following command.

```
% qalter -p prio job_id ...
```

In the command above, `prio` specifies the priority to be assigned to the list of jobs as specified in the trailing blank separated Job Id list.

Note – The term job priorities should not be mixed up with the priority queue configuration parameter (see the `queue_conf` manual page in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual*) which defines the `nice` value being set for all jobs executed in a particular queue. Also note that the second column in the `qstat` output shows the priorities currently assigned to the submitted jobs.

Equal Share Sort

The default *first-in-first-out* scheduling policy described above is well known to yield rather unfair results if a user submits a series of jobs one after each other in a short time (e.g. by use of a shell script procedure). The jobs of this user would cover the suitable resources for a very long time offering no chance for other users to allocate these queues.

In this case, the cluster administration may change the scheduling policy to the so called *equal share sort*. If this scheduling alternative is in effect and a user already has a running job in the system all his other jobs are sorted behind the jobs of other users in the same priority class (see the previous section for details about priority classes).

The equal share sort is turned on if the scheduler configuration parameter `user_sort` is set to `TRUE` (see the `sched_conf` manual page in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual*).

Scaling System Load

The Sun Grid Engine system uses the system load information on the machines hosting queues to select the executing queue for a job. This queue selection scheme builds up a load balanced situation thus guaranteeing better utilization of the available resources in a cluster.

However, the system load may not always tell the truth. If, for example, a multi CPU machine is compared to a single CPU system the multiprocessor system usually reports higher load figures as it most probably runs more processes and 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 which are reported by default by `sge_execd` (see the section, “Load Parameters” on page 206 and the <*sge_root*>`/doc/load_parameters.asc` file for details). Use these load parameters instead of the raw load values to avoid the problem described above.

Another example for potentially improper interpretation of load values are systems with strong differences in their performance potential or in their price performance ratio for both of which equal load values do not mean that arbitrary hosts can be selected to execute a job. In this kind of situation, the Sun Grid Engine administrator should define load scaling factors for the concerning execution hosts and load parameters (see “How To Configure Execution Hosts with `QMON`” on page 148, and related sections).

Note – The scaled load parameters are also used to compare them against the load threshold lists *load_thresholds* and *migr_load_thresholds* (see the `queue_conf` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for details).

A further problem associated with load parameters is the need for an application and site dependent interpretation of the values and their relative importance. The CPU load may be dominant for a certain type of application which is common at a particular site, while the memory load is much more important for another site and for the application profile to which the site’s compute cluster is typically dedicated to. To address this problem, Sun Grid Engine allows the administrator to specify a so called *load formula* in the scheduler configuration file, `sched_conf` (refer to the corresponding *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* section for more detail). Site-specific information on resource utilization and capacity planning can be taken into account by using site defined load parameters (see the section, “Adding Site-Specific Load Parameters” on page 207) and consumable resources (see the section, “Consumable Resources” on page 194) in the load formula.

Finally, the time dependency of load parameters needs to be taken into account. The load, which is imposed by the Sun Grid Engine jobs running on a system varies in time, and often—for example, for the CPU load—requires some amount of time to be reported in the appropriate quantity by the operating system. Consequently, if a job was started very recently, the reported load may not provide a sufficient representation of the load which is already imposed on that host by the job. The reported load will adapt to the real load over time, but the period of time, in which the reported load is too low, may already lead to an oversubscription of that host. Sun Grid Engine allows the administrator to specify *load adjustment* factors which are used in the Sun Grid Engine scheduler to compensate for this problem. Refer to the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* dealing with the scheduler configuration file `sched_conf` for detailed information on how to set these load adjustment factors.

Selecting Queue by Sequence Number

Another way to change the default queue selection scheme is to set the global Sun Grid Engine cluster configuration parameter `queue_sort_method` to `seq_no` instead of the default `load` (see the `sched_conf` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual*). In this case, the system load is no longer the primary method used to select queues. Instead, the sequence number—as assigned to the queues by the queue configuration parameter `seq_no` (see the `queue_conf` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual*)—is used to define a fixed order between the queue in which they are selected (if they are suitable for the considered job and if they are free).

This queue selection policy may be useful if the machines offering batch services at your site are ranked in a monotonous price per job order: e.g., a job running on machine A costs 1 unit of money while it costs 10 units on machine B and 100 units on machine C. Thus the preferred scheduling policy would be to first fill up host A then host B and only if no alternative remains use host C.

Note – If you have changed the method of queue selection to `seq_no`, and the considered queues all share the same sequence number, then queues will be selected by the default `load`.

Restricting the Number of Jobs per User or Group

The Sun Grid Engine administrator may assign an upper limit to the number of jobs which are allowed to be run by any user or any UNIX group at any point of time. In order to enforce this feature, set the `maxujobs` and / or `maxgjobs` as described in the `sched_conf` section of the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual*.

▼ How To Change the Scheduler Configuration with QMON

1. **From the QMON Main menu, click Scheduler Configuration.**

   The Scheduler Configuration dialogue box is presented. The dialogue box is separated into the General Parameters section and the Load Adjustment section. You select either one, depending on what you want to accomplish.

a. To change general scheduling parameters, click the General Parameters tab.

The General Parameters Dialogue box is similar to the example in FIGURE 9-5.

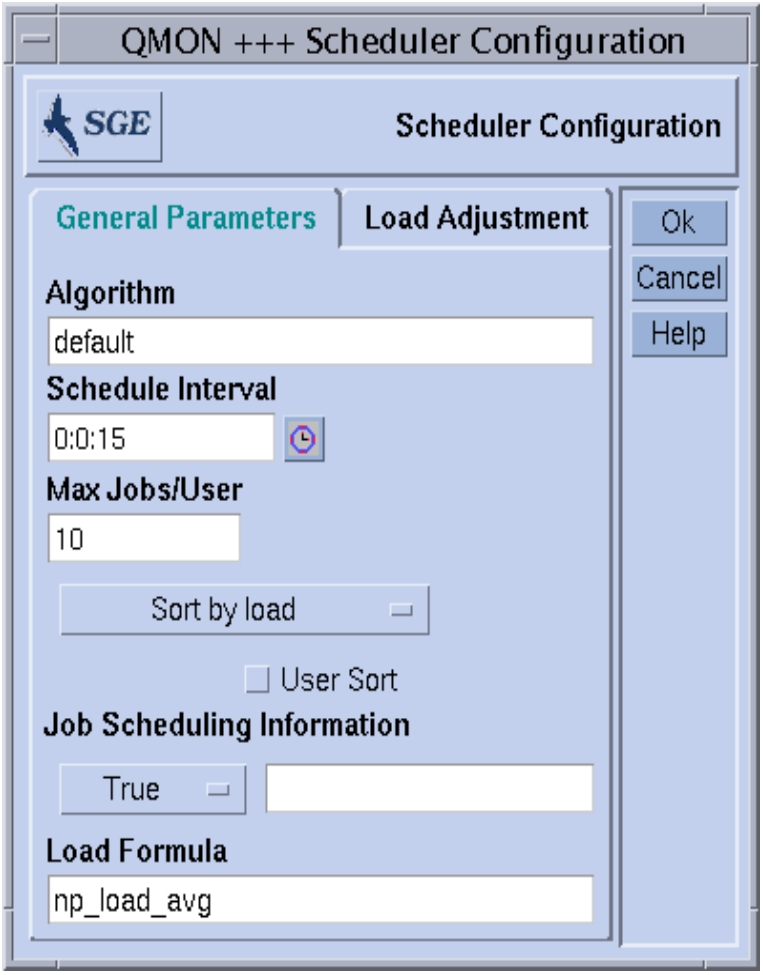


FIGURE 9-5 Scheduler Configuration Dialogue Box—General Parameters

From the General Parameters dialogue box, you can set the following parameters.

- The scheduling algorithm (see “Changing the Scheduling Algorithm” on page 225)
- The regular time interval between scheduler runs
- The maximum number of jobs allowed concurrently to run per user and per UNIX group (see “Restricting the Number of Jobs per User or Group” on page 228).

- The queue sorting scheme—either sorting by load or sorting by sequence number (see “Selecting Queue by Sequence Number” on page 228).
- Whether Equal Share Sort (`User Sort` flag) is activated (see the section, “Equal Share Sort” on page 226).
- Whether job scheduling information is accessible through `qstat -j` or whether this information should only be collected for a range of job IDs specified in the attached input field. It is recommended to switch on general collection of job scheduling information only temporarily in case of extremely high numbers of pending jobs.
- The load formula to be used to sort hosts and queues

b. To change load adjustment parameters, select the Load Adjustment tab.

The Load Adjustment Parameters dialogue box is similar to the example in FIGURE 9-6.

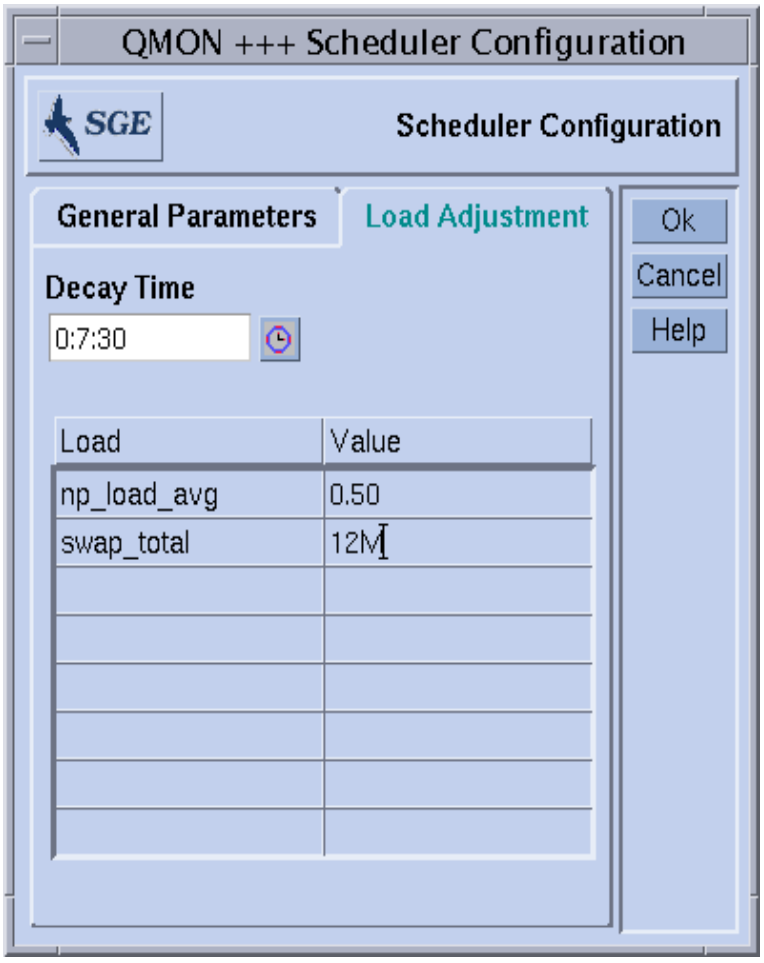


FIGURE 9-6 Scheduler Configuration Dialogue Box—Load Adjustment

The Load Adjustment dialogue box enables you to define the following parameters.

- The load adjustment decay time
- A table of load adjustment values in the lower half of the dialogue enlisting all load and consumable attributes for which an adjustment value currently is defined. The list can be enhanced by clicking to the Load or Value button at the top. This will open a selection list with all attributes attached to the hosts (i.e., the union of all attributes configured in the global, the host and the administrator-defined complexes). The Attribute Selection dialogue box is shown in FIGURE 6-6. Selecting one of the attributes and confirming the selection with the OK button will add the attribute to the Load column of the Consumable/Fixed Attributes

table and will add the pointer to the corresponding Value field. Modify an existing value can by double-clicking the Value field. Delete an attribute by selecting the corresponding table line and then typing CTRL-D—or by clicking the *right* mouse button to open a deletion box and then confirming the deletion.

See “Scaling System Load” on page 227 for background information. Refer to the `sched_conf` manual page in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for further detail on the scheduler configuration.

About Path Aliasing

In Solaris and other networked UNIX environments, a user very often has the same home directory (or part of it) on different machines if it has been made accessible across NFS. However, sometimes the home directory path is not exactly the same on all machines.

For example, consider user home directories being available via NFS and *automounter*. If a user has a home directory, `/home/foo`, on the NFS server, he will be able to access the home directory under this path on all properly installed NFS clients running automounter. However, it is important to notice that `/home/foo` on a client will be just a *symbolic link* to `/tmp_mnt/home/foo`, the actual location on the NFS server from where automounter physically mounts the directory.

If, in such a situation, the user would submit a job on a client from somewhere within the home directory tree, accompanying it with the `qsub -cwd` flag (execute job in current working directory), the Sun Grid Engine system could be presented with a problem trying to locate the current working directory on the execution host—if that host is the NFS server. This is because the `qsub` command will reach the current working directory on the submit host and will get `/tmp_mnt/home/foo/`—as this is the physical location on the submit host. This path will be passed over to the execution host and cannot be resolved if the execution host is the NFS server with a physical home directory path of `/home/foo`.

Other occasions usually causing similar problems are fixed (non-automounted) NFS mounts with different mount point paths on different machines (e.g., mounting home directories under `/usr/people` on one host and `/usr/users` on another) or symbolic links from outside into a network-available file system.

To prevent such problems, Sun Grid Engine software enables both the administrator and the user to configure a *path aliasing file*. The locations of two such files follow.

- *<sge_root>*`/`*<cell>*`/common/sge_aliases`—This is a cluster global path-aliasing file.
- `$HOME/.sge_aliases`—This ia a user-specific path-aliasing file.

Note – Only the qualified administrator should modify the cluster global file.

File Format

Both files share the same format.

- Blank lines and lines with a # sign in the first column 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 and the execution host entries may consist of only a * sign, which matches any host.

How Path-Aliasing Files Are Interpreted

The files are interpreted as follows.

- After `qsub` has retrieved the physical current working directory path, the cluster global path-aliasing file is read, if present. The user path-aliasing file is read afterwards, as if it were appended to the global file.
- Lines not to be skipped are read from the top of the file, one by one, while the translations specified by those lines are stored, if necessary.
- A translation is stored only if the submit host entry matches the host on which the `qsub` command is executed, and if the source path forms the initial part either of the current working directory or of the source path replacements already stored.
- As soon as both files are read, the stored path-aliasing information is passed along with the submitted job.
- On the execution host, the aliasing information will be evaluated. The leading part of the current working directory will be replaced by the source path replacement if the execution host entry of the path alias matches the executing host. Note that the current working directory string will be changed in this case, and that subsequent path aliases must match the replaced working directory path to be applied.

Example of a Path-Aliasing File

is an example how the NFS/*automounter* problem described above can be resolved with an aliases file entry.

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

Example of Path-Aliasing File

About Configuring Default Requests

Batch jobs are normally assigned to queues by the Sun Grid Engine system with respect to a request profile defined by the user for a particular job. The user assembles a set of requests which need to be met to successfully run the job and the Sun Grid Engine scheduler only considers queues satisfying the set of requests for this job.

If a user doesn't specify any requests for a job, the scheduler will consider any queue the user has access to without further restrictions. However, Sun Grid Engine software allows for configuration of *default requests* which may define resource requirements for jobs even though the user did not specify them explicitly.

Default requests can be configured globally for all users of a Sun Grid Engine cluster, as well as privately for any user. The default request configuration is represented in *default request files*. The *global request file* is located under *<sge_root>/<cell>*`/common/sge_request`, while the *user-specific request file*, called `.sge_request`, can be located in the user's home directory or in the current working directory in which the `qsub` command is executed.

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 line have higher precedence as the requests in the default request files (see Chapter 4 for details on how to request resources for jobs explicitly).

Note – Unintended influence of the default request files can be prohibited by use of the `qsub -clear` option, which discards any previous requirement specifications.

Format of Default Request Files

The format of both the local and the global default request files are described in the following list.

- The default request files may contain an arbitrary number of lines. Blank lines and lines with a # sign in the first column are skipped.
- Each line not to be skipped may contain any `qsub` option, as described in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual*. More than one option per line is allowed. The batch script file and argument options to the batch script are not considered as `qsub` options and thus are not allowed in a default request file.
- The `qsub -clear` option discards any previous requirement specifications in the currently evaluated request file or in request files processed before.

Example of a Default Request File

As an example, suppose a user's local default request file is configured the same as `test.sh`, the script in CODE EXAMPLE 9-1.

```
# 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
```

CODE EXAMPLE 9-1 Example of Default Request File

To execute the script, the user would enter the following command.

```
% qsub test.sh
```

The effect of executing the `test.sh` script would be the same as if the user had 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 via `qsub`, interactive jobs submitted via `qsh` will consider default request files also. Interactive or batch jobs submitted via `QMON` will also take respect to these request files.

About Gathering Accounting and Utilization Statistics

The Sun Grid Engine command, `qacct`, can be used to generate alphanumeric accounting statistics. If invoked without switches, `qacct` displays the aggregate utilization on all machines of the Sun Grid Engine cluster as generated by all jobs having finished and being contained in the cluster accounting file, *<sge_root>*/*<cell>*/`common/accounting`. In this case, `qacct` just reports three times in seconds:

- REAL—This is the wallclock time; i.e., the time between the job's start and finishe.
- USER—This is the CPU time spent in the user processes.
- SYSTEM—This is the CPU time spent in system calls.

Several switches are available to report accounting information about all or certain queues, all or certain users, and the like. It is possible, in particular, to request information about all jobs having completed and matching a resource requirement specification expressed with the same `-l` syntax as used with the `qsub` command to submit the job. Refer to the `qacct` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for more information.

A `qacct` option—`-j` *[job_id|job_name]*—provides direct access to the complete resource usage information stored by the Sun Grid Engine system, including the information as provided by the `getrusage` system call (refer to the corresponding entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual*).

This option reports the resource usage entry for the jobs with job-id *[job_id]* or with job name *[job_name]* respectively. If no argument is given, all jobs contained in the referenced accounting file are displayed. If a job-id is selected, and if more than one entry is displayed, either job-id numbers have wrapped around (the range for job-ids is 1 to 999999) or a checkpointing job having migrated is shown.

About Checkpointing Support

Checkpointing is a facility to freeze the status of an executing job or application, save this status (the so called checkpoint) to disk and to restart from that checkpoint later on if the job or application has otherwise failed to complete (e.g. due to a system shutdown). If a checkpoint can be moved from one host to another, checkpointing can be used to migrate applications or jobs in a cluster without considerable loss of computational resources. Hence, dynamic load balancing can be provided by the help of a checkpointing facility.

The Sun Grid Engine system supports two levels of checkpointing.

- *User-level checkpointing*

  At this level, providing the checkpoint generation mechanism is entirely the responsibility of the user or the application. Examples of user-level checkpointing include the following.

  - The periodic writing of restart files encoded in the application at prominent algorithmic steps, combined with proper processing of these files upon restart of the application
  - The use of a checkpoint library which needs to be linked with the application and which thereby installs a checkpointing mechanism.

Note – A variety of third party applications provides an integrated checkpoint facility based on writing of restart files. Checkpoint libraries are available from the public domain (refer to the *Condor* project of the University of Wisconsin for example) or from hardware vendors.

- *Kernel-level transparent checkpointing*

This level of checkpointing must be provided by the operating system (or enhancements to it) which can be applied to potentially arbitrary jobs. No source code changes or re-linking of your application needs 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—while user-level checkpointing is usually restricted to single programs. Thus, the job in which such programs are embedded needs to properly handle the case if the entire job gets restarted.

Kernel-level checkpointing, as well as checkpointing based on checkpointing libraries, can be very resource consuming because the complete virtual address space in use by the job or application at the time of the checkpoint needs to be dumped to disk. As opposed to this, user-level checkpointing based on restart files can restrict the data written to the checkpoint on the important *information* only.

Checkpointing Environments

To reflect the different types of checkpointing methods and the potential variety of derivatives of these methods on different operating system architectures, Sun Grid Engine provides a configurable attribute description for each checkpointing method in use.

This attribute description is called a *checkpointing environment*. Default checkpointing environments are provided with the Sun Grid Engine distribution and can be modified corresponding to the site's needs.

New checkpointing methods can be integrated in principal, but this may become a challenging task and should be performed only by experienced personnel or your Sun Grid Engine support team.

▼ How To Configure Checkpointing Environments with QMON

1. **From the `QMON` Main menu, click the Checkpointing Configuration icon.**

The Checkpointing Configuration dialogue box, which is similar to the example displayed in FIGURE 9-7, is presented.

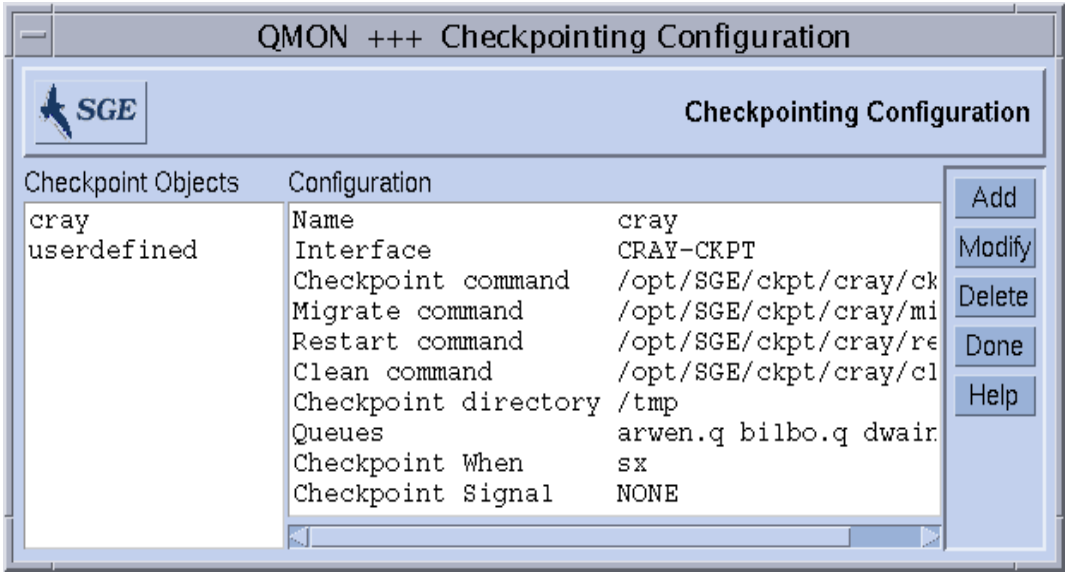


FIGURE 9-7 Checkpointing Configuration Dialogue Box

2. **From the Checkpointing Configuration dialogue box, do one of the following, depending on what you want to accomplish.**

View Configured Checkpointing Environments

- **To view previously configured checkpointing environments, select one of the checkpointing environment names enlisted in the Checkpoint Objects column.**

  The corresponding configuration will be displayed in the Configuration column.

Delete Configured Checkpointing Environments

- **To delete a configured checkpointing environment, highlight its name from the Checkpoint Objects column and press Delete.**

Modify Configured Checkpointing Environments

1. **In the Checkpoint Objects column, highlight the name of the configured checkpointing environment you want to modify and then press Modify.**

   The Change Checkpoint Object dialogue box, similar to the example in FIGURE 9-8, is presented, along with the current configuration of the selected checkpointing environment.

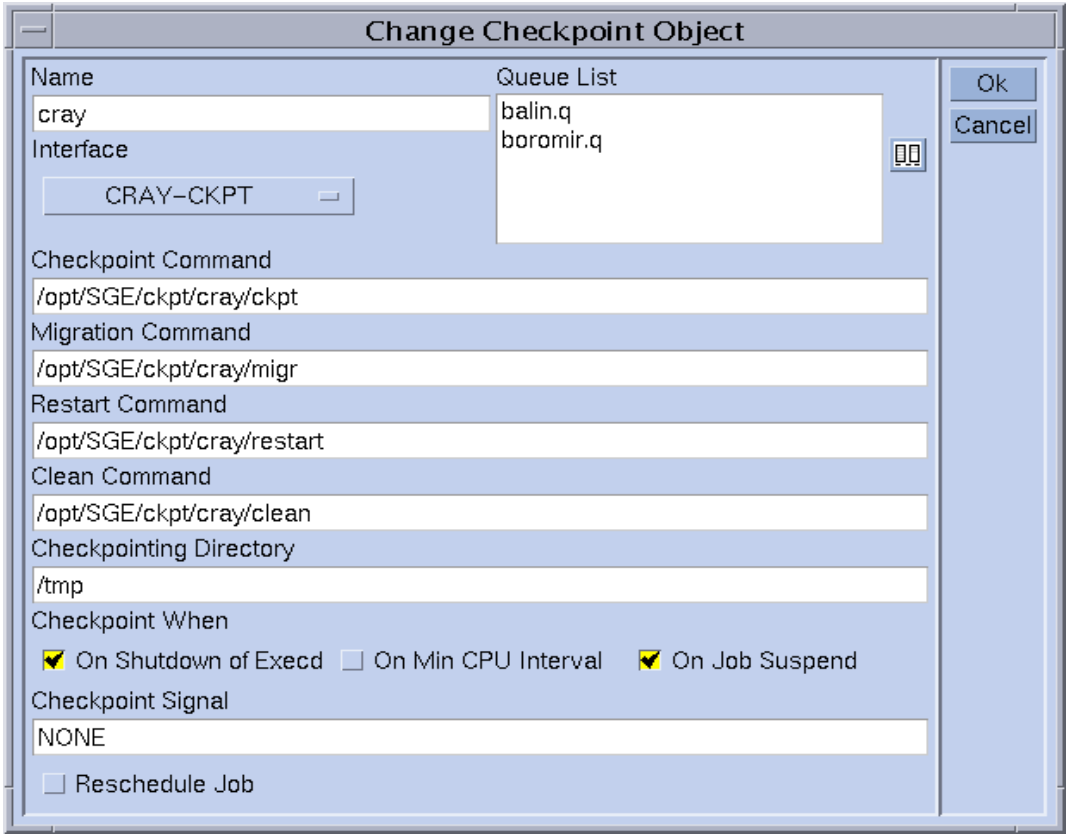


FIGURE 9-8 Change Checkpoint Object Dialogue Box

2. **Modify the selected checkpointing environment according to the following guidelines.**

   The Change Checkpoint Object dialogue box enables you to change the following.

   - Name
   - `Checkpoint`, `migrate`, `restart`, `clean` command strings
   - Directory in which checkpointing files are stored
   - Occasions when checkpoints must be initiated
   - Signal to be sent to job or application when a checkpoint is initiated

Note – Refer to the `checkpoint` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for details on these parameters. In addition, you must define the interface (also called *checkpointing method*) to be used. Select one of those provided in the corresponding selection list and refer to the `checkpoint` entry for details on the meaning of the different interfaces.

3. ***Important* – For the checkpointing environments provided with the Sun Grid Engine distribution, change only the Name, Checkpointing Directory, and Queue List parameters.**

   To change the Queue List parameter, go to "Step a." Otherwise, skip "Step a" and go on to Step 4.

   a. **Click the icon to the right of the Queue List window (see** FIGURE 9-8**).**

      The Select Queues dialogue box, which is similar to the example in FIGURE 9-9, is presented.

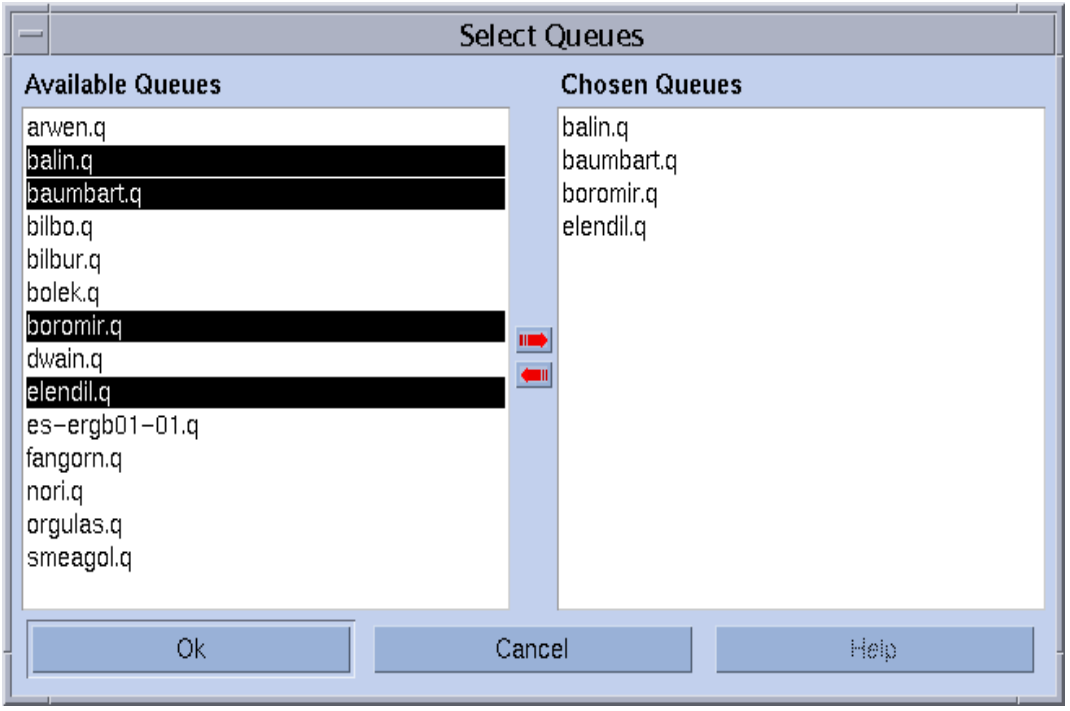


FIGURE 9-9 Checkpointing Queue Selection Dialogue Box

   b. **Select the queues you want to include in the checkpointing environment from the Available Queues list and add them to the Chosen Queues list.**

c. Press OK.

Pressing OK enters these queues into the Queue List window of the Change Checkpoint Object dialogue box.

4. **Press OK to register your changes with** `sge_qmaster`**, or press Cancel to discard your changes.**

Add a Checkpointing Environment

1. **In the Checkpointing Configuration dialogue box, click Add.**

   The Change Checkpoint Object dialogue box, which is similar to the example shown in FIGURE 9-8, is presented, along with a template configuration that you can edit.

2. **Fill out the template with the requested information.**

3. **Press OK to register your changes with** `sge_qmaster`**, or press Cancel to discard your changes.**

▼ How To Configure the Checkpointing Environment from the Command Line

- **Enter the** `qconf` **command and appropriate options, guided by the following sections.**

`qconf` Checkpointing Options

- `qconf -ackpt` *ckpt_name*

  *A*dd checkpointing environment—This command brings up an editor (default `vi` or corresponding to the `$EDITOR` environment variable) with a checkpointing environment configuration template. The parameter *ckpt_name* specifies the name of the checkpointing environment and is already filled into the corresponding field of the template. Configure the checkpointing environment by changing the template and saving to disk. See the `checkpoint` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for a detailed description of the template entries to be changed.

- `qconf -dckpt` *ckpt_name*

  Delete checkpointing environment—This command deletes the specified checkpointing environment.

- `qconf -mckpt` *ckpt_name*

Modify checkpointing environment—This command brings up an editor (default `vi` or corresponding to the `$EDITOR` environment variable) with the specified checkpointing environment as configuration template. Modify the checkpointing environment by changing the template and saving to disk. See the `checkpoint` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for a detailed description of the template entries to be changed.

- `qconf -sckpt` *ckpt_name*

  Show checkpointing environment—This command prints the configuration of the specified checkpointing environment to standard output.

- `qconf -sckptl`

  Show checkpointing environment list—This command displays a list of the names of all checkpointing environments currently configured.

CHAPTER 10

Managing Parallel Environments

This chapter includes information relating to management and administration of parallel environments.

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

About Parallel Environments

A *Parallel Environment* (PE) is a software package designed for concurrent computing in networked environments or parallel platforms. A variety of systems have evolved over the past years into viable technology for distributed and parallel processing on various hardware platforms. Examples for two of the most common message-passing environments today are PVM (Parallel Virtual Machine, Oak Ridge National Laboratories) and 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 be able to handle arbitrary parallel jobs running on top of such systems, the Sun Grid Engine system provides a flexible and powerful interface that satisfies the various needs.

Arbitrary PEs can be interfaced by Sun Grid Engine as long as suitable startup and stop procedures are provided as described in the section, “The PE Startup Procedure” on page 253 and in the section, “Termination of the PE” on page 254, respectively.

▼ How To Configure PEs with QMON

1. **From the QMON Main menu, click the PE Configuration button.**

   The Parallel Environment Configuration dialogue box, which is similar to the example in FIGURE 10-1, is presented.

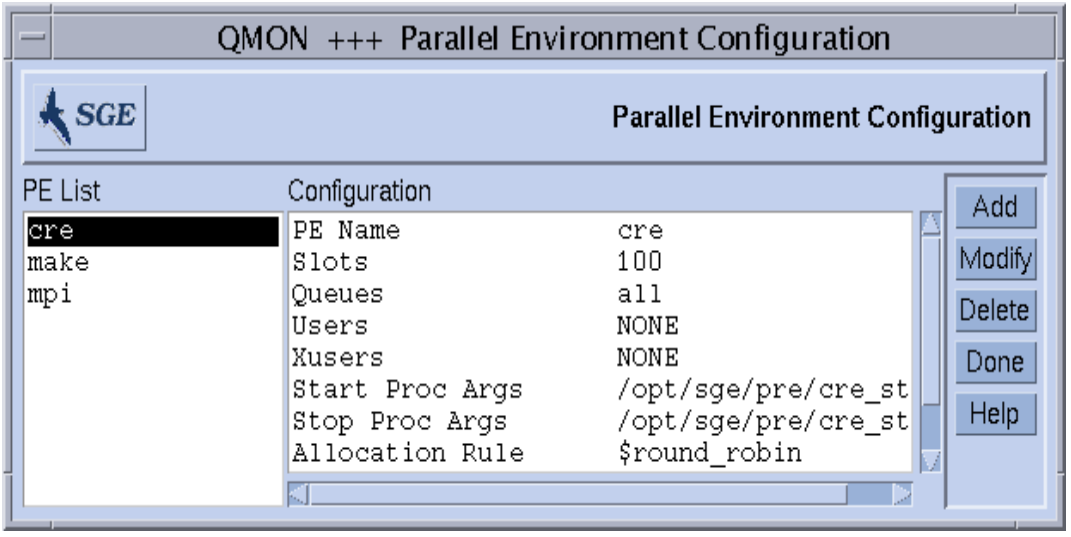


FIGURE 10-1 Parallel Environment Configuration Dialogue Box

PEs that have already been configured are displayed in the PE List selection list on the left side of the screen.

2. **From the Parallel Environment Configuration dialogue box, do one of the following, depending on what you want to accomplish.**

▼ Display the Contents of a PE

- **To display the contents of a PE, click its name in the PE List selection list.**

  The PE configuration’s content is displayed in the Configuration display region.

▼ Delete a PE

- **To delete a selected PE, highlight its name in the PE List selection list and then press Delete (on the right side of the window).**

▼ Modify a PE

1. **To modify a selected PE, press the Modify button.**

   The PE Definition dialogue box, similar to the example shown in FIGURE 10-2, is presented.

2. **Modify the PE definitions according to guidance in the section, "Explanation of the Parallel Environment Definition Parameters" on page 248.**

3. **Press OK to save changes, or Cancel to discard changes.**

   Pressing either OK or Cancel dismisses the dialogue box.

▼ Add a PE

1. **To add new PEs, press the Add button.**

   The PE Definition dialogue box, similar to the example shown in FIGURE 10-2, is presented.

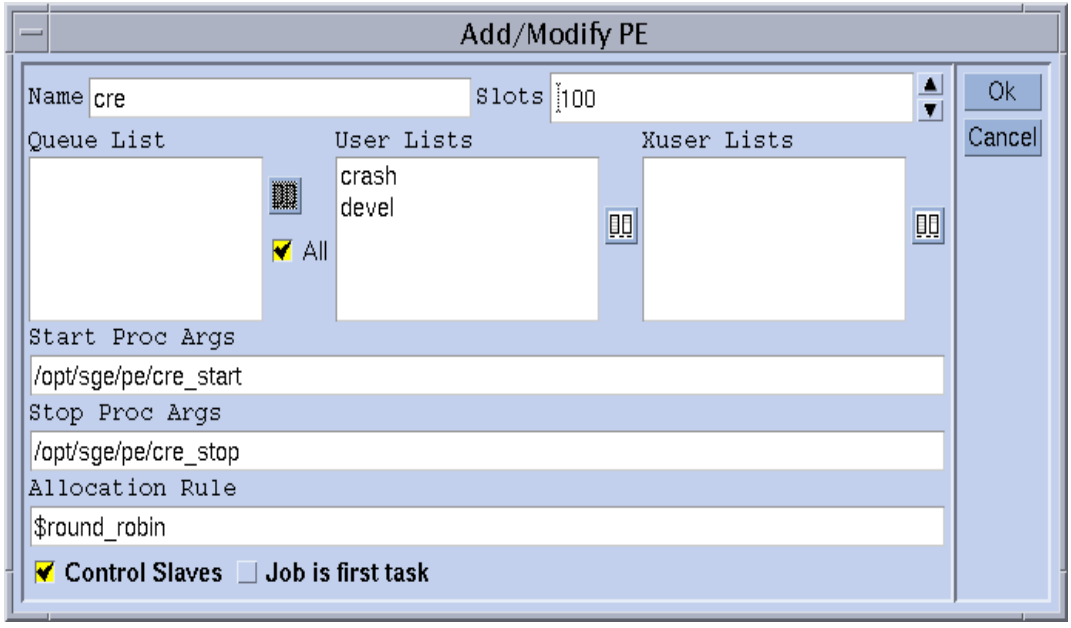


FIGURE 10-2 Parallel Environment Definition Dialogue Box

2. **Add the PE definitions according to guidance in the section, "Explanation of the Parallel Environment Definition Parameters" on page 248.**

3. **Press OK to save changes, or Cancel to discard changes.**

   Pressing either OK or Cancel dismisses the dialogue box.

Explanation of the Parallel Environment Definition Parameters

- The *Name* input window either displays the name of the selected PE in the case of a modify operation or can be used to enter the name of the PE to be declared.
- The *Slots* spin box has to be used to enter the number of job slots in total which may be occupied by all PE jobs running concurrently.
- The *Queue List* display region shows the queues which can be used by the PE. By clicking the icon button on the right side of the Queue List display region, a Select Queues dialogue box, similar to the example in FIGURE 10-3, is presented for you to modify the PE queue list.

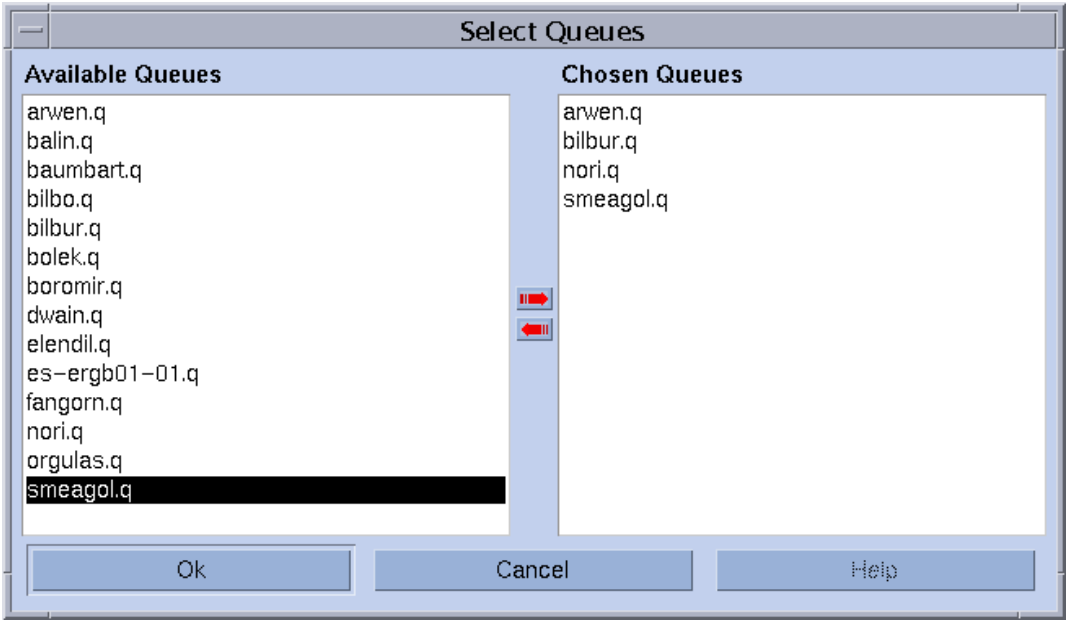


FIGURE 10-3 Select Queues Dialogue Box

- The *User Lists* display region contains the user access lists (see the section, "About User Access Permissions" on page 219) that are allowed to access the PE.
- The *Xuser Lists* display region shows those access lists to which access is denied.

Clicking the icon buttons associated with both display regions presents the Select Access Lists dialogue boxes, similar to the example in FIGURE 10-4. You use these dialogue boxes to modify the content of both access list display regions.

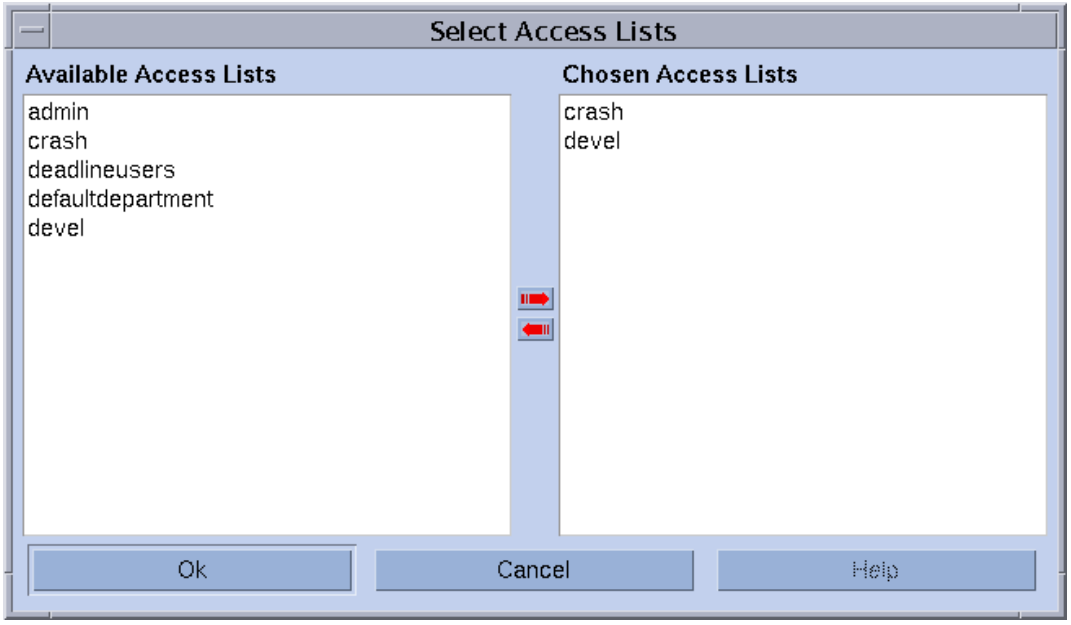


FIGURE 10-4 Select Access Lists Dialogue Box

- The *Start Proc Args* and *Stop Proc Args* input windows are provided to enter the precise invocation sequence of the PE startup and stop procedures (see the sections, “The PE Startup Procedure” on page 253 and “Termination of the PE” on page 254 respectively). The first argument usually is the start or stop procedure itself. The remaining parameters are command-line arguments to the procedures.

  A variety of special identifiers (beginning with a $ prefix) are available to pass Sun Grid Engine internal run-time information to the procedures. The `sge_pe` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* contains a list of all available parameters.

- The *Allocation Rule* input window defines the number of parallel processes to be allocated on each machine which is used by a PE. Currently, only positive integer numbers and the special value `$pe_slots` are supported. `$pe_slots` denotes that all processes that are created have to be located on a single host.

- The *Control Slaves* toggle button declares whether parallel tasks are generated via Sun Grid Engine (i.e., via `sge_execd` and `sge_shepherd`) or whether the corresponding PE performs its own process creation. It is advantageous if the Sun Grid Engine system has full control over slave tasks (correct accounting and

resource control), but this functionality is only available for PE interfaces especially customized for Sun Grid Engine. Refer to the section, “Tight Integration of PEs and Sun Grid Engine Software” on page 255 for further details.

- The *Job is first task* toggle button is meaningful only if Control Slaves has been switched on. It indicates that the job script or one of its child processes acts as one of the parallel tasks of the parallel application (this is usually the case for PVM, for example). If it is switched off, the job script initiates the parallel application but does not participate (e.g., in case of MPI when using `mpirun`).

▼ How To Configure PEs from the Command Line

Enter the `qconf` command with appropriate options, guided by the following sections.

`qconf` PE Options

- `qconf -ap` *pe_name*

  Add parallel environment—This command brings up an editor (default `vi` or corresponding to the `$EDITOR` environment variable) with a PE configuration template. The parameter, *pe_name*, specifies the name of the PE and is already filled into the corresponding field of the template. Configure the PE by changing the template and saving to disk. See the `sge_pe` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for a detailed description of the template entries to be changed.

- `qconf -dp` *pe_name*

  Delete parallel environment—This command deletes the specified PE.

- `qconf -mp` *pe_name*

  Modify parallel environment—This command brings up an editor (default `vi` or corresponding to the `$EDITOR` environment variable) with the specified PE as configuration template. Modify the PE by changing the template and saving to disk. See the `sge_pe` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for a detailed description of the template entries to be changed.

- `qconf -sp` *pe_name*

  Show parallel environment—This command prints the configuration of the specified PE to standard output.

- `qconf -spl`

  Show parallel environment list—This command displays a list of the names of all parallel environments currently configured.

▼ How To Display Configured PE Interfaces from the Command Line

- **Enter the following commands.**

```
% qconf -spl
% qconf -sp pe_name
```

The first command prints a list of the names of the currently available PE interfaces. The second command displays the configuration of a particular PE interface. Refer to the `sge_pe` manual page for details on the PE configuration.

▼ How To Display Configured PE Interfaces with QMON

- **In the `QMON` Main menu, press the PE Config button.**

The Parallel Environment Configuration dialogue box is displayed (see the section, “How To Configure PEs with `QMON`” on page 246).

The example from the section, “Advanced Example” on page 81 already defines a parallel job requesting the PE interface `mpi` (for *message passing interface*) to be used with at least four, but up to (and preferably) 16 processes. The button to the right of the Parallel Environment (PE) Specification window can be used to pop-up a dialogue box to select the desired parallel environment from a list of available PEs (see FIGURE 10-5). The requested range for the number of parallel tasks initiated by the job can be added after the PE name in the PE Specification window of the Advanced Submission screen.

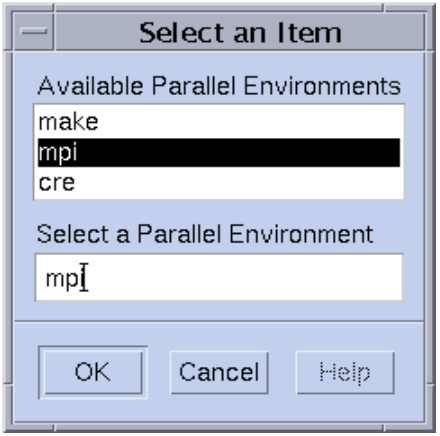


FIGURE 10-5 PE Selection

The command line submit command corresponding to the parallel job specification described above is given in section "How To Submit Jobs from the Command Line" on page 93 and shows how the `qsub -pe` option has to be used to formulate an equivalent request. The `qsub` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* provides more detail on the `-pe` syntax.

It is important to select a suitable PE interface for a parallel job. PE interfaces may utilize no or different message passing systems, they may allocate processes on single or multiple hosts, access to the PE may be denied to certain users, only a specific set of queues may be used by a PE interface and only a certain number of queue slots may be occupied by a PE interface at any point of time. You should therefore ask the Sun Grid Engine administration for the available PE interface(s) best suited for your type(s) of parallel jobs.

You can specify resource requirements as explained in the section, "Resource Requirement Definition" on page 86 together with your PE request. This will further reduce the set of eligible queues for the PE interface to those queues also fitting the resource requirement definition you specified. Assume, for example, that you have submitted the following command:

```
% qsub -pe mpi 1,2,4,8 -l nastran,arch=osf nastran.par
```

The queues suitable for this job are those which are associated to the PE interface `mpi` by the PE configuration and also satisfy the resource requirement specification specified by the `qsub -l` option.

Note – The Sun Grid Engine PE interface facility is highly configurable. In particular, the Sun Grid Engine administration can configure the PE start-up and stop procedures (see the `sge_pe` manual page) to support site specific needs. The `qsub -v` and `-V` options to export environment variables may be used to pass information from the user who submits the job to the PE start-up and stop procedures. Ask the Sun Grid Engine administrator if you are required to export certain environment variables.

The PE Startup Procedure

The Sun Grid Engine system starts the PE by invoking a startup procedure via the `exec` system call. The name of the startup executable and the parameters passed to this executable are configurable from within the Sun Grid Engine system. An example for such a startup procedure for the PVM environment is contained within the Sun Grid Engine distribution tree. It consists 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 operations required are handled by the shell script.

The shell script is located under <*sge_root*>/`pvm/startpvm.sh`. The C-program file can be found under <*sge_root*>/`pvm/src/start_pvm.c`.

Note – The startup procedure could have been covered by a single C-program as well. The shell script is used to allow for easier customizing of the sample startup procedure.

The example script, `startpvm.sh`, requires the following three arguments.

- The path of a host file generated by Sun Grid Engine software, containing the names of the hosts from where PVM is going to be started
- The host on which the `startpvm.sh` procedure was invoked
- The path of the PVM root directory (as usually contained in the `PVM_ROOT` environment variable)

These parameters can be passed to the startup script via the means described in “How To Configure PEs with `QMON`” on page 246. The parameters are among those provided to PE startup and stop scripts by Sun Grid Engine during runtime. The required host file, as an example, is generated by Sun Grid Engine and the name of the file can be passed to the startup procedure in the PE configuration by the special parameter name, `$sge_hostfile`. A description of all available parameters is given in the `sge_pe` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual*.

The host file has the following format.

- Each line of the file refers to a host on which parallel processes are to be run.
- The first entry of each line denotes the host name.
- The second entry specifies the number of parallel processes to be run on the host.
- The third entry denotes a processor range to be used in case of a multiprocessor machine.

This file format is generated by Sun Grid Engine and is fixed. PEs, which need a different file format (as, for example, PVM) need to translate it within the startup procedure (see the `startpvm.sh` file).

As soon as the PE startup procedure has been started by the Sun Grid Engine system, it launches the PE. The startup procedure should exit with a zero exit status. If the exit status of the startup procedure is not zero, Sun Grid Engine software resports an error and will not start the parallel job.

Note – It is beneficial to test any startup procedures first from the command line—without Sun Grid Engine—to remove all errors that may be hard to trace if the procedure is integrated into the Sun Grid Engine framework.

Termination of the PE

When a parallel job finishes or is aborted (via `qdel`), a procedure to halt the parallel environment is called. The definition and semantics of this procedure are very similar to those described for the startup program. The stop procedure can also be defined in a PE configuration (see, for example, “How To Configure PEs with `QMON`” on page 246).

The stop procedure’s purpose is to shut down the PE and to reap all associated processes.

Note – If the stop procedure fails to clean up PE processes, the Sun Grid Engine system may have no information about the processes running under PE control and thus cannot clean up. Sun Grid Engine software, of course, cleans up the processes directly associated with the job script that it has launched.

The Sun Grid Engine distribution tree also contains an example of a stop procedure for the PVM PE. It resides under <*sge_root*>/`pvm/stoppvm.sh`. It takes the following two arguments.

- The path to the host file generated by the Sun 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 exit status zero on success and a non-zero exit status on failure.

Note – It is beneficial to test any stop procedures first from the command line—without Sun Grid Engine—to remove all errors that may be hard to trace if the procedure is integrated into the Sun Grid Engine framework.

Tight Integration of PEs and Sun Grid Engine Software

The explanation of the Control Slaves parameter in the section, “How To Configure PEs with QMON” on page 246 mentions that PEs for which the creation of parallel tasks is performed by the Sun Grid Engine components `sge_execd` and `sge_shepherd` offer benefits over PEs that perform their own process creation. This is due to the fact that 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 PEs do not implement these features and hence do not provide a sufficient interface for the integration with a resource management system like Sun Grid Engine. To overcome this problem, the Sun Grid Engine system provides an advanced PE interface for the tight integration with PEs, which transfers the responsibility for the task creation from the PE to Sun Grid Engine software.

The Sun Grid Engine distribution contains two examples of such a tight integration for the PVM public domain version and 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 contain a *loosely* integrated variant of the interfaces for comparison in addition, as well as `README` files describing the usage and any current restrictions. Refer to those `README` files for further detail.

Note – Performing a tight integration with a PE is an advanced task and may require expert knowledge of the PE and the Sun Grid Engine PE interface. You may want to contact your Sun support representative distributor for assistance.

CHAPTER 11

Error Messaging

This chapter describes Sun Grid Engine 5.3 error messaging procedures. Use this information to help you troubleshoot any problems you may encounter with the software.

How Sun Grid Engine 5.3 Software Retrieves Error Reports

Sun Grid Engine software reports errors or warnings by logging messages into certain files and/or by electronic mail (e-mail). The logfiles used are:

- Messages Files:

  There are separate messages files for the `sge_qmaster`, the `sge_schedd` and the `sge_execd`s. The files have the same file name `messages`. The `sge_qmaster` logfile resides in the master spool directory, the `sge_schedd` messages file in the scheduler spool directory and the execution daemons' logfiles reside in the spool directories of the execution daemons (see the section, "Spool Directories Under the Root Directory" on page 25 for more information about the spool directories).

  The messages files have the following format:

  - Each message occupies a single line.
  - The messages are subdivided into 5 components separated by the vertical bar sign (|).
  - The first component is a time stamp for the message.
  - The second specifies the Sun Grid Engine daemon generating the message.
  - The third is the hostname the daemon runs on.

- The fourth is a message type which is either N for notice, I for info (both for informational purposes only), W for warning (s.th. may be wrong), E for error (an error condition has been detected) or C for critical (may lead to a program abort).
- The fifth is the message text.

Note – If, for some reason, an error logfile is not accessible, Sun Grid Engine will try 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.

- 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 file name and the location either complies to a default or may be specified by certain `qsub` command line switches. Please refer to the *Sun Grid Engine 5.3 Administration and User's Guide* and the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for detailed information.

In some circumstances Sun Grid Engine notifies users and/or administrators about error events via e-mail. The mail messages sent by Sun Grid Engine do not contain a message body. The message text is fully contained in the mail subject field.

Consequences of Different Error or Exit Codes

TABLE 11-1 lists the consequences of different job-related error or exit codes. These codes are valid for every type of Sun Grid Engine job.

TABLE 11-1 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

TABLE 11-2 lists the consequences of error or exit codes of jobs related to parallel environment (PE) configuration.

TABLE 11-2 PE-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
	Rest	Queue set to error state, job not requeued

TABLE 11-3 lists the consequences of error or exit codes of jobs related to Queue configuration. These are valid only if corresponding methods have been overwritten.

TABLE 11-3 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

TABLE 11-4 lists the consequences of error or exit codes of jobs related to checkpointing.

TABLE 11-4 Checkpointing-Related Error or Exit Codes

Script/Method	Exit or Error Code	Consequence
Checkpoint	0	Success
	Rest	Success—For kernel checkpoint, however, special meaning: Checkpoint not sucessful; it did not happen.
Migrate	0	Success
	Rest	Success—For kernel checkpoint, however, special meaning: Checkpoint not sucessful; it did not happen. Migration will occur.
Restart	0	Success
	Rest	Success, no other special meaning
Clean	0	Success
	Rest	Success, no other special meaning

Running Sun Grid Engine Programs in Debug Mode

For some severe error conditions the error logging mechanism may not yield sufficient information to identify the problems. Therefore, Sun Grid Engine offers the ability to run almost all ancillary programs and the daemons in *debug* mode. There are different debug levels varying in the extent and depth of information which is provided. The debug levels range from 0 to 10, with 10 being the level delivering the most detailed information and 0 switching off debugging.

To set a debug level an extension to your .cshrc or .profile resource files is provided with the Sun Grid Engine distribution. 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 need to be "sourced" into your standard resource file. As csh or tcsh user please include the line:

```
source <sge_root>/util/dl.csh
```

into your .cshrc file. As sh or ksh user, adding the line:

```
. <sge_root>/util/dl.sh
```

into your .profile file is the equivalent. As soon as you now logout and login again you can use the following command to set a debug level *level*:

```
% dl level
```

If *level* is greater than 0, starting a Sun Grid Engine command hereafter will force the command to write trace output to STDOUT. The trace output may contain warnings, status and error messages as well as the names of the program modules being called internally together with source code line number information (which is helpful for error reporting) depending on the debug level being enforced.

Note – It may be useful to watch a debug trace in a window with a considerable scroll line buffer (e.g. 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.

Running one of the Sun Grid Engine daemons in debug mode will have the result, that the daemons keep their terminal connection to write the trace output. They can be aborted by typing the interrupt character of the terminal emulation you use (e.g. Control-C).

Note – To switch off the debug mode, set the debug level back to 0.
