



Sun™ Grid Engine, Enterprise Edition 5.3 Administration and User's Guide

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

Part No. 816-4739-10
April 2002, Revision 01

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

Copyright 2002 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, CA 95054 U.S.A. All rights reserved.

This product or document is distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, AnswerBook2, docs.sun.com, and Solaris are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. The Energy Star logo is a registered trademark of EPA. The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

Federal Acquisitions: Commercial Software—Government Users Subject to Standard License Terms and Conditions.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2002 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, CA 95054 Etats-Unis. Tous droits réservés.

Ce produit ou document est distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, AnswerBook2, docs.sun.com, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

Achats fédéraux : logiciel commercial - Les utilisateurs gouvernementaux doivent respecter les conditions du contrat de licence standard.

LA DOCUMENTATION EST FOURNIE "EN L'ETAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFAÇON.



Contents

Preface xvii

How This Book Is Organized xvii

Using UNIX Commands xviii

Typographic Conventions xix

Shell Prompts xix

Related Documentation xix

Accessing Sun Documentation Online xx

Sun Welcomes Your Comments xx

Part I. Background and Definitions

1. Introduction to Sun Grid Engine, Enterprise Edition 5.3 23

What Is Grid Computing? 23

Managing Workload by Managing Resources and Policies 25

How the System Operates 26

 Matching Resources to Requests 26

 Jobs and Queues: The Sun Grid Engine World 27

 Variety of Utilization Policies 28

 Policy Administration with the Ticket Paradigm 29

Sun Grid Engine, Enterprise Edition Components 30

Hosts	30
Master Host	30
Execution Host	30
Administration Host	31
Submit Host	31
Daemons	31
sge_qmaster – the Master Daemon	31
sge_schedd – the Scheduler Daemon	31
sge_execd – the Execution Daemon	32
sge_commd – the Communication Daemon	32
Queues	32
Client Commands	32
QMON, the Sun Grid Engine, Enterprise Edition Graphical User Interface	35
Customizing QMON	35
Glossary of Sun Grid Engine Terms	36

Part II. Getting Started

2. Installation 43

Basic Installation Overview	43
Phase 1 - Planning	44
Phase 2 - Installing the Software	44
Phase 3 - Verifying the Installation	45
Planning the Installation	45
Prerequisite Tasks	45
The Installation Directory <SGE_ROOT>	45
Spool Directories Under the Root Directory	46
Directory Organization	46
Disk Space Requirements	47

Installation Accounts	48
File Access Permissions	48
Network Services	48
Master Host	49
Shadow Master Hosts	49
Execution Hosts	50
Administrative Hosts	50
Submit Hosts	50
Cells	50
User Names	50
Queues	51
▼ How To Plan the Installation	52
▼ How To Read the Distribution Media	52
pkgadd Method	53
tar Method	54
Performing the Basic Installation	54
▼ How To Install the Master Host	55
▼ How To Install Execution Hosts	56
▼ How To Install Administration and Submit Hosts	57
Installing with Increased Security	57
Additional Setup Required	58
▼ How To Install and Set Up a CSP-Secured System	58
▼ How To Generate Certificates and Private Keys for Users	67
▼ How To Check Certificates	69
Display a Certificate	69
Check Issuer	69
Check Subject	69
Show Email of Certificate	70

	Show Validity	70
	Show Fingerprint	70
	Verifying the Installation	70
▼	How To Verify the Installation	71
Part III.	Using Sun Grid Engine	
	Enterprise Edition 5.3 Software	
3.	Navigating Through	
	Sun Grid Engine, Enterprise Edition	77
	Sun Grid Engine, Enterprise Edition User Types and Operations	77
	Queues and Queue Properties	78
	The QMON Browser	79
▼	How To Launch the QMON Browser	79
	The Queue Control QMON Dialogue Box	79
▼	How To Display a List of Queues	80
▼	How To Display Queue Properties	80
	Using the QMON Browser	80
	From the Command Line	82
	Interpreting Queue Property Information	82
	Host Functionality	83
▼	How To Find the Name of the Master Host	83
▼	How To Display a List of Execution Hosts	83
▼	How To Display a List of Administration Hosts	84
▼	How To Display a List of Submit Hosts	84
	Requestable Attributes	84
▼	How To Display a List of Requestable Attributes	85
	User Access Permissions	88
	Managers, Operators and Owners	90

4. Submitting Jobs	91
Running a Simple Job	91
▼ How To Run a Simple Job from the Command Line	92
▼ How To Submit Jobs From the Graphical User Interface, QMON	93
Submitting Batch Jobs	97
About Shell Scripts	98
Example of a Script File	99
Submitting Extended and Advanced Jobs with QMON	99
Extended Example	100
Advanced Example	105
Resource Requirement Definition	110
How Sun Grid Engine, Enterprise Edition Allocates Resources	112
Extensions to Regular Shell Scripts	113
How a Command Interpreter Is Selected	113
Output Redirection	114
Active Sun Grid Engine, Enterprise Edition Comments	114
Environment Variables	115
▼ How To Submit Jobs from the Command Line	117
Default Requests	118
Array Jobs	119
▼ How To Submit an Array Job from the Command Line	120
▼ How To Submit an Array Job with QMON	120
Submitting Interactive Jobs	121
Submitting Interactive Jobs with QMON	122
▼ How To Submit Interactive Jobs with QMON	122
Submitting Interactive Jobs with qsh	124
▼ How To Submit Interactive Jobs With qsh	125

	Submitting Interactive Jobs with <code>qlogin</code>	125
▼	How To Submit Interactive Jobs With <code>qlogin</code>	125
	Transparent Remote Execution	125
	Remote Execution with <code>qssh</code>	126
▼	How To Invoke Transparent Remote Execution with <code>qssh</code>	127
	Transparent Job Distribution with <code>qtcsh</code>	127
	<code>qtcsh</code> Usage	128
	Parallel Makefile Processing with <code>qmake</code>	130
	<code>qmake</code> Usage	131
	How Sun Grid Engine, Enterprise Edition Jobs Are Scheduled	132
	Job Priorities	133
	Tickets	133
	Queue Selection	134
5.	Checkpointing, Monitoring, and Controlling Jobs	137
	About Checkpointing Jobs	137
	User-Level Checkpointing	138
	Kernel-Level Checkpointing	138
	Migration of Checkpointing Jobs	138
	Composing a Checkpointing Job Script	139
▼	How To Submit, Monitor, or Delete a Checkpointing Job from the Command Line	140
▼	How To Submit a Checkpointing Job with <code>QMON</code>	141
	File System Requirements	142
	Monitoring and Controlling Sun Grid Engine, Enterprise Edition Jobs	143
▼	How To Monitor and Control Jobs with <code>QMON</code>	143
	Additional Information with the <code>QMON</code> Object Browser	152
▼	How To Monitor Jobs with <code>qstat</code>	153

- ▼ How To Monitor Jobs by Electronic Mail 156
 - Controlling Sun Grid Engine, Enterprise Edition Jobs from the Command Line 156
- ▼ How To Control Jobs from the Command Line 157
 - Job Dependencies 158
 - Controlling Queues 158
- ▼ How To Control Queues with QMON 158
- ▼ How To Control Queues with qmod 162
- Customizing QMON 163

Part IV. Administration

6. Host and Cluster Configuration 167

- About Master and Shadow Master Configuration 168
- About Daemons and Hosts 169
 - About Configuring Hosts 170
 - Invalid Host Names 170
- ▼ How To Configure Administration Hosts with QMON 171
- ▼ How To Delete an Administration Host 172
- ▼ How To Add an Administration Host 172
- ▼ How To Configure Administration Hosts From the Command Line 172
- ▼ How To Configure Submit Hosts with QMON 173
- ▼ How To Delete a Submit Host 174
- ▼ How To Add a Submit Host 174
- ▼ How To Configure Submit Hosts from the Command Line 174
- ▼ How To Configure Execution Hosts with QMON 175
- ▼ How To Delete an Execution Host 176
- ▼ How To Shut Down the Execution Host Daemon 176
- ▼ How To Add or Modify an Execution Host 177

▼	How To Configure Execution Hosts from the Command Line	181
▼	How To Monitor Execution Hosts With qghost	182
▼	How To Kill Daemons from the Command Line	183
▼	How To Restart Daemons from the Command Line	184
	The Basic Cluster Configuration	184
▼	How To Display the Basic Cluster Configurations from the Command Line	185
▼	How To Modify the Basic Cluster Configurations from the Command Line	185
▼	How To Display a Cluster Configuration with QMON	186
▼	How To Delete a Cluster Configuration with QMON	187
▼	How To Display a Global Cluster Configuration with QMON	187
▼	How To Use QMON To Modify Global and Host Configurations	188
7.	Configuring Queues and Queue Calendars	191
	About Configuring Queues	191
▼	How To Configure Queues with QMON	192
▼	How To Configure General Parameters	193
▼	How To Configure Execution Method Parameters	195
▼	How To Configure Checkpointing Parameters	196
▼	How To Configure Load and Suspend Thresholds	197
▼	How To Configure Limits	198
▼	How To Configure User Complexes	200
▼	How To Configure Subordinate Queues	202
▼	How To Configure User Access	203
▼	How To Configure Project Access	204
▼	How To Configure Owners	205
▼	How To Configure Queues from the Command Line	206
	About Queue Calendars	207

▼	How To Configure Queue Calendars With QMON	207
▼	How To Configure Calendars From the Command Line	210
8.	The Complexes Concept	213
	About Complexes	213
▼	How To Add Or Modify a Complex Configuration	214
	Complex Types	216
	The Queue Complex	216
	The Host Complex	217
	The Global Complex	219
	User-Defined Complexes	220
	Consumable Resources	224
▼	How To Set Up Consumable Resources	224
	Examples of Setting Up Consumable Resources	226
	Configuring Complexes	235
▼	How To Modify Complex Configurations From the Command Line	235
	Example of the qconf Command	236
	Load Parameters	237
	The Default Load Parameters	237
	Adding Site-Specific Load Parameters	237
▼	How to Write Your Own Load Sensors	238
	Rules	238
	Example of a Script	239
9.	Managing User Access and Policies	243
	About Setting Up a User	244
	About User Access	245
▼	How To Configure Accounts with QMON	246
▼	How To Configure Manager Accounts with QMON	246

▼	How To Configure Manager Accounts from the Command Line	247
	Available Switches	247
▼	How To Configure Operator Accounts with QMON	248
▼	How To Configure Operator Accounts from the Command Line	249
	Available Switches	249
	About Queue Owner Accounts	250
	About User Access Permissions	250
▼	How To Configure User Access Lists with QMON	251
▼	How To Configure User Access Lists from the Command Line	253
	Available Options	253
	About Using Usersets To Define Projects and Departments	254
	About User Object Configuration	254
▼	How To Configure the User Object with QMON	254
▼	How To Assign a Default Project	256
▼	How To Configure the User Object from the Command Line	257
	Available Options	257
	About Projects	258
▼	How To Define Projects with QMON	258
▼	How To Define Projects from the Command Line	262
	Available Options	262
	About Scheduling	263
	Scheduling Strategies	263
	Dynamic Resource Management	264
	Queue Sorting	265
	Job Sorting	266
	What Happens in a Scheduler Interval	266
	Scheduler Monitoring	267
	Scheduler Configuration	267

	Default Scheduling	267
	Scheduling Alternatives	268
▼	How To Change the Scheduler Configuration with QMON	271
▼	How To Administer Policy/Ticket Based Advanced Resource Management with QMON	273
	Edit Tickets Region	274
	Policy Button Region	274
	About the Share-Based Policy	275
▼	How To Edit the Share Tree Policy From QMON	278
	Node Attributes Display	279
	Share Tree Policy Parameters	282
	About the Special User, default	283
▼	How To Configure the Share-Based Policy from the Command Line	284
	About the Functional Policy	285
	Functional Shares	285
	The share_functional_shares Parameter	285
▼	How To Configure the Functional Share Policy From QMON	287
▼	How To Configure the Functional Share Policy from the Command Line	290
	About the Deadline Policy	291
	Deadline Tickets	291
	The share_deadline_tickets Parameter	291
	About the Override Policy	294
	The share_override_tickets Parameter	294
▼	How To Configure the Override Policy	296
▼	How To Configure the Override Policy from the Command Line	298
	About Policy Hierarchy	298
	About Path Aliasing	300
	File Format	301

How Path-Aliasing Files Are Interpreted	301
Example of a Path-Aliasing File	302
About Configuring Default Requests	302
Format of Default Request Files	303
Example of a Default Request File	304
About Gathering Accounting and Utilization Statistics	304
About Checkpointing Support	305
Checkpointing Environments	306
▼ How To Configure Checkpointing Environments with QMON	307
View Configured Checkpointing Environments	307
Delete Configured Checkpointing Environments	307
Modify Configured Checkpointing Environments	308
Add a Checkpointing Environment	310
▼ How To Configure the Checkpointing Environment from the Command Line	310
qconf Checkpointing Options	310
10. Managing Parallel Environments	313
About Parallel Environments	313
▼ How To Configure PEs with QMON	314
▼ Display the Contents of a PE	315
▼ Delete a PE	315
▼ Modify a PE	315
▼ Add a PE	316
▼ How To Configure PEs from the Command Line	319
qconf PE Options	319
▼ How To Display Configured PE Interfaces from the Command Line	320
▼ How To Display Configured PE Interfaces with QMON	320
The PE Startup Procedure	322

Termination of the PE 324

Tight Integration of PEs and Sun Grid Engine, Enterprise Edition
Software 324

11. Error Messaging 327

How Sun Grid Engine, Enterprise Edition Software Retrieves Error Reports 327

Consequences of Different Error or Exit Codes 328

Running Sun Grid Engine, Enterprise Edition Programs in Debug
Mode 330

Preface

The *Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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 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 full fresh and upgrade installations.

- Part 3 – Using Sun Grid Engine, Enterprise Edition 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 <code>.login</code> file. Use <code>ls -a</code> to list all files. % You have mail.
AaBbCc123	What you type, when contrasted with on-screen computer output	% su Password:
<i>AaBbCc123</i>	Book titles, new words or terms, words to be emphasized. Replace command-line variables with real names or values.	Read Chapter 6 in the <i>User's Guide</i> . These are called <i>class</i> options. You <i>must</i> be superuser to do this. To delete a file, type <code>rm filename</code> .

Shell Prompts

Shell	Prompt
C shell	<i>machine-name%</i>
C shell superuser	<i>machine-name#</i>
Bourne shell and Korn shell	\$
Bourne shell and Korn shell superuser	#

Related Documentation

Application	Title	Part Number
Reference	<i>Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual</i>	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 (816-4739-10) of your document in the subject line of your email.

PART I Background and Definitions

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

- **Chapter 1** – “Introduction to Sun Grid Engine, Enterprise Edition 5.3” on page 23.

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, Enterprise Edition 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, Enterprise Edition environment

Introduction to Sun Grid Engine, Enterprise Edition 5.3

This chapter provides background information about the Sun Grid Engine, Enterprise Edition 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 explanation of grid computing
- An overview of QMON, the Sun Grid Engine, Enterprise Edition 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, Enterprise Edition terminology

What Is Grid Computing?

Conceptually, a grid is quite simple. It is a collection of computing resources that perform tasks. In its simplest form, a grid appears to users as a large system that provides a single point of access to powerful distributed resources. In their more complex forms (explained later in this section), grids can provide many access points to users. In any case, users treat the grid as a *single* computational resource. Resource management software, such as Sun Grid Engine, Enterprise Edition, accepts jobs submitted by users and schedules them for execution on appropriate systems in the grid based upon resource management policies. Users can submit literally millions 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* are the simplest, consisting of computer *hosts* working together to provide a single point of access to users in a single project or department.
- *Campus grids* enable multiple projects or departments within an organization to share computing resources. Organizations can use campus grids to handle a wide variety of tasks, from cyclical business processes to rendering, data mining, and more.
- *Global grids* are a collection of campus grids that cross organizational boundaries to create very large virtual systems. Users have access to compute power that far exceeds the resources available within their own organization.

FIGURE 1-1 is a graphical representation of the three classes of clusters. In the Cluster grid, a user's job would be handled by one of the systems within the cluster. However, if the user's Cluster grid were part of the more complex Campus grid—and if the Campus grid were part of the largest Global grid—the user's job could be handled by any member *execution* host located anywhere in the world.

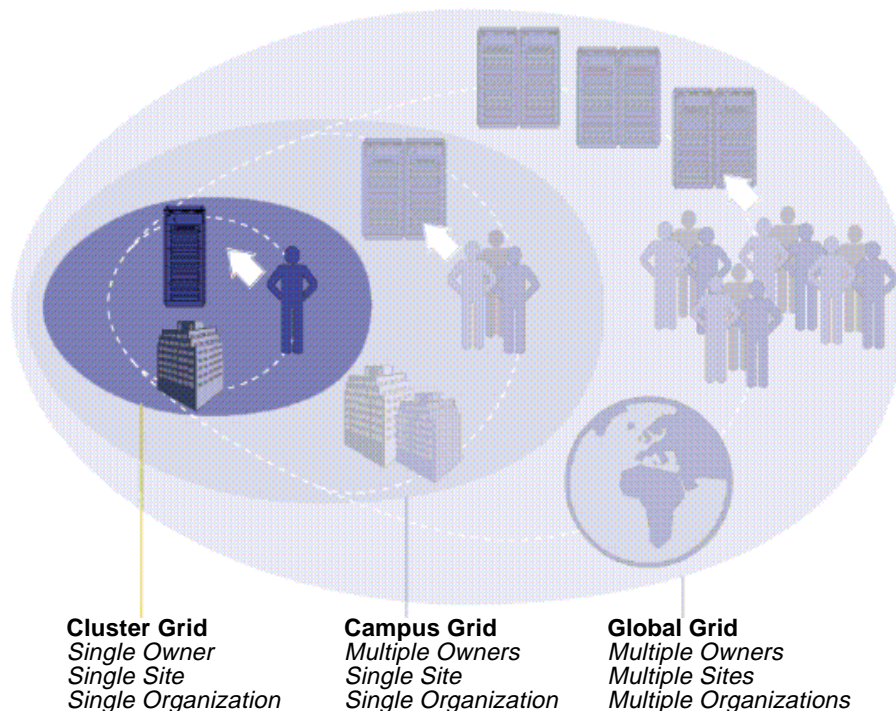


FIGURE 1-1 Three Classes of Grids

Sun Grid Engine, Enterprise Edition 5.3 software, the newest version of Sun's resource management software solution, provides the power and flexibility required for Campus grids. The product is very useful for existing cluster grids enabled by

its relative, Sun Grid Engine, as it facilitates a smooth transition to create a Campus grid by consolidating all existing Sun Grid Engine cluster grids on the campus. In addition, Sun Grid Engine, Enterprise Edition is a good start for an enterprise campus that makes the move to the grid computing model for the first time.

Sun Grid Engine, Enterprise Edition 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 Campus grid, gathers these resources, and then allocates and delivers them automatically in a way that optimizes usage across the Campus grid.

To enable cooperation within the Campus grid, project owners using the grid need to negotiate policies, have flexibility in the policies for manual overrides for unique project requirements, and have the policies automatically monitored and enforced.

Sun Grid Engine, Enterprise Edition 5.3 software can mediate among the entitlements of a multitude of departments and projects that are competing for computational resources..

Managing Workload by Managing Resources and Policies

The Sun Grid Engine, Enterprise Edition system is an advanced resource management tool for heterogeneous, distributed computing environments. Workload management—controlling the use of shared resources to best achieve an enterprise's goals (such as productivity, timeliness, level-of-service, and so forth)—is accomplished through *resource management* and *policy administration*. Sites configure the system to maximize utilization and throughput while supporting varying levels of timeliness (job deadlines) and importance (job priority and user share).

Sun Grid Engine, Enterprise Edition software provides advanced resource management and policy administration for UNIX environments that are composed of multiple shared resources. The Sun Grid Engine, Enterprise Edition system is superior over standard *load management* tools with respect to the following major capabilities.

- Innovative dynamic scheduling and resource management that allows Sun Grid Engine, Enterprise Edition software to enforce site-specific management policies.
- Dynamic performance-data collection to provide the scheduler with up-to-the-moment job level resource consumption and system load information.

- Availability of enhanced security by way of *Certificate Security Protocol (CSP)*-based encryption. Instead of transferring messages in clear text, the messages within this more secure system are encrypted with a secret key.
- High-level policy administration for the definition and implementation of enterprise goals such as productivity, timeliness, and level-of-service.

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

How the System Operates

The Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition bank," one bank employee may be able to provide assistance to several customers at the same time. The Sun Grid Engine, Enterprise Edition system would try to assign new customers to the least loaded and most suitable bank employee.
- As bank manager, the Sun Grid Engine, Enterprise Edition system would allow the bank to define service policies. Typical service policies would be "provide preferential service to commercial customers because they generate more profit," "make sure a certain customer group is served well, because they have received bad service so far," "ensure that customers with an appointment get timely response," or "prefer a certain customer on direct demand of a bank executive."
- Such policies would be implemented, monitored, and adjusted automatically by the Sun Grid Engine, Enterprise Edition manager. Customers with preferential access would be served sooner, they would receive more attention from employees whose assistance they have to share with other customers, and the Sun Grid Engine, Enterprise Edition manager will recognize if the customers do not make the expected progress and will immediately respond by adjusting service levels in order to comply with the bank's service policies.

Jobs and Queues: The Sun Grid Engine World

In a Sun Grid Engine, Enterprise Edition 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, Enterprise Edition software arbitrates available resources and job requirements in the following fashion.

- A user who submits a job through the Sun Grid Engine, Enterprise Edition 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 is scheduled to be available for execution of a new job, the Sun Grid Engine, Enterprise Edition system determines suitable jobs for the queue and immediately dispatches the job with the highest priority or longest waiting time.
- Sun Grid Engine, Enterprise Edition queues may allow concurrent execution of many jobs. The Sun Grid Engine, Enterprise Edition system will try to start new jobs in the least loaded and suitable queue.

Variety of Utilization Policies

The administrator of a Sun Grid Engine, Enterprise Edition cluster can define high-level utilization policies, customized according to whatever is appropriate for the site. Four such policies are available.

- **Functional** – Using this policy, an administrator can provide special treatment because of a user's or job's affiliation with a certain user group, project, or so forth.
- **Share-based** – Under this policy, the level of service depends on an assigned share entitlement, the corresponding shares of other users and user groups, the past usage of resources by all users, and the current presence of users within the system.
- **Deadline** – This policy is invoked whenever a job must be finished before or at a certain point in time and therefore may require special treatment to achieve this.
- **Override** – This policy requires manual intervention by the Sun Grid Engine, Enterprise Edition cluster administrator, who modifies the automated policy implementation.

Sun Grid Engine, Enterprise Edition's policy management will automatically control the use of shared resources in the cluster to best achieve the goals of the administration. High-priority jobs are dispatched preferentially and receive higher CPU entitlements if they compete for resources with other jobs. Sun Grid Engine, Enterprise Edition software monitors the progress of all jobs and adjusts their relative priorities correspondingly and with respect to the goals defined in the policies.

Policy Administration with the Ticket Paradigm

The policies are all defined via a unique Sun Grid Engine, Enterprise Edition concept called *tickets*. Tickets can be compared to shares of a public company's stock. The more stock shares you own, the more important you are to the company. If shareholder A owns twice as many shares as shareholder B, then A also has twice the votes of B and, hence, twice the importance to the company. The more tickets a Sun Grid Engine, Enterprise Edition job has, the more important it is. If a job A has twice the tickets of job B, then job A is entitled to twice the resource usage of job B.

Sun Grid Engine, Enterprise Edition jobs can retrieve tickets from all four policies, and the total number of tickets—as well as the number retrieved from each policy—often changes over time.

The Sun Grid Engine, Enterprise Edition cluster administrator controls the number of tickets that are allocated to each policy in total. Just as it does for jobs, this allocation determines the relative importance of the *policies* among each other. Through the ticket pool assigned to particular policies, the administration can run a Sun Grid Engine, Enterprise Edition system in a share-based mode only, or it can run a mix; for example, 90% share-based and 10% functional. FIGURE 1-2 is a representation of this correlation between policies and tickets.

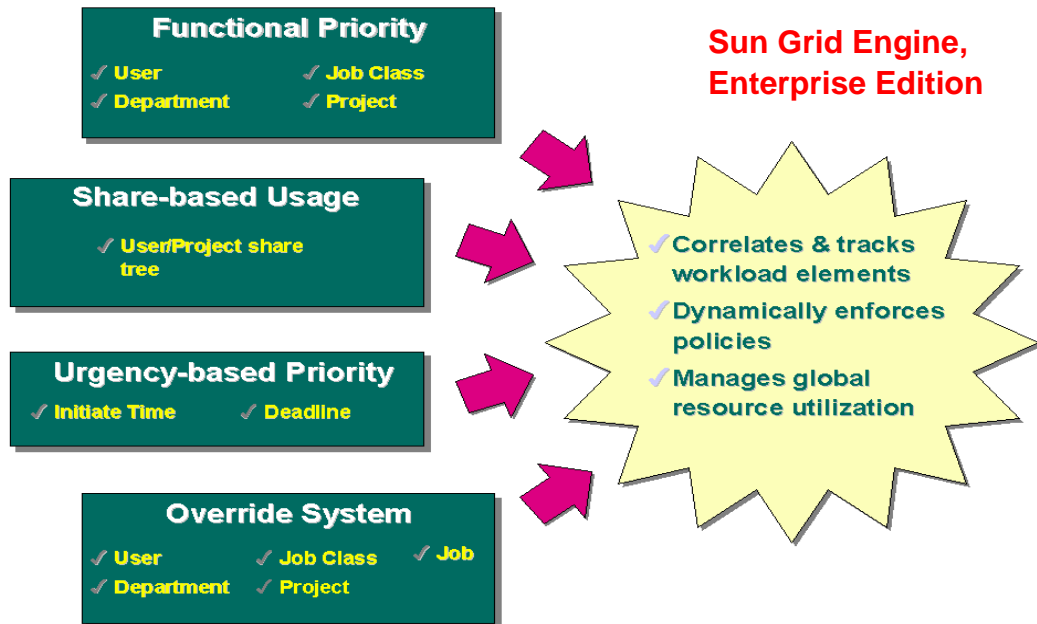


FIGURE 1-2 Correlation Between Policies and Tickets in a Sun Grid Engine, Enterprise Edition System

Sun Grid Engine, Enterprise Edition Components

FIGURE 1-3 displays the most important Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition components, such as queues and jobs, and maintain tables about the status 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, Enterprise Edition jobs. Therefore, they are hosting Sun Grid Engine, Enterprise Edition queues and run the Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition OSF/1 Motif graphical user's interface, `QMON`, which is described in the section, “`QMON`, the Sun Grid Engine, Enterprise Edition Graphical User Interface” on page 35.

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, Enterprise Edition 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 decisions:

- Which jobs are dispatched to which queues
- How jobs are to be reordered and reprioritized to maintain share, priority, or deadline

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, Enterprise Edition components.

Queues

A Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition daemons will be unable to select a lower-loaded or better-suited device.

Client Commands

Sun Grid Engine, Enterprise Edition'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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition 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 batch jobs to the Sun Grid Engine, Enterprise Edition system.

- `qtcs`h – This command is a fully compatible replacement for the widely known and used Unix C-Shell (`cs`h) derivative, `tc`sh. 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, Enterprise Edition 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, Enterprise Edition system, depicted in FIGURE 1-3.

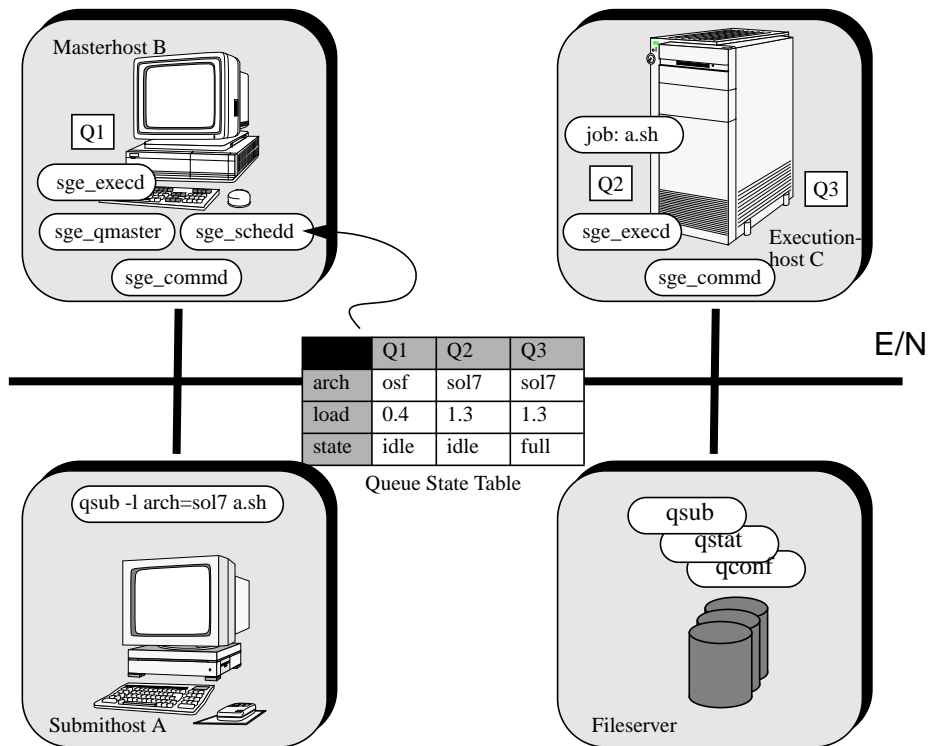


FIGURE 1-3 Component Interaction in the Sun Grid Engine, Enterprise Edition System

QMON, the Sun Grid Engine, Enterprise Edition Graphical User Interface

Using QMON, the graphical user interface (GUI) tool, you can accomplish most—if not all—Sun Grid Engine, Enterprise Edition tasks. FIGURE 1-4 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.

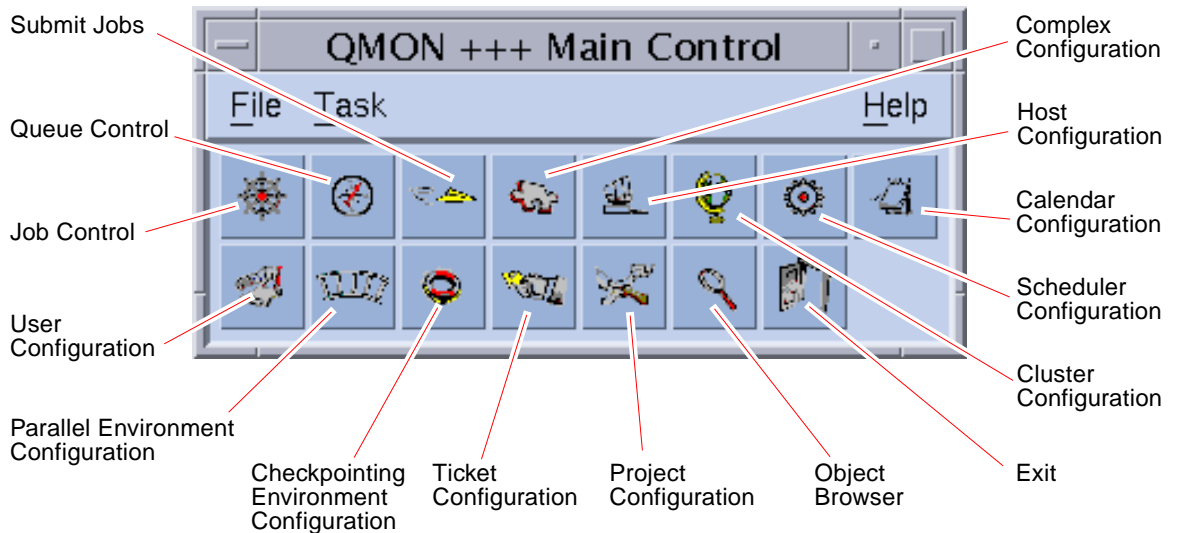


FIGURE 1-4 QMON Main Menu, Defined

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 `<sgc_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 be 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, Enterprise Edition 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, Enterprise Edition 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.
- cell** A separate Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition functions occur.
complex	A set of attributes that can be associated with a queue, a host, or the entire cluster.
deadline policy	A Sun Grid Engine, Enterprise Edition policy which guarantees preferential access to resources to jobs which have to finish before or at a given dead-line. An administrator can determine the level of importance a dead-line job may reach and the set of users who are allowed to submit dead-line jobs.
department	A list of users and groups who are treated alike in the functional and override scheduling policies of Sun Grid Engine, Enterprise Edition. Users and groups may belong to only one department.
entitlement	The same as share (see below). Sun Grid Engine, Enterprise Edition only. The amount of resources being planned to be consumed by a certain job, user, user group or project.
functional policy	A Sun Grid Engine, Enterprise Edition policy which assigns specific levels of importance to jobs, users, user groups, projects and job classes. A high priority project (and all its jobs), for instance, may receive a higher resource share through the functional policy than a low priority project.
group	A UNIX group.
hard resource requirements	The resources which must be allocated before a job may be started. Contrasts with <i>soft resource requirements</i> .
host	A machine on which Sun Grid Engine, Enterprise Edition functions occur.
job	<p>A batch job is a UNIX shell script that can be executed without user intervention and does not require access to a terminal.</p> <p>An interactive job is a session started with the Sun Grid Engine, Enterprise Edition commands, <code>qrsh</code>, <code>qsh</code>, or <code>qlogin</code> that will open an <i>xterm</i> window for user interaction or provide the equivalent of a remote login session, respectively.</p>
job array	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).

job class	A set of jobs that are equivalent in some sense and treated similarly. In Sun Grid Engine, Enterprise Edition a job class is defined by the identical requirements of the corresponding jobs and the characteristics of the queues being suitable for those queues.
manager	A user who can manipulate all aspects of Sun Grid Engine, Enterprise Edition. 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.
override policy	A Sun Grid Engine, Enterprise Edition policy commonly used to override the automated resource entitlement management of the functional, share-based and dead-line policy. Sun Grid Engine, Enterprise Edition can assign override to jobs, users, user groups, job classes and projects.
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, Enterprise Edition configuration entity, which defines the necessary interfaces for Sun Grid Engine, Enterprise Edition 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, Enterprise Edition administrator can use define the behavior of Sun Grid Engine, Enterprise Edition. Policies will be implemented automatically by Sun Grid Engine, Enterprise Edition.
priority	The relative level of importance of a Sun Grid Engine, Enterprise Edition job compared to others.
project	A Sun Grid Engine, Enterprise Edition project.
queue	A container for a certain class and number of jobs being allowed to execute on a Sun Grid Engine, Enterprise Edition 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.

share	The same as entitlement (see above). Sun Grid Engine, Enterprise Edition only. The amount of resources being planned to be consumed by a certain job, user, user group or project.
share-based policy	A Sun Grid Engine, Enterprise Edition policy which allows definition of the entitlements of user and projects and arbitrary groups thereof in a hierarchical fashion. An enterprise, for instance may be subdivided subsequently in divisions, department, projects active in the departments, user groups working on those projects and users in those user groups. The share base hierarchy is called share-tree and once a share-tree is defined, its entitlement distribution is automatically implemented by Sun Grid Engine, Enterprise Edition.
share-tree	The hierarchical definition of a Sun Grid Engine, Enterprise Edition share-based policy.
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 <i>hard resource requirements</i> .
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.
ticket	A generic unit for resource share definition in Sun Grid Engine, Enterprise Edition. The more shares a Sun Grid Engine, Enterprise Edition job, user, project, etc. has, the more important it is. If a job has twice the amount of tickets than another job, for example, the job is entitled to twice the resource consumption.
usage	Another term for “resources consumed.” In the Sun Grid Engine, Enterprise Edition system, the usage is determined by an administrator configurable weighted sum of CPU time consumed, memory occupied over time and amount of I/O performed.
user	May submit jobs to and execute jobs with Sun Grid Engine, Enterprise Edition if he or she has a valid login on at least one submit host and an execution host.
userset	Either an access list (see above) or a department (see above).

PART II Getting Started

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

■ **Chapter 2** – “Installation” on page 43

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

Installation

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

- Full, fresh installation of the Sun Grid Engine, Enterprise Edition software
- Secure installation with special encryption features
- Installation verification

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, Enterprise Edition 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.

Basic Installation Overview

Note – These instructions are for a *fresh* basic Sun Grid Engine, Enterprise Edition 5.3 installation only. For instructions on how to install a new system with additional *security* protection, see the section, “How To Install and Set Up a CSP-Secured System” on page 58. For instructions on how to *upgrade* an existing installation of an older version of the Sun Grid Engine product, see the *Sun Grid Engine, Enterprise Edition 5.3 Release Notes*.

Full installation consists of the following broad tasks.

- Planning the Sun Grid Engine, Enterprise Edition configuration and environment
- Reading the Sun Grid Engine, Enterprise Edition 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, Enterprise Edition system
- Registering information about administrative and submit hosts
- Verifying the installation

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

Phase 1 - Planning

The planning phase of installation consists of the following tasks.

- Deciding whether your Sun Grid Engine, Enterprise Edition environment will be a single cluster or a collection of sub-clusters called *cells*
- Selecting the machines that will be Sun Grid Engine, Enterprise Edition 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, Enterprise Edition users have the same user names on all submit and execution hosts
- Deciding what the Sun Grid Engine, Enterprise Edition directory organization will be. For example, you could decide to organize directories as a complete tree on each workstation, or you could cross-mount directories, or you could set up a partial directory tree on some workstations. You must also decide where each Sun Grid Engine, Enterprise Edition 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 52) 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, Enterprise Edition executes simple commands
- Submitting test jobs

Planning the Installation

Before you begin installing the Sun Grid Engine, Enterprise Edition software, you must carefully plan how 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, Enterprise Edition system.

The Installation Directory `<SGE_ROOT>`

Prepare a directory to read in the contents of the Sun Grid Engine, Enterprise Edition distribution media. This directory is called the Sun Grid Engine, Enterprise Edition *root directory* and later on, while the Sun Grid Engine, Enterprise Edition system is in operation, it 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, Enterprise Edition directory tree. Each Sun Grid Engine, Enterprise Edition component in a cell (see the section, “Cells” on page 50) needs read access to `<SGE_ROOT>/<cell>/common` on startup. See the section, “File Access Permissions” on page 48, 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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition installation procedure will install the Sun Grid Engine, Enterprise Edition system, manuals, spool areas and the configuration files in a directory hierarchy (see FIGURE 2-1, “Sample Directory Hierarchy” on page 47) 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 48.

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).

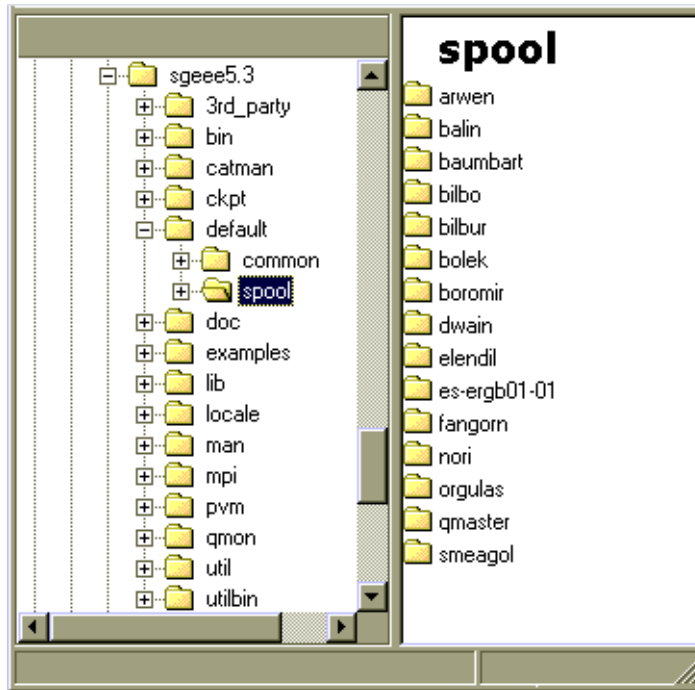


FIGURE 2-1 Sample Directory Hierarchy

Disk Space Requirements

The Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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 167 for instructions).

Installation Accounts

You can install Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition software to run the entire file handling of all Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition root directory. Also, the installation procedure assumes that the host from which you will read in the Sun Grid Engine, Enterprise Edition distribution media can access this directory.

Network Services

Determine whether your site’s network services are defined as an 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, Enterprise Edition 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, Enterprise Edition is controlled. It runs the master daemon, `sge_qmaster`. The master host is central to the operation of Sun Grid Engine, Enterprise Edition functionality, so it must comply with the following requirements.

- 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, Enterprise Edition daemons. 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.
- *Optionally*, it should have the Sun Grid Engine, Enterprise Edition 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, Enterprise Edition daemons to make a host into a shadow host.

Execution Hosts

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

Administrative Hosts

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

Submit Hosts

Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition 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 names on some machines.

Note – The user names 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, Enterprise Edition 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 into which Sun Grid Engine, Enterprise Edition software is installed, most settings created by the Sun Grid Engine, Enterprise Edition installation procedure can be changed during operation of the system on the fly.

In case you are already familiar with Sun Grid Engine, Enterprise Edition 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 191, 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	

FIGURE 2-2 Template Form To Be Filled In Before Installation

2. Ensure that the file system(s) and directories that will contain the Sun Grid Engine, Enterprise Edition 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, Enterprise Edition is distributed on CD-ROM. Ask your system administrator or refer to your local system documentation for information on how to access CD-ROMs. The CD-ROM distribution contains a directory named

Sun_Grid_Engine_Enterprise_5.3. The product distribution is in this directory, in both tar format and the Sun Microsystems pkgadd format. The pkgadd format is the preferred format.

1. **Create the admin user account (see the section, “File Access Permissions” on page 48).**
2. **Provide access to the distribution media and log in to a system—preferably a system that has direct connection to a file server.**
3. **Create the installation directory as described in the section, “The Installation Directory <SGE_ROOT>” on page 45 to read in the Sun Grid Engine, Enterprise Edition installation kit, making sure that the access permissions for the installation directory are set properly.**

In these instructions, the installation directory is abbreviated as *<install_dir>*.

4. **Install the binaries for all binary architectures that will be used by any of your qmaster, execution, and submit hosts in your Sun Grid Engine, Enterprise Edition cluster.**

Depending on which installation method you are using, do one of the following.

pkgadd Method

As you enter the following commands, you must be prepared to respond to script questions about your base directory (the default is */gridware/sge*), the admin user (the default is *sgeadmin*), and the admin user group (the default is *adm*). The script requests the choices you made during the planning steps of this installation (see the section, “How To Plan the Installation” on page 52).

- a. **At the command prompt, enter the following commands, responding to the resulting script questions.**

```
# cd <cdrom_mount_point>/Sun_Grid_Engine_Enterprise_5.3/Packages
# pkgadd -d . SDRMEcomm
# pkgadd -d . SDRMEDoc
# pkgadd -d . SDRMEsp32 (This is optional; at least one binary set is required)
# pkgadd -d . SDRMEsp64 (This is optional; at least one binary set is required)
```

These commands install the following packages.

- SDRMEcomm – For the architecture independent files
- SDRMEDoc – For the documentation
- SDRMEsp32 – For the Solaris (SPARC® platform) 32-bit binaries for Solaris 2.6, Solaris 7, Solaris 8, and Solaris 9 Operating Environments

- SDRMEsp64 – For the Solaris (SPARC platform) 64-bit binaries for Solaris 7, Solaris 8, and Solaris 9 Operating Environments

tar Method

- b. Enter the following commands at the command prompt (in the example, *<tardir>* is the abbreviation for the full directory, *<cdrom_mount_point>/Sun_Grid_Engine_Enterprise_5.3/tar*).**

```
# cd <sge_root>
# gzip -dc <tar_dir>/sgeee-5_3-common.tar.gz | tar xvpf -
# gzip -dc <tardir>/sgeee-5_3-doc | tar xvpf -
# gzip -dc <tardir>/sgeee-5_3-bin-solsparc32.tar.gz | tar xvpf -
# gzip -dc <tardir>/sgeee-5_3-bin-solsparc64.tar.gz | tar xvpf -
# util/setfileperm.sh <adminuser> <admingroup> <sge_root>
```

- The solsparc32 tar file contains the Solaris (SPARC® platform) 32-bit binaries for Solaris 2.6, Solaris 7, Solaris 8, and Solaris 9 Operating Environments.
- The solsparc64 tar file contains the Solaris (SPARC platform) 64-bit binaries for Solaris 7, Solaris 8, and Solaris 9 Operating Environments.

5. 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 tape archive file on the CD-ROM. This will read in the Sun Grid Engine, Enterprise Edition installation kit.

Performing the Basic Installation

The following sections describe how to install all the components of the Sun Grid Engine, Enterprise Edition 5.3 system, including the master, execution, administration, and submit hosts.

Note – If you want to install the system with enhanced security, see the section, “Installing with Increased Security” on page 57 before you continue installation.

▼ How To Install the Master Host

Note – The Sun Grid Engine, Enterprise Edition installation procedure creates a default configuration for the system on which it is executed. It inquires the operating system type hosting the installation and makes meaningful settings based on this information.

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 58).

```
% ./install_qmaster -m
```

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, Enterprise Edition 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 58).

```
% ./install_execd -x
```

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. Also, 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, Enterprise Edition 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 `superuser` 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 169 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 transferring 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 52, “How To Read the Distribution Media” on page 52, “How To Install the Master Host” on page 55, “How To Install Execution Hosts” on page 56, and “How To Install Administration and Submit Hosts” on page 57.

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, “Basic Installation Overview” on page 43, “Planning the Installation” on page 45, and “Performing the Basic Installation” on page 54—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 -m`, 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 -m -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_cal4364.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_cal4364.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_cal4364.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_cal4364.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_cal4364.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 >/var/(...)/userkeys/eddy/rand.seed<
1513428 semi-random bytes loaded
Using configuration from /tmp/sge_cal4364.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_cal4364.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_cal4364.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_cal4364.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_cal4364.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_cal4364.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

```
SGEEE startup script
-----

Your system wide SGEEE 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 56.

Remember to add the `-csp` flag when you call the “`./install_execd`” 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 67.

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 4 in the section, “How To Install Execution Hosts” on page 56.

After completing all remaining installation steps, turn to the instructions in the section, “How To Generate Certificates and Private Keys for Users” on page 67.

▼ 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 `COMMID_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
```

Verifying the Installation

To make sure that the Sun Grid Engine, Enterprise Edition daemons are running, you must look for the `sge_qmaster`, `sge_schedd` and `sge_commd` daemons on the master host and then on the execution hosts. You then attempt to use Sun Grid Engine, Enterprise Edition commands and finally prepare to submit jobs.

▼ 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, Enterprise Edition 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, Enterprise Edition 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 UNIX system, 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, Enterprise Edition 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, Enterprise Edition 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 184). 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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition `qstat` command to monitor the job's behavior.**

See "Submitting Batch Jobs" on page 97 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, "Error Messaging" on page 327.

PART III Using Sun Grid Engine Enterprise Edition 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 165), this part of the *Sun Grid Engine, Enterprise Edition 5.3 Administration and User’s Guide* consists of three chapters.

- **Chapter 3** – “Navigating Through Sun Grid Engine, Enterprise Edition” on page 77

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

- **Chapter 4** – “Submitting Jobs” on page 91

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

- **Chapter 5** – “Checkpointing, Monitoring, and Controlling Jobs” on page 137

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, Enterprise Edition system.

Navigating Through Sun Grid Engine, Enterprise Edition

This chapter introduces you to some basic Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 5.3” on page 23.

This chapter also includes instructions for accomplishing the following tasks.

- “How To Launch the QMON Browser” on page 79
- “How To Display a List of Queues” on page 80
- “How To Display Queue Properties” on page 80
- “How To Find the Name of the Master Host” on page 83
- “How To Display a List of Execution Hosts” on page 83
- “How To Display a List of Administration Hosts” on page 84
- “How To Display a List of Submit Hosts” on page 84
- “How To Display a List of Requestable Attributes” on page 85

Sun Grid Engine, Enterprise Edition User Types and Operations

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

- **Managers** – Managers have full capabilities to manipulate Sun Grid Engine, Enterprise Edition. By default, the superusers of all administrative hosts 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 88, but no cluster or queue management capabilities.

TABLE 3-1 shows the Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition system.

The QMON Browser

Sun Grid Engine, Enterprise Edition features a graphical user interface (GUI) command tool, the QMON browser. The QMON browser provides a myriad of Sun Grid Engine, Enterprise Edition 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 1-4 to identify the meaning of the icons).

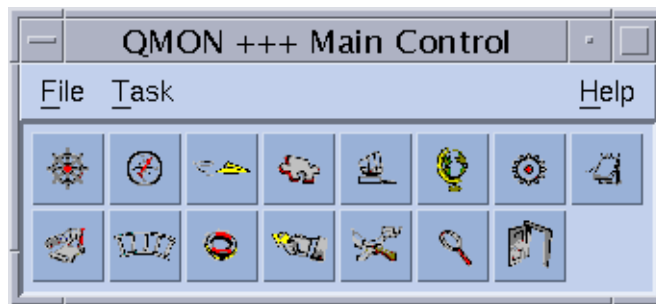


FIGURE 3-1 QMON Main Control Menu

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 35.)

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 158 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.

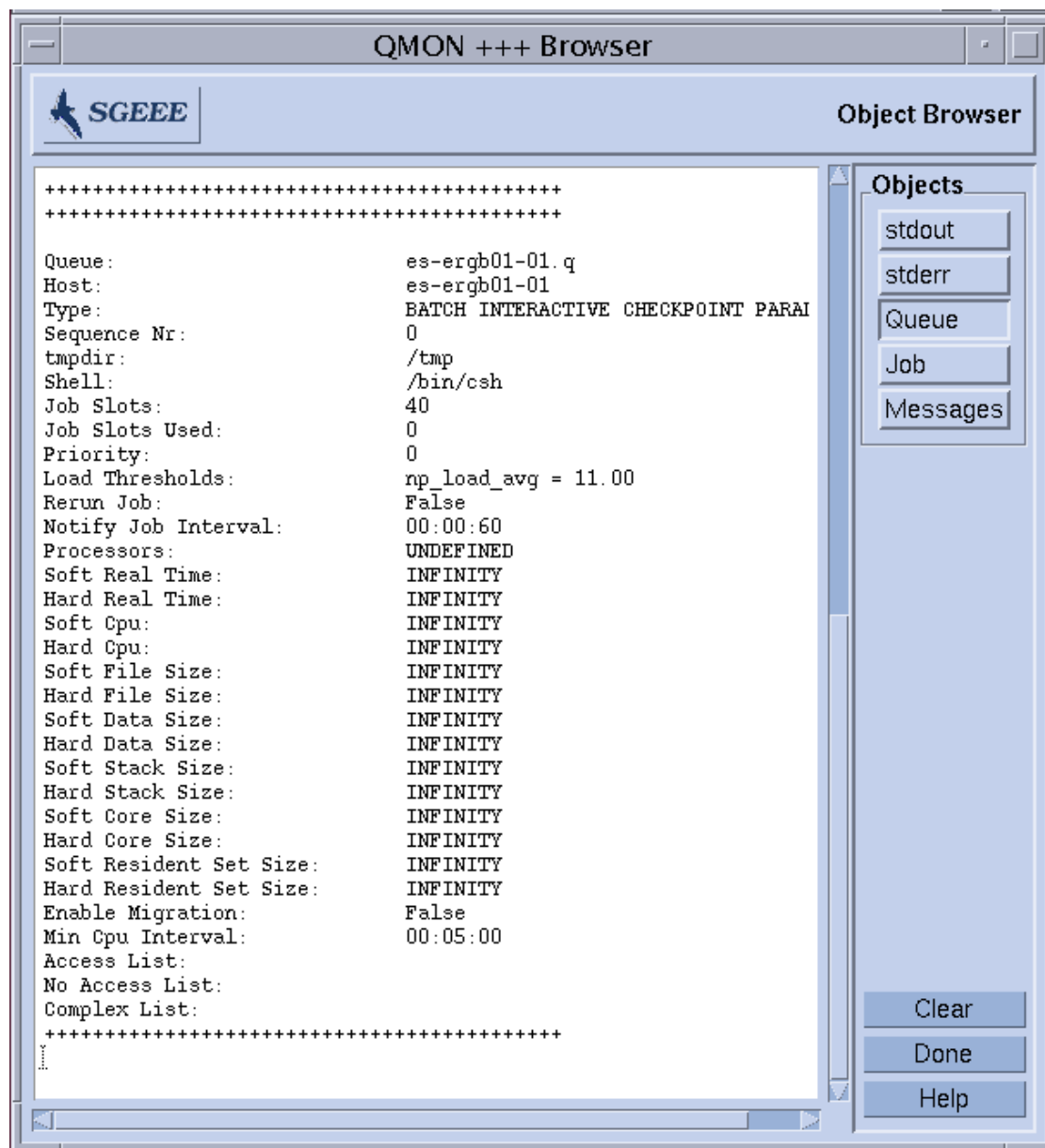


FIGURE 3-2 QMON Browser Display of Queue Properties

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 90
- `user_lists` – The user or group identifiers in the user access lists (see “User Access Permissions” on page 88) 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 88) enlisted under this parameter may *not* access the queue.
- `project_lists` – Jobs submitted with the project identifiers (see “About Projects” on page 258) enlisted under this parameter may access the queue.
- `xproject_lists` – Jobs submitted with the project identifiers (see “About Projects” on page 258) 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 84).
- `complex_values` – Assigns capacities as provided for this queue for certain complex attributes (see “Requestable Attributes” on page 84).

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, Enterprise Edition cluster. However, without Sun Grid Engine, Enterprise Edition 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 169. 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 command:

```
% qconf -sh
```

▼ How To Display a List of Submit Hosts

The list of submit host can be displayed with the command:

```
% qconf -ss
```

Requestable Attributes

When submitting a Sun Grid Engine, Enterprise Edition 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, Enterprise Edition will map these job requirements onto the host and queue configurations of the Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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 78), the list of global and host-related attributes (see “Complex Types” on page 216), as well as administrator-defined attributes. For convenience, however, the Sun Grid Engine, Enterprise Edition 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 97 for detailed information on how to submit jobs). They are enlisted in the Available Resources selection list.

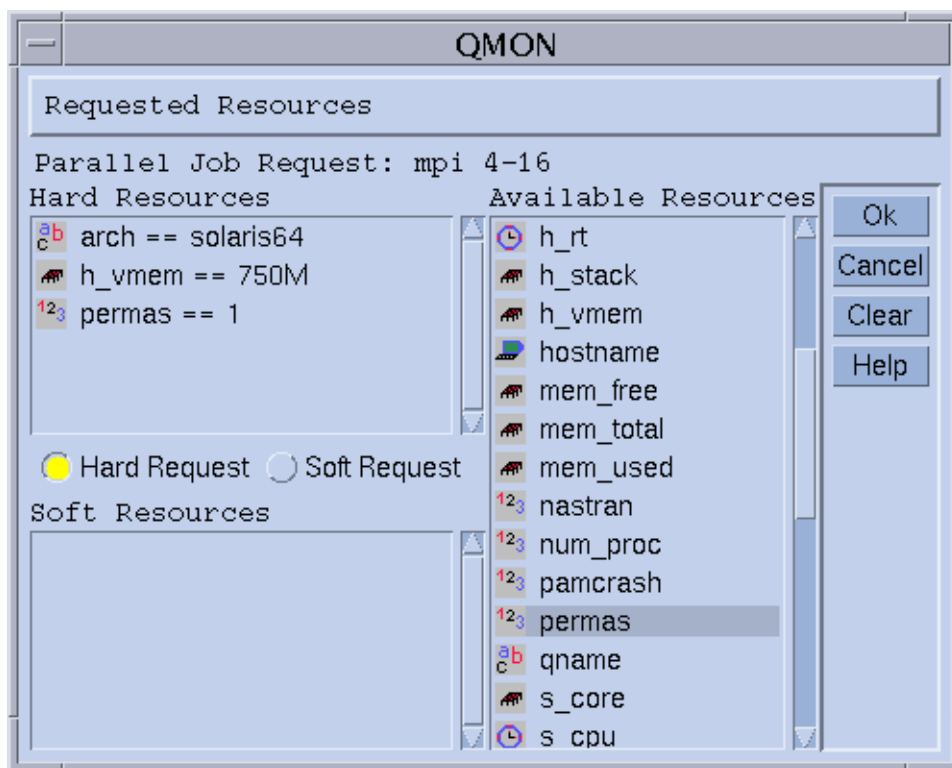


FIGURE 3-3 Requested Resources Dialogue Box

▼ 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 (optional) cluster global attributes
- **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 213 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. The following command displays the attributes of a particular complex.

```
% 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
max_no_migr	mmn	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, Enterprise Edition 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 `ghostname` 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, Enterprise Edition.

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 224). The user requests consumables just like any other attribute. The Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition facilities (e.g., parallel environment interfaces; see “About Parallel Environments” on page 313) can be restricted for certain users or user groups by the Sun Grid Engine, Enterprise Edition administrator.

Note – Sun Grid Engine, Enterprise Edition 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.

ACLs are also used to define Sun Grid Engine, Enterprise Edition projects, to which the corresponding users may have access, i.e. to which they can subordinate their jobs. The administrator can restrict access to cluster resources also on a per project basis.

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 243 for details.

Sun Grid Engine, Enterprise Edition project access can be displayed via the Project Configuration icon in the QMON Main menu. Details are described in section “About Projects” on page 258.

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 78 and “How To Configure PEs with QMON” on page 314. 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.

Sun Grid Engine, Enterprise Edition **project** configurations can be controlled from the command-line via the commands:

```
% qconf -sprjl
% qconf -sprj <project_name>
```

displaying a list of list of defined projects and particular project configurations respectively. The projects are defined via ACLs and you will need to query the ACL configurations as described above.

If you have access to a project, you are allowed to submit jobs subordinated to the project. From the command-line, this can be done via:

```
% qsub -p <project_name> <other options>
```

The cluster, host and queue configurations define project access in the same way as for ACLs via the `project_lists` and `xproject_lists` parameters.

Managers, Operators and Owners

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

```
% qconf -sm
```

and a list of operators by:

```
% qconf -so
```

Note – The superuser of a Sun Grid Engine, Enterprise Edition 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 78. This database can be retrieved by executing:

```
% qconf -sq queue_name
```

The concerning queue configuration entry is called `owners`.

Submitting Jobs

This chapter provides background information about, and instructions for, using Sun Grid Engine, Enterprise Edition 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.

- “How To Run a Simple Job from the Command Line” on page 92
- “How To Submit Jobs From the Graphical User Interface, QMON” on page 93
- “How To Submit Jobs from the Command Line” on page 117
- “How To Submit an Array Job from the Command Line” on page 120
- “How To Submit an Array Job with QMON” on page 120
- “How To Submit Interactive Jobs with QMON” on page 122
- “How To Submit Interactive Jobs With qsh” on page 125
- “How To Submit Interactive Jobs With qlogin” on page 125
- “How To Invoke Transparent Remote Execution with qrsh” on page 127

Running a Simple Job

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

Note – If you have installed the Sun Grid Engine, Enterprise Edition program under an unprivileged account, you must log in as that particular user to be able to run jobs (see “Prerequisite Tasks” on page 45 for details).

▼ How To Run a Simple Job from the Command Line

Prior to executing any Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition settings for all interactive session you will start later.

2. Submit the following simple job script to your Sun Grid Engine, Enterprise Edition cluster.

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

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

A more convenient method of submitting and controlling Sun Grid Engine, Enterprise Edition jobs and of getting an overview of the Sun Grid Engine, Enterprise Edition 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 left on the Job Control button and then the Submit button.

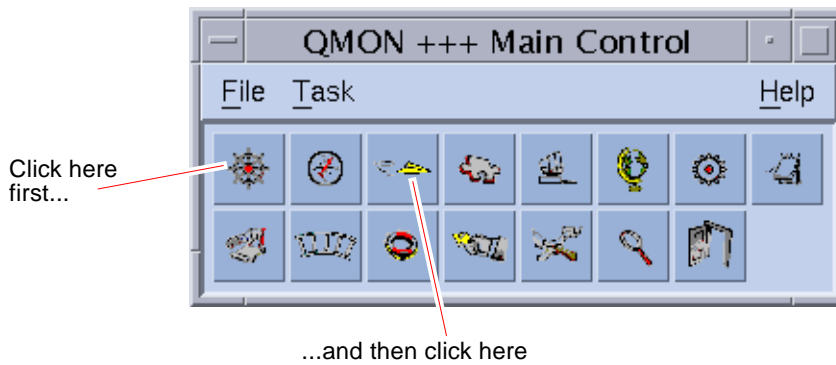


FIGURE 4-1 QMON Main Menu

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.

First, click here to
select the script file...

...then click Submit
to submit the job.

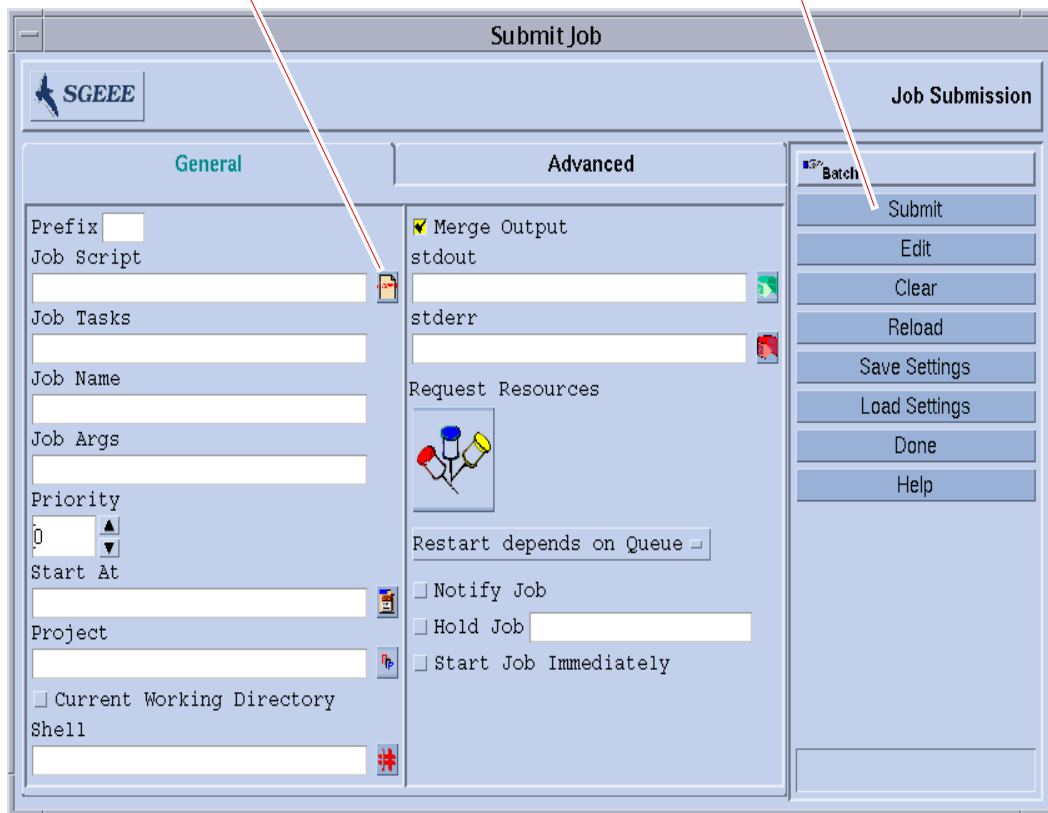


FIGURE 4-2 QMON Job Submission Dialogue Box

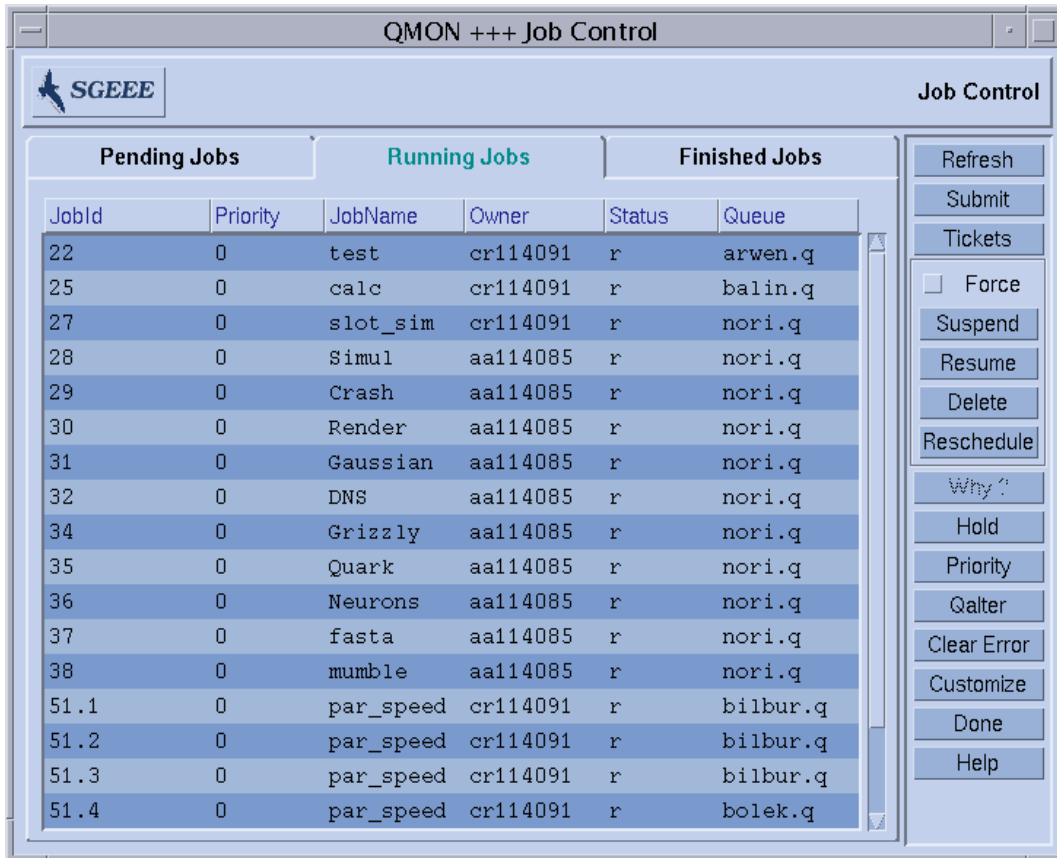


FIGURE 4-3 QMON Job Control Dialogue Box

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

The Job Script Selection box is displayed.

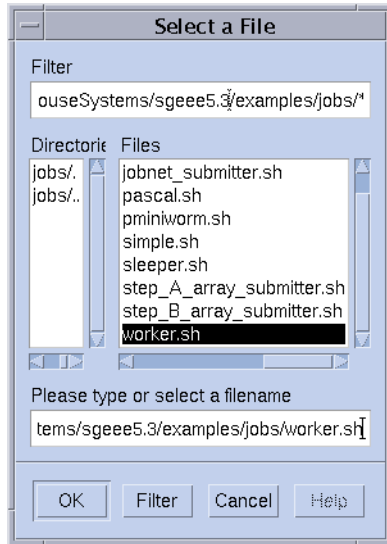


FIGURE 4-4 Job Script Selection Box

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, Enterprise Edition system.

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, Enterprise Edition configuration in effect for the particular host and queue executing the job.

Note – The Sun Grid Engine, Enterprise Edition 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,
# Enterprise Edition.

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, Enterprise Edition.

In general, you can submit to Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition submit options (see the section, “Active Sun Grid Engine, Enterprise Edition Comments” on page 114 for detailed information)
- The job script to be used

Pushing the associated file button opens a file selection box (see FIGURE 4-4).

- The task ID range for submitting array jobs (see “Array Jobs” on page 119)
- The name of the job (a default is set after a job script is selected)
- Arguments to the job script
- A counting box for setting the job’s initial priority

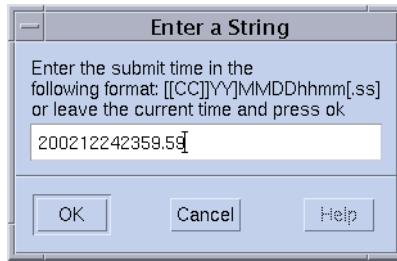
In Sun Grid Engine, Enterprise Edition, this priority ranks a single user’s jobs among themselves. It tells the Sun Grid Engine, Enterprise Edition scheduler how to choose among a single user’s jobs when several jobs are in the system simultaneously.

Note – The administrator has to assign tickets to the functional policies and shares to the functional job category to enable the user to weight among his or her own jobs.

- The time at which the job is to be considered eligible for execution
If the associated file button is pushed, a dialogue box becomes available for entering the correctly formatted time (see FIGURE 4-5)
- The Sun Grid Engine, Enterprise Edition project to which the job is subordinated
The button next to the input field allows the selection among the available projects (see FIGURE 4-6).
- 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 113)
If the associated button is pushed, a dialogue box becomes available for entering the command interpreter specifications of the job (see FIGURE 4-7).
- 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 114)

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 114).

- 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, Enterprise Edition Jobs” on page 143 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 119).
- A flag forcing the job to be either started immediately if possible or being rejected
Jobs are not queued if this flag is selected.



Enter a String

Enter the submit time in the following format: [[CC]]YY]MMDDhhmm[.ss] or leave the current time and press ok

200212242359.59

OK Cancel Help

FIGURE 4-5 At Time Input Box



Select an Item

Available projects

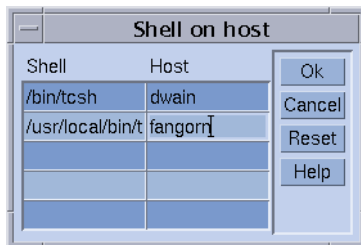
- devel
- crash

Select a project

devel

OK Cancel Help

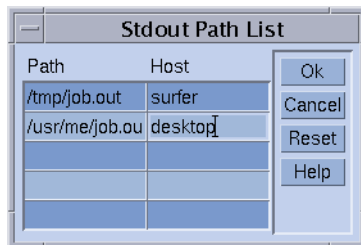
FIGURE 4-6 Project Selection Box



Shell	Host
/bin/tcsh	dwain
/usr/local/bin/t	fangorn

Ok Cancel Reset Help

FIGURE 4-7 Shell Selection Box



Path	Host
/tmp/job.out	surfer
/usr/me/job.ou	desktop

Ok Cancel Reset Help

FIGURE 4-8 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, Enterprise Edition Comments” on page 114), parse default settings (see the section, “Default Requests” on page 118) 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 118).
- **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 4-9 shows the Job Submission dialogue box with most of the parameters set.

Submit Job

SGEE **Job Submission**

General **Advanced**

Prefix # \$

Job Script
flow.sh

Job Tasks

Job Name
Flow

Job Args
big.data

Priority
-111

Start At
200212240000.00

Project
devel

☐ Current Working Directory

Shell
/bin/tcsh

☒ Merge Output
stdout
flow.out
stderr

Request Resources

☐ Restart depends on Queue

☐ Notify Job

☐ Hold Job

☐ Start Job Immediately

Batch

Submit
Edit
Clear
Reload
Save Settings
Load Settings
Done
Help

FIGURE 4-9 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 Sun Grid Engine, Enterprise Edition specific project definition means that the job is subordinated to project `devel`. 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
- 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 4-10). 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 4-11), which can be used to store and communicate job related information accessible anywhere from within a Sun Grid Engine, Enterprise Edition 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 137)

- 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, Enterprise Edition 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 are currently defined for jobs.
- 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 4-12).

- 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 101.

- 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*.

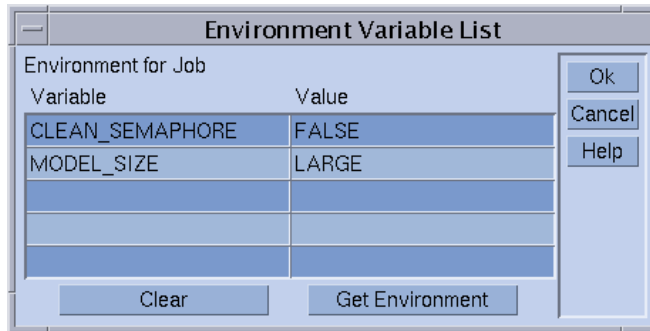
- An ID-list of jobs which need to be finished successfully before the job to be submitted can be started

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

- The deadline initiation time for deadline jobs

Deadline initiation defines the point in time at which a deadline job must have reached maximum priority to finish before a given deadline. It is recommended to subtract a conservative estimation for the runtime (at maximum priority) of a deadline job from its desired deadline time to determine the deadline initiation time. Clicking the button next to the Deadline input window opens the helper dialogue box shown in FIGURE 4-13.

Note – Not all Sun Grid Engine, Enterprise Edition users are allowed to submit deadline jobs. Ask your system administrator if you are permitted to submit deadline jobs. Also contact the cluster administrator for information about the maximum priority that is given to deadline jobs.



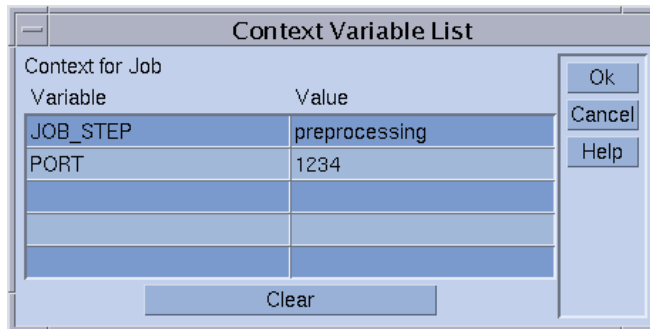
Environment for Job

Variable	Value
CLEAN_SEMAPHORE	FALSE
MODEL_SIZE	LARGE

Clear Get Environment

Ok
Cancel
Help

FIGURE 4-10 Job Environment Definition



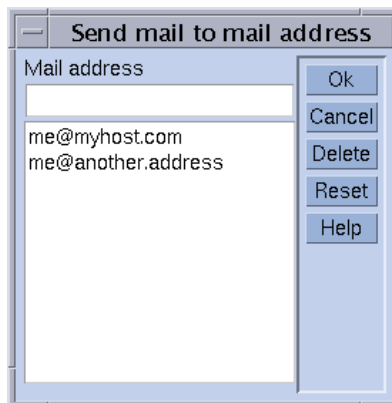
Context for Job

Variable	Value
JOB_STEP	preprocessing
PORT	1234

Clear

Ok
Cancel
Help

FIGURE 4-11 Job Context Definition



Send mail to mail address

Mail address

me@myhost.com
me@another.address

Ok
Cancel
Delete
Reset
Help

FIGURE 4-12 Mail Address Specification

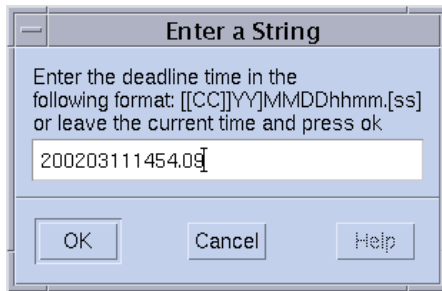


FIGURE 4-13 Deadline Time Input Box

The job defined in FIGURE 4-14 has the following additional characteristics as compared to the job definition from the section, “Extended Example” on page 100.

- 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 4-14 shows an example of an advanced job submission.

Submit Job

SGEEE

Job Submission

General

Advanced

Batch

Parallel Environment

mpi 4-16

Environment

CLEAN_SEMAPHORE=FALSE,MODEL_S:

Context

JOB_STEP=preprocessing,PORT=1:

Checkpoint Object

Account

FLOW

Verify Mode

Warning

Mail

☒ Start of Job
☒ End of Job
☐ Abort of Job
☐ Suspend of Job

Mail To

me@myhost.com,me@another.addr

Hard Queue List

Soft Queue List

big_q

Master Queue List

Job Dependencies

Deadline

Submit

Edit

Clear

Reload

Save Settings

Load Settings

Done

Help

FIGURE 4-14 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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition 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 84. 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 4-15 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 4-15 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.

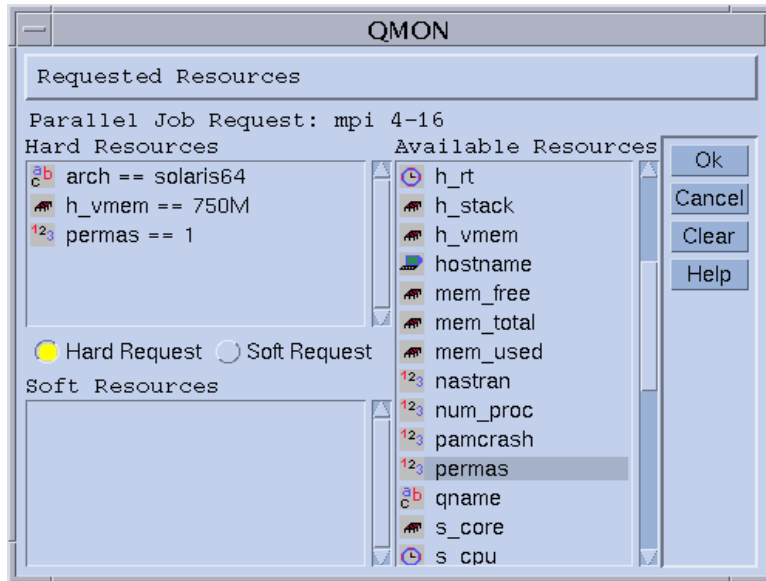


FIGURE 4-15 Requested Resources Dialogue Box

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, Enterprise Edition 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 Sun Grid Engine, Enterprise Edition Allocates Resources

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

1. Read in and parse all default request files (see the section, "Default Requests" on page 118).
2. Process the script file for embedded options (see the section, "Active Sun Grid Engine, Enterprise Edition Comments" on page 114).
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, Enterprise Edition 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, Enterprise Edition 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 4-9). 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 78 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, Enterprise Edition 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, Enterprise Edition.

In case of array job tasks (see the section, “Array Jobs” on page 119), 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 4-14 and FIGURE 4-8 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, Enterprise Edition Comments

Lines with a leading `#` sign are treated as comments in shell scripts. Sun Grid Engine, Enterprise Edition, 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, Enterprise Edition

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, Enterprise Edition default
#shell
#$ -S /bin/csh
# This is a sample script file for compiling and
# running a sample FORTRAN program under Sun Grid Engine,
# Enterprise Edition.
# We want Sun Grid Engine, Enterprise Edition 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, Enterprise Edition job is run, a number of variables are preset into the job’s environment, as listed below.

- **ARC** – The Sun Grid Engine, Enterprise Edition 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, Enterprise Edition root directory as set for `sge_execd` before start-up or the default `/usr/SGE`
- **SGE_CELL** – The Sun Grid Engine, Enterprise Edition 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`
- **LAST_HOST** – The name of the preceding host in case of migration of a checkpointing job
- **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, Enterprise Edition
 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 92—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 4-9—would look as follows:

```
% qsub -N Flow -p -l11 -P devel -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 4-14, for example, would look as follows:

```
% qsub -N Flow -p -l11 -P devel -a 200012240000.00 -cwd \  
-S /bin/tcsh -o flow.out -j y -pe mpi 4-16 \  
-v SHARED_MEM=TRUE,MODEL_SIZE=LARGE \  
-ac JOB_STEP=preprocessing,PORT=1234 \  
-A FLOW -w w -r y -m s,e -q big_q\  
-M me@myhost.com,me@other.address \  
flow.sh big.data
```

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, Enterprise Edition Comments” on page 114) or can utilize so called *default requests*.

The cluster administration may setup a default request file for all Sun Grid Engine, Enterprise Edition 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, Enterprise Edition jobs in a single or multiple lines. The location of the cluster global default request file is `<sgc_root>/<cell>/common/sgc_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 may 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, Enterprise Edition *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, Enterprise Edition, 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, Enterprise Edition. 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 93, 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 93. The only difference is that the Job Tasks input window shown in FIGURE 4-9 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, Enterprise Edition Jobs” on page 143 and “Controlling Sun Grid Engine, Enterprise Edition Jobs from the Command Line” on page 156, as well as the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* sections about `qstat`, `qhold`, `qrsls`, `qmod`, and `qdel`, contain the pertinent information about monitoring and controlling Sun Grid Engine, Enterprise Edition jobs in general and array jobs in particular.

Note – Array jobs offer full access to all Sun Grid Engine, Enterprise Edition 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, Enterprise Edition system to create interactive job.

- `qlogin` – This is a telnet-like session that is started on a host selected by Sun Grid Engine, Enterprise Edition software.
- `qssh` – 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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition cluster is too busy currently.

This default behavior can be changed with the `-now no` option to `qsh`, `qlogin` and `qcrsh`. 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 191 for details).

The subsequent sections outline the usage of the `qlogin` and `qsh` facilities. The `qcrsh` command is explained in a broader context in the section, “Transparent Remote Execution” on page 125.

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, Enterprise Edition.

▼ 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 4-16 and FIGURE 4-17).

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 97. The basic difference is that several input fields are set insensitive because they do not apply for interactive jobs

Submit Job

Job Submission

General

Prefix

Job Script

Job Tasks

Job Name

Job Args

Priority

Start At

Project

☐ Current Working Directory

Shell

Advanced

☐ Merge Output

stdout

stderr

Request Resources

Restart depends on Queue ☐

☐ Notify Job

☐ Hold Job

☒ Start Job Immediately

Submit

Edit

Clear

Reload

Save Settings

Load Settings

Done

Help

FIGURE 4-16 Interactive Job Submission Dialogue Box, General

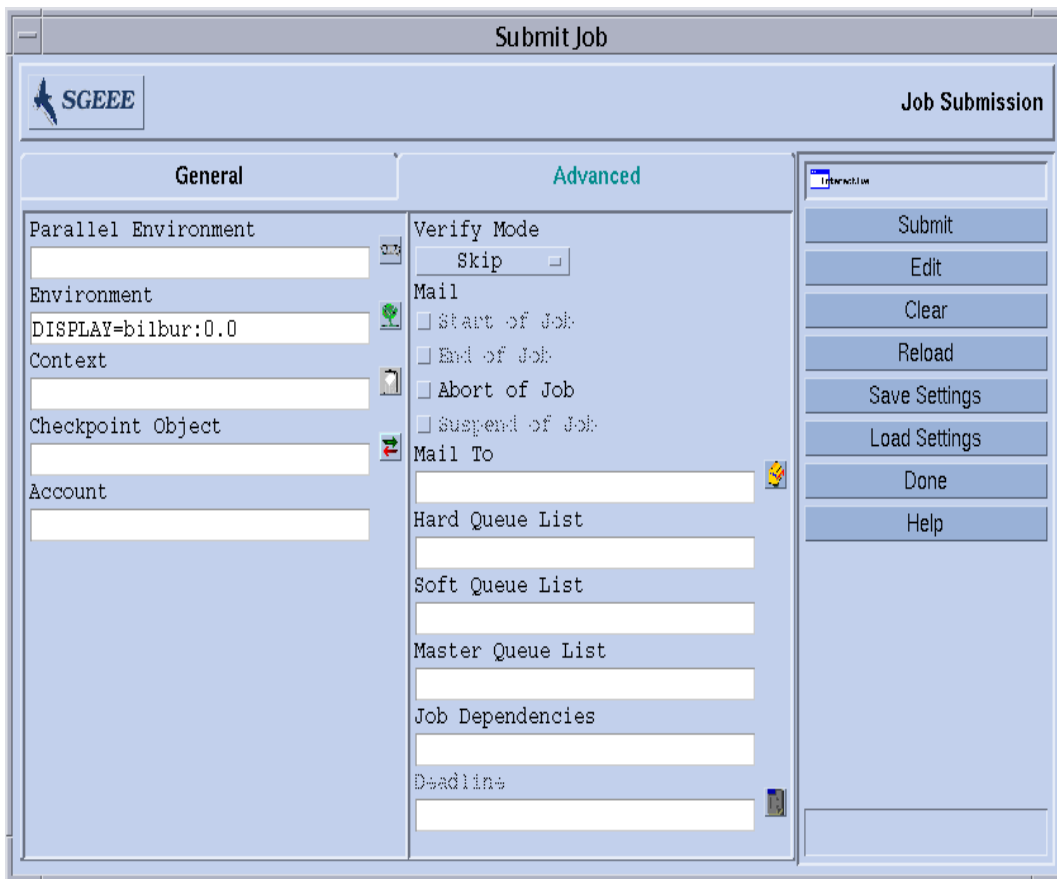


FIGURE 4-17 Interactive Job Submission Dialogue Box, Advanced

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, Enterprise Edition.

▼ 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, Enterprise Edition system, you may have to enter your user name, your password, or both at a login prompt.

Transparent Remote Execution

Sun Grid Engine, Enterprise Edition 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 126. Building on top of `qrsh`, two high level facilities—`qtcsh` and `qmake`—allow the transparent distribution of implicit computational tasks via Sun Grid Engine, Enterprise Edition, thereby enhancing the standard UNIX facilities `make` and `csh`. `qtcsh` is explained in the section, "Transparent Job Distribution with `qtcsh`" on page 127 and `qmake` is described in the section, "Parallel Makefile Processing with `qmake`" on page 130.

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, Enterprise Edition comparable to the standard UNIX facility, `rsh` (also called `remsh` for HP-UX).
- To offer interactive login session capabilities via Sun Grid Engine, Enterprise Edition similar to the standard UNIX facility, `rlogin` (note that `qlogin` is still required as a Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition Software" on page 324).

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, Enterprise Edition with parallel environments such as MPI or PVM.

▼ How To Invoke Transparent Remote Execution with `qcrsh`

- Enter the `qcrsh` command, adding options and arguments as guided by the following synopsis.

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

`qcrsh` 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` – `qcrsh` does not go through the Sun Grid Engine, Enterprise Edition 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 `qcrsh` commonly is used within `qmake` and within a tight parallel environment integration. The default is not to inherit external job resources.
- `-noshell` – With this option, you do not start the command line given to `qcrsh` in a user's login shell, but execute it without the wrapping shell. The option can be used to speed up execution as some overhead, such as the shell startup and sourcing the shell resource files, is avoided.
- `-nostdin` – Suppress the input stream, `STDIN`. With this option set, `qcrsh` will pass the `-n` option to the `rsh(1)` command. This is especially useful if multiple tasks are executed in parallel using `qcrsh`; e.g., in a `make(1)` process. Which process would get the input would be undefined.
- `-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 (`csch`) derivative `tcsh` (`qcrsh` 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,

Enterprise Edition. 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, Enterprise Edition 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, Enterprise Edition.
- 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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition via `qrsh` taking the resource requirements specified in the `qmake` command-line into account. Sun Grid Engine, Enterprise Edition 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 architecture 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, Enterprise Edition 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 `qmake` command-line syntax (without Sun Grid Engine, Enterprise Edition 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`.

How Sun Grid Engine, Enterprise Edition Jobs Are Scheduled

Sun Grid Engine, Enterprise Edition software's policy management automatically controls the use of shared resources in the cluster to best achieve the goals of the administration. High priority jobs will be dispatched preferentially and receive better access to resources. The administration of a Sun Grid Engine, Enterprise Edition cluster can define high-level utilization policies. The available policies follow.

- **Functional** – Special treatment will be accorded because of affiliation with a certain user group, project, etc.
- **Share-based** – Level of service depends on an assigned share entitlement, the corresponding shares of other users and user groups, the past usage of resources by all users and the current presence of users in the system.
- **Deadline** – A job has to be finished before or at a certain point in time and may require special treatment in order to achieve this.

- **Override** – Manual intervention by the Sun Grid Engine, Enterprise Edition cluster administrator modifies the automated policy implementation.

Sun Grid Engine, Enterprise Edition software can be set up to routinely use either a share-based policy, a functional policy, or both. These policies can be combined in any proportion, from giving zero weight to one and using only the second to giving both equal weight.

Along with the routine policies, jobs may be submitted with an initiation deadline (see description of the deadline submission parameter under “Advanced Example” on page 105). Deadline jobs perturb routine scheduling. Administrators may also *override* share-based, functional and initiation deadline scheduling temporarily. An override may be applied to an individual job, or all jobs associated with a user, a department, a project, or a job class.

Job Priorities

In addition to the four policies for mediating among all jobs, Sun Grid Engine, Enterprise Edition sometimes lets users set priorities among their own jobs. A user submitting several jobs may say, for example, that job 3 is the most important and jobs 1 and 2 are equally important but less important than job 3.

Note – This is possible only if Sun Grid Engine, Enterprise Edition software’s combination of policies include the functional policy with shares granted to the functional category “jobs”.

Priorities for jobs are set via the QMON general job submission screen parameter Priority (see FIGURE 4-10) or via the `-p` option to `qsub`. A priority range of -1024 (lowest) to 1023 (highest) can be given. This priority ranks a single user’s jobs among themselves. It tells the Sun Grid Engine, Enterprise Edition scheduler how to choose among a single user’s jobs when several jobs are in the system simultaneously. The relative importance assigned to a particular job depends on the maximum and minimum priorities given to any of that user’s jobs and on the priority value of the specific job.

Tickets

Scheduling policies are implemented with tickets. Each policy has a pool of tickets from which it allocates tickets to jobs entering the multi-machine Sun Grid Engine, Enterprise Edition system. Each routine policy that is in force allocates some tickets to each new job and possibly reallocates tickets to the executing job at each scheduling interval. The criteria each policy uses to allocate tickets are explained below.

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

Tickets are allocated to the routine policies at system configuration by Sun Grid Engine, Enterprise Edition managers. Managers and operators may change ticket allocations at any time. Additional tickets are injected into the system temporarily to indicate a deadline or an override. Policies are combined by assignment of tickets—when tickets are allocated to multiple policies a job gets a portion of its tickets, which indicate its importance, from each policy in force.

Sun Grid Engine, Enterprise Edition grants tickets to jobs entering the system to indicate their importance under each policy in force. Each executing job may gain (for example, from an override or because a deadline is approaching), lose (for example, because it is getting more than its fair share of resources) or keep the same number of tickets at each scheduling interval. The number of tickets a job holds represent the resource share Sun Grid Engine, Enterprise Edition tries to grant that job during each scheduling interval.

The number of tickets a job holds can be displayed via `QMON` ("How To Monitor and Control Jobs with QMON" on page 143) or via `qstat -ext`. The `qstat` command also displays the priority value assigned to a job; for example, via `qsub -p` (see the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for further details on `qstat`).

Queue Selection

The Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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 queueing 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, Enterprise Edition scheduler configuration entry `queue_sort_method` to `seqno`, 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.

Checkpointing, Monitoring, and Controlling Jobs

After you have submitted jobs by way of the Sun Grid Engine, Enterprise Edition 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 tasks.

- “How To Submit, Monitor, or Delete a Checkpointing Job from the Command Line” on page 140
- “How To Submit a Checkpointing Job with QMON” on page 141
- “How To Monitor and Control Jobs with QMON” on page 143
- “How To Monitor Jobs with qstat” on page 153
- “How To Monitor Jobs by Electronic Mail” on page 156
- “How To Control Jobs from the Command Line” on page 157
- “How To Control Queues with QMON” on page 158
- “How To Control Queues with qmod” on page 162

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, Enterprise Edition supports operating system checkpointing if available. Please refer to the Sun Grid Engine, Enterprise Edition 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, Enterprise Edition's migration and dynamic load balancing mechanism. If requested, checkpointing Sun Grid Engine, Enterprise Edition jobs are aborted on demand and migrated to other machines in the Sun Grid Engine, Enterprise Edition 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 140).

A migrating job moves back to `sge_qmaster` and is subsequently dispatched to another suitable queue if any is available. In such a case, the `qstat` output shows `R` as the status.

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, Enterprise Edition 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, Enterprise Edition
#$ -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 305) 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 `sgexecd` 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 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, Enterprise Edition Jobs from the Command Line” on page 156.

▼ How To Submit a Checkpointing Job with QMON

- Follow the instructions in “Advanced Example” on page 105, 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 105, 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 305.

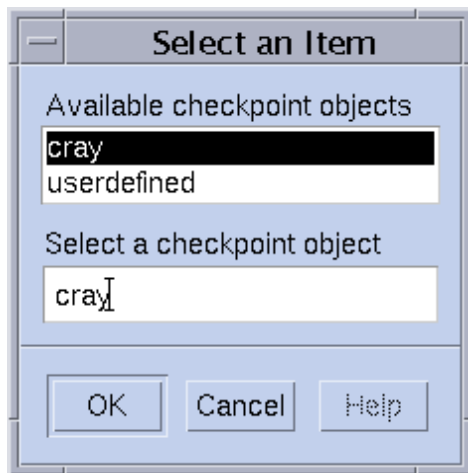


FIGURE 5-1 Checkpoint Object Selection

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, Enterprise Edition Jobs

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

- With the Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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.

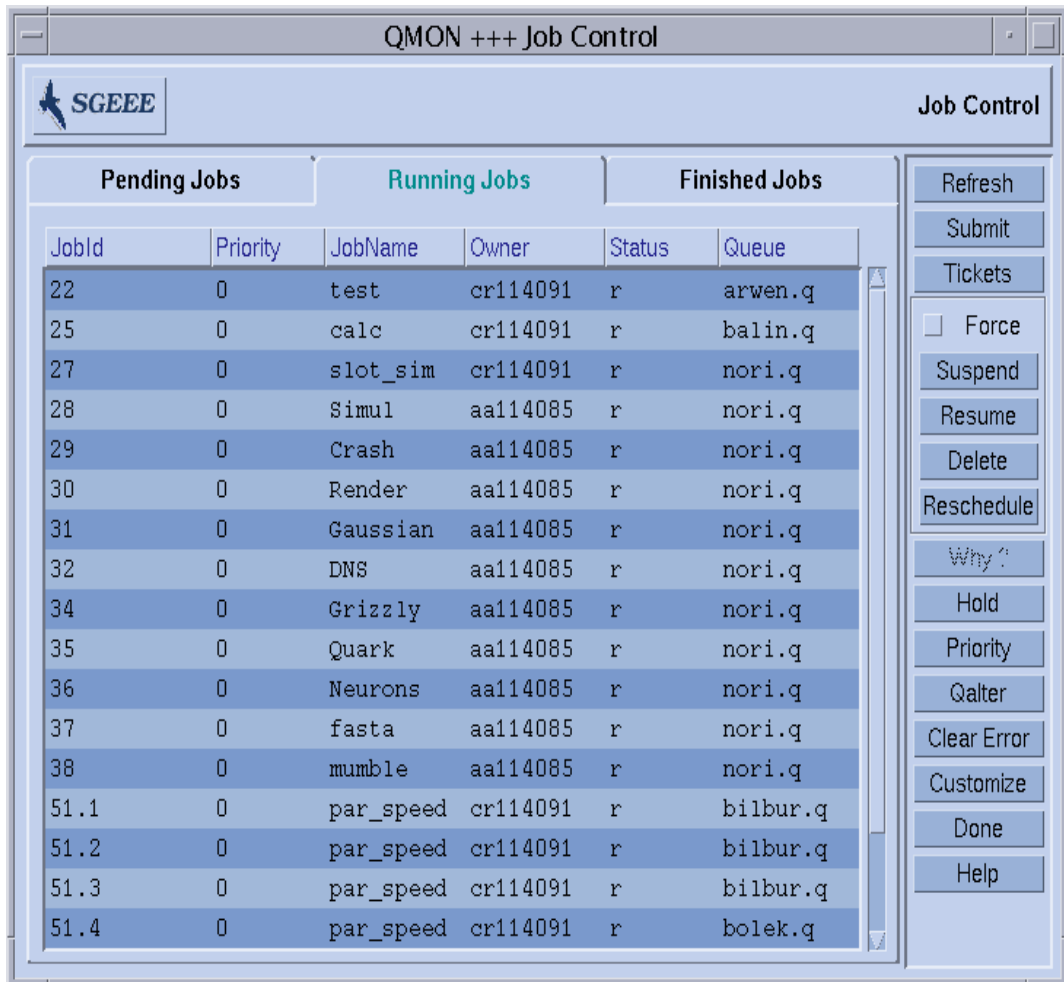


FIGURE 5-2 Job Control Dialogue Box—Standard Form

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.

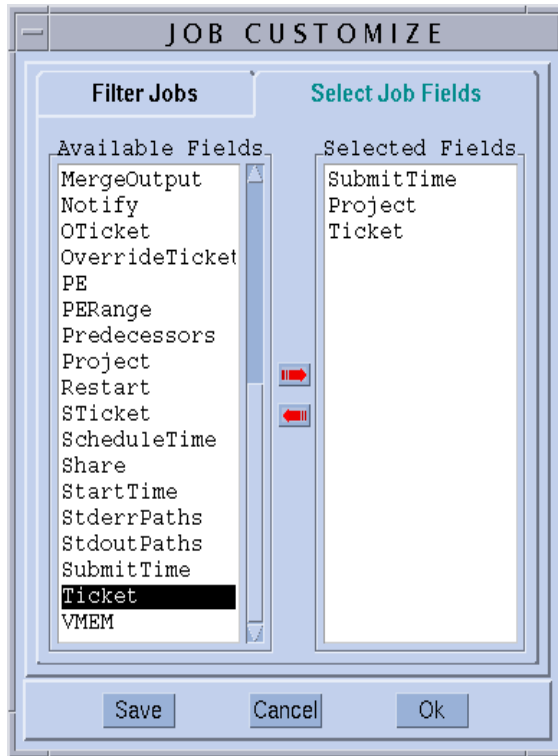


FIGURE 5-3 Job Control Customization Dialogue Box

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

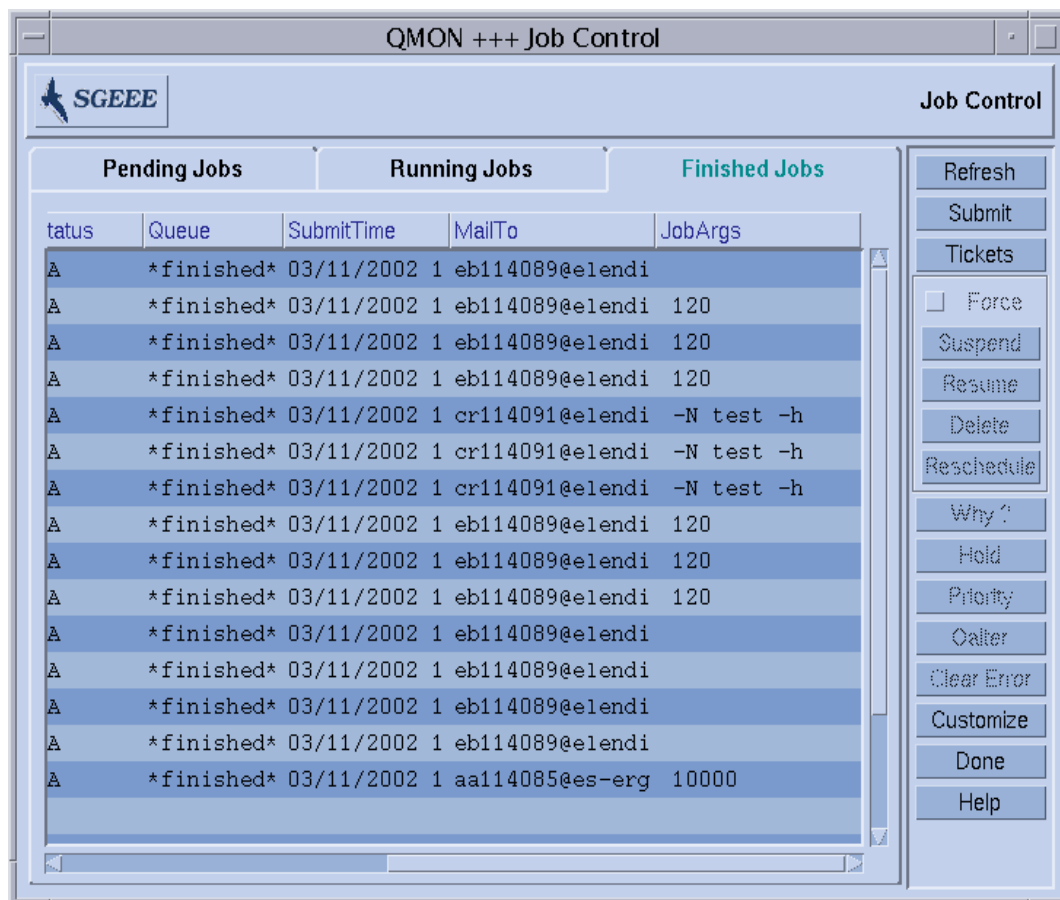


FIGURE 5-4 Job Control Dialogue Box Finished Jobs—Enhanced

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

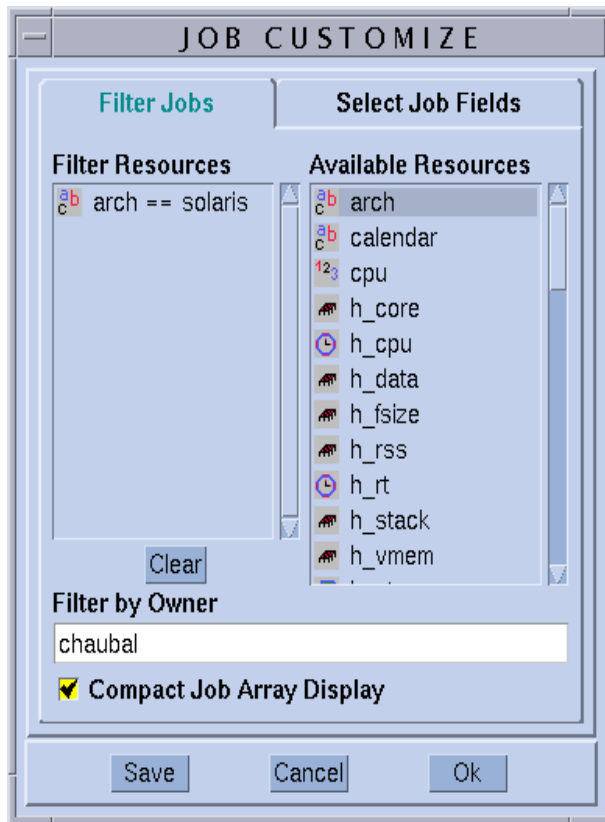


FIGURE 5-5 Job Control Filtering

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

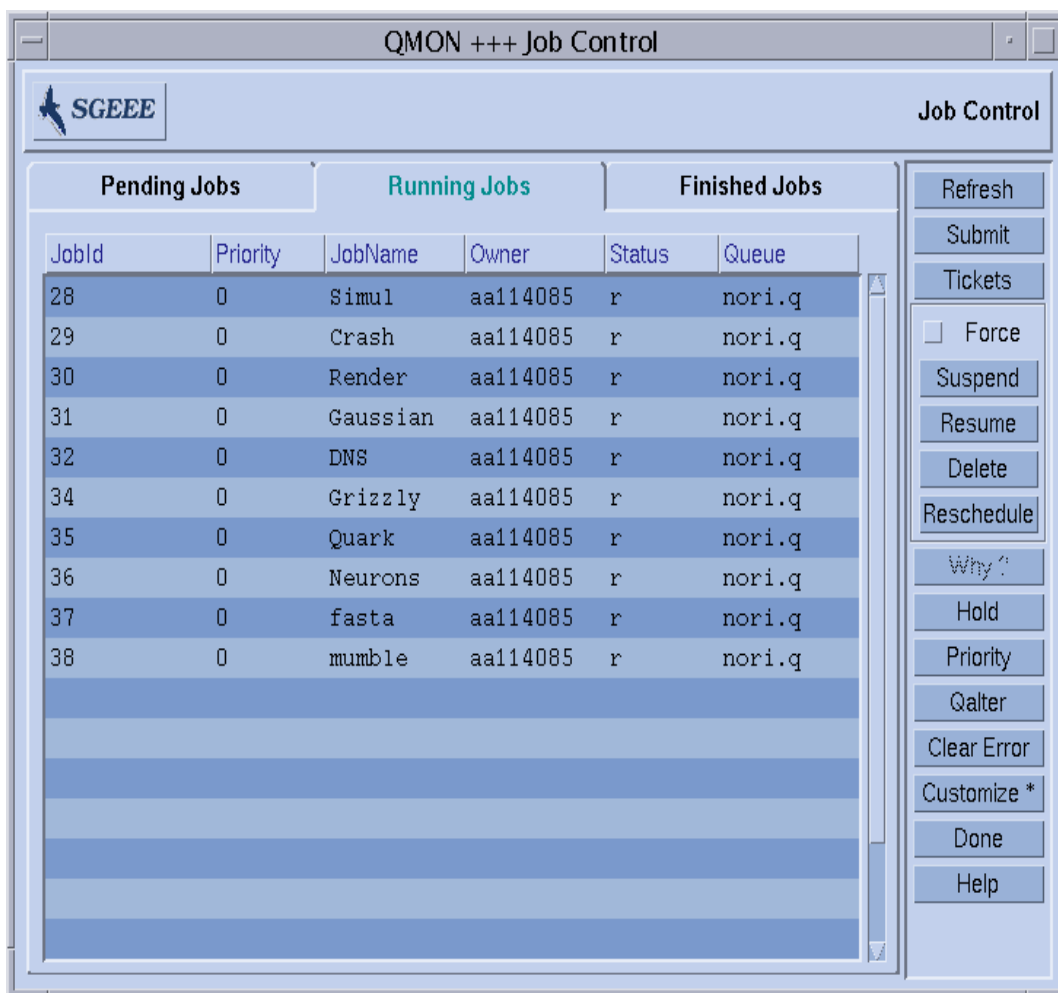


FIGURE 5-6 Job Control Dialogue Box—After Filtering

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, Enterprise Edition managers and operators (see “Managers, Operators and Owners” on page 90). 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 you use the *Hold* button on a selected pending job, the Set Hold sub-dialogue box is opened (see FIGURE 5-7).

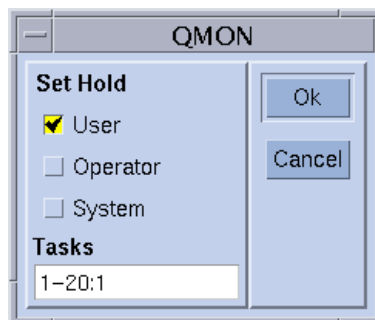


FIGURE 5-7 Job Control Holds

The Set Hold sub-dialogue box enables setting and resetting user, system, and operator holds. User holds can be set or reset by the job owner as well as Sun Grid Engine, Enterprise Edition operators and managers. Operator holds can be set or reset by managers and operators, and manager holds can be set or reset by managers only. As long as any hold is assigned to a job, it is not eligible for execution. Alternate ways to set or 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*).

The *Tasks* field of the Set Hold button applies to Array jobs. You use this button to place a hold on particular subtasks of an array job. Note the format of the text in the Tasks field in FIGURE 5-7. The task id range specified in this field may be a single number, a simple range of the form *n-m*, or a range with a step size. Hence, the task id range specified by, for example, `2-10:2` would result in the task id indexes 2, 4, 6, 8, and 10; i.e., in a total of five identical tasks with the environment variable `SGE_TASK_ID` containing one of the five index numbers each. For detailed information about job holds, see the `qsub` entries in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual*, or the `qsub(1)` man page.

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, as well as running, jobs for Sun Grid Engine, Enterprise Edition. In Sun Grid Engine, Enterprise Edition, the priority ranks a single user's jobs among themselves. It tells the Sun Grid Engine, Enterprise Edition scheduler how to choose among a single user's jobs when several jobs are in the system simultaneously.

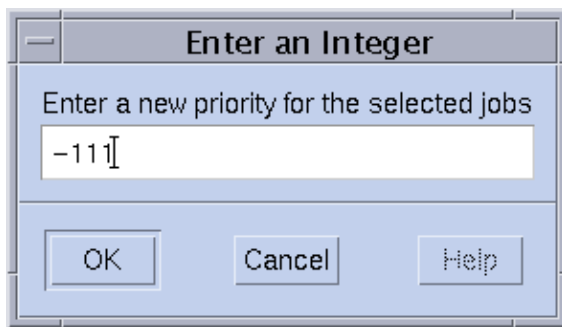


FIGURE 5-8 Job Control Priority Definition

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 93 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, Enterprise Edition 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 105 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, Enterprise Edition scheduler from dispatching the job in its most recent pass. An example browser screen displaying such a message is shown in FIGURE 5-9.

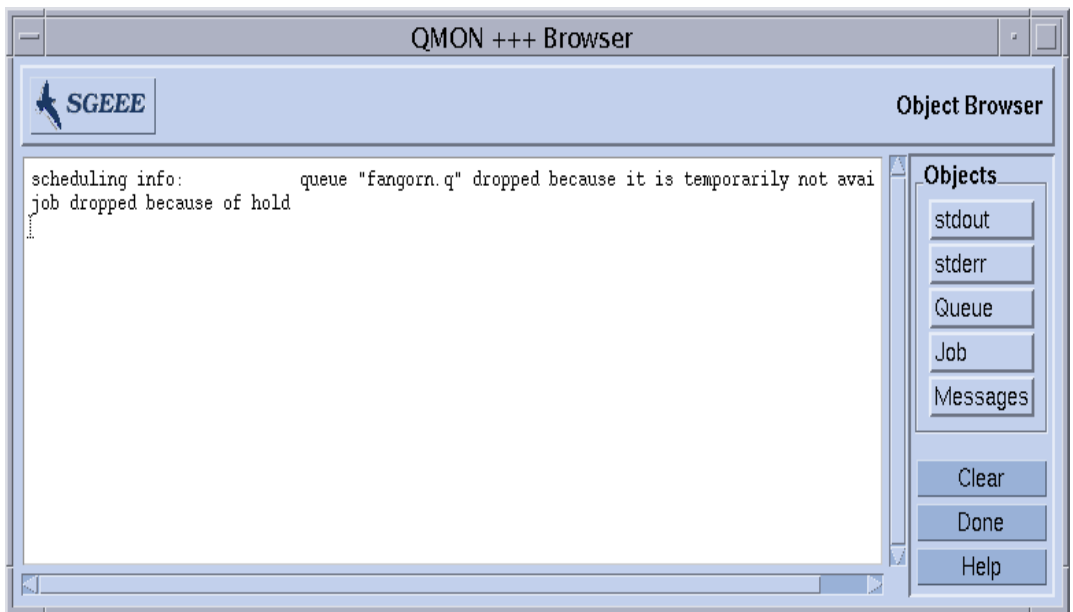


FIGURE 5-9 Browser Displaying Scheduling Information

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, Enterprise Edition 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 143.

The Object Browser is opened upon pushing the Browser icon button in the QMON main menu. The browser displays information about Sun Grid Engine, Enterprise Edition 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.

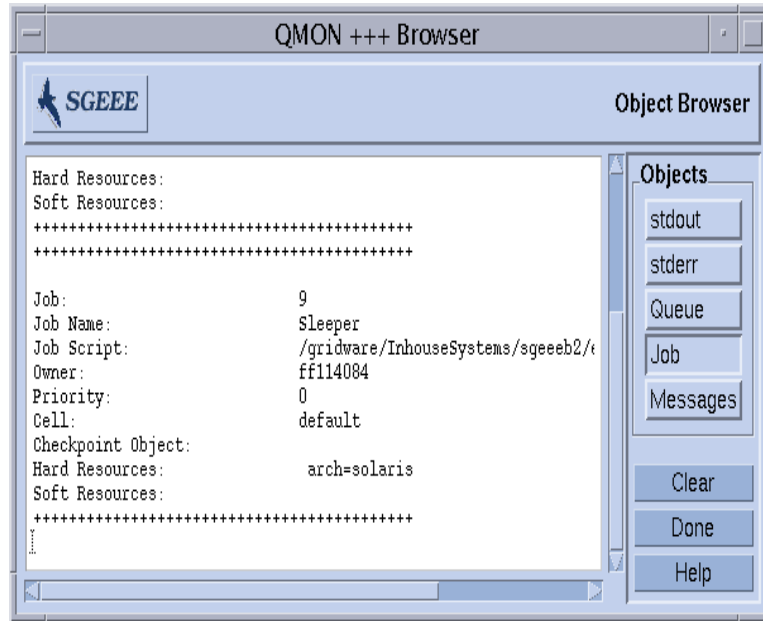


FIGURE 5-10 Object Browser—Job

▼ 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
% qstat -ext
```

The first form provides an overview of the submitted jobs only (see TABLE 5-1). The second form includes information about the currently configured queues in addition (see TABLE 5-2). The third form contains details such as up-to-date job usage and tickets assigned to a job.

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 *sgc_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*(larm), *s*(uspended), *d*(isabled), *E*(rror), or combinations thereof.

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

In the third, Sun Grid Engine, Enterprise Edition specific form, the usage and ticket values assigned to a job are contained in the following columns.

- *cpu/mem/io* – This is the currently accumulated CPU, memory, and I/O usage.
- *tckts/ovrts/otckt/dtckt/ftckt/stckt* – These values relate to the tickets assigned to the job in total, via *galter -ot*, through the override policy, through a deadline, through the functional policy and via the share-based policy.

In addition, the deadline initiation time is displayed in the column *deadline* (if applicable) and the *share* column shows the current resource share which each job has with respect to the usage generated by all jobs in the cluster. See the *qstat* manual page for further detail.

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 110 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 and TABLE 5-2 show examples of output from *qstat* and *qstat -f* commands.

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

queue	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 105.

Controlling Sun Grid Engine, Enterprise Edition Jobs from the Command Line

The section “How To Monitor and Control Jobs with QMON” on page 143 explains how Sun Grid Engine, Enterprise Edition jobs can be deleted, suspended and resumed with the Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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 153).

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, Enterprise Edition manager or operator (see “Managers, Operators and Owners” on page 90).

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, Enterprise Edition 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 78, 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, Enterprise Edition 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.

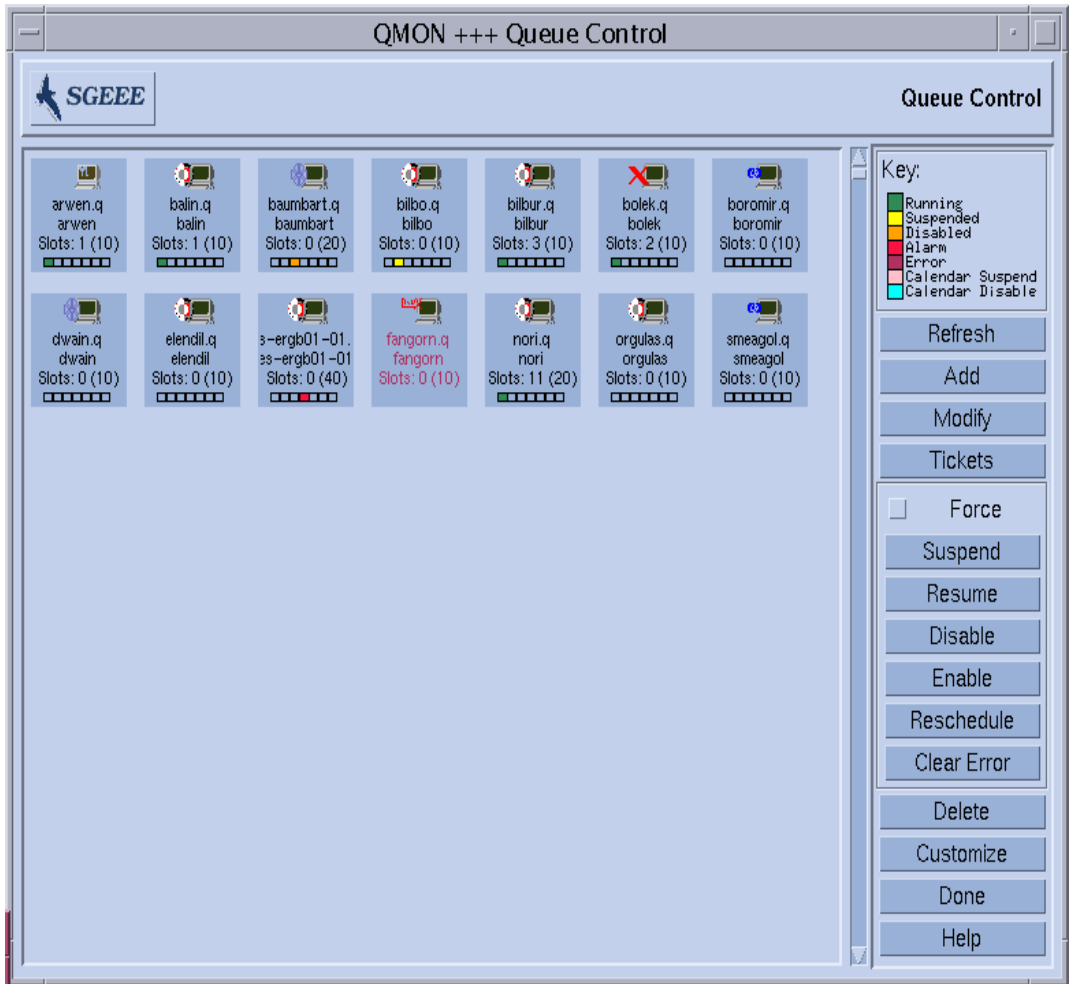


FIGURE 5-11 Queue Control Dialogue Box

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, Disable, or 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 143. 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.

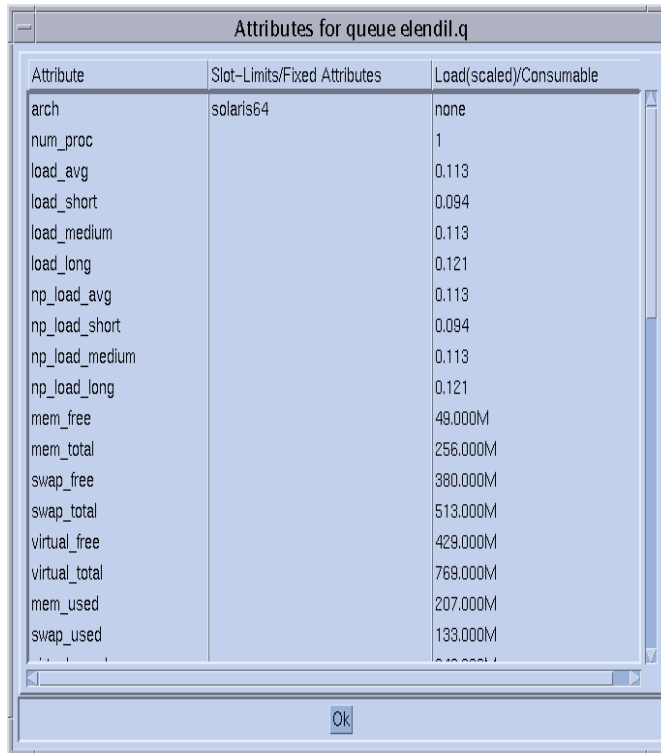
Queues which are disabled are closed, however, the jobs executing in those queues are allowed to continue. To disable a queue is commonly used to “drain” a queue. After the queue is enabled, it is eligible for job execution again. No action on still executing jobs is performed.

The suspend/unsuspend and disable/enable operations require queue owner or Sun Grid Engine, Enterprise Edition manager or operator permission (see the section, “Managers, Operators and Owners” on page 90).

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 192 for details).



Attribute	Slot-Limits/Fixed Attributes	Load(scaled)/Consumable
arch	solaris64	none
num_proc		1
load_avg		0.113
load_short		0.094
load_medium		0.113
load_long		0.121
np_load_avg		0.113
np_load_short		0.094
np_load_medium		0.113
np_load_long		0.121
mem_free		49.000M
mem_total		256.000M
swap_free		380.000M
swap_total		513.000M
virtual_free		429.000M
virtual_total		769.000M
mem_used		207.000M
swap_used		133.000M

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 237) and about available resource capacities based on the Sun Grid Engine, Enterprise Edition consumable resources facility (see the section, “Consumable Resources” on page 224).

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 50.

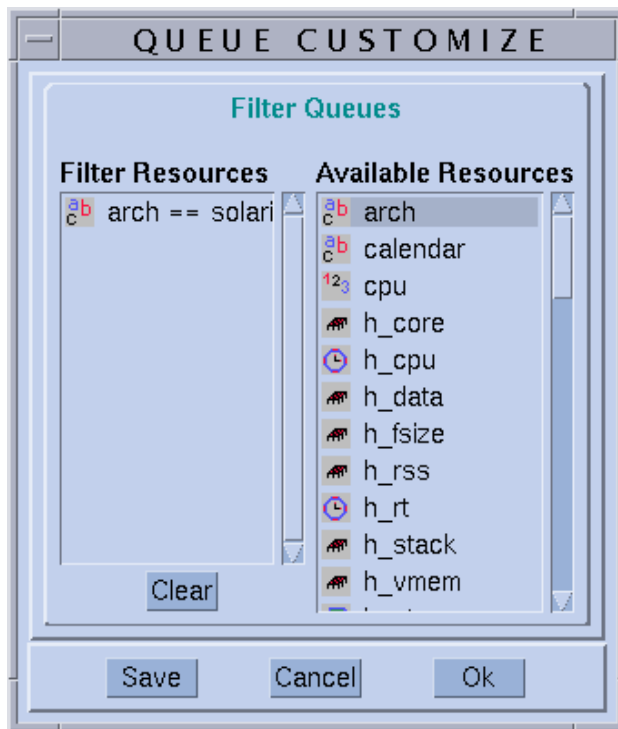


FIGURE 5-13 Queue Control Customization

▼ How To Control Queues with `qmod`

The section, “How To Control Jobs from the Command Line” on page 157 explained how the Sun Grid Engine, Enterprise Edition command, `qmod`, can be used to suspend/unsuspend Sun Grid Engine, Enterprise Edition 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, Enterprise Edition manager or operator permission (see the section, “Managers, Operators and Owners” on page 90).

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 be 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, Enterprise Edition 5.3 Administration and User's Guide* includes six chapters.

- **Chapter 6** – “Host and Cluster Configuration” on page 167

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

- **Chapter 7** – “Configuring Queues and Queue Calendars” on page 191

This chapter includes a description of the important concept of *queues*—which serve as “containers” for different categories of Sun Grid Engine, Enterprise Edition jobs. Complete instructions for configuring queues are included.

- **Chapter 8** – “The Complexes Concept” on page 213

This chapter explains how the Sun Grid Engine, Enterprise Edition 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 243

This chapter provides full background information about the types of user policies that are available through the Sun Grid Engine, Enterprise Edition system, and provides instructions on how to match these policies to the computing environment.

- **Chapter 10** – “Managing Parallel Environments” on page 313

In addition to describing how the Sun Grid Engine, Enterprise Edition system fits in with *parallel environments*, this chapter provides full configuration instructions to address them.

- **Chapter 11** – “Error Messaging” on page 327

This chapter explains the Sun Grid Engine, Enterprise Edition procedure for error message retrieval and describes how to run the software in debug mode.

Host and Cluster Configuration

This chapter provides background information about, and instructions for, configuring various aspects of the Sun Grid Engine, Enterprise Edition system. You will find instructions in this chapter for the following tasks.

- “How To Configure Administration Hosts with QMON” on page 171
- “How To Delete an Administration Host” on page 172
- “How To Add an Administration Host” on page 172
- “How To Configure Administration Hosts From the Command Line” on page 172
- “How To Configure Submit Hosts with QMON” on page 173
- “How To Delete a Submit Host” on page 174
- “How To Add a Submit Host” on page 174
- “How To Configure Submit Hosts from the Command Line” on page 174
- “How To Configure Execution Hosts with QMON” on page 175
- “How To Delete an Execution Host” on page 176
- “How To Shut Down the Execution Host Daemon” on page 176
- “How To Add or Modify an Execution Host” on page 177
- “How To Configure Execution Hosts from the Command Line” on page 181
- “How To Monitor Execution Hosts With qhost” on page 182
- “How To Kill Daemons from the Command Line” on page 183
- “How To Restart Daemons from the Command Line” on page 184
- “How To Display the Basic Cluster Configurations from the Command Line” on page 185
- “How To Modify the Basic Cluster Configurations from the Command Line” on page 185
- “How To Display a Cluster Configuration with QMON” on page 186
- “How To Delete a Cluster Configuration with QMON” on page 187
- “How To Display a Global Cluster Configuration with QMON” on page 187
- “How To Use QMON To Modify Global and Host Configurations” on page 188

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, Enterprise Edition 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, Enterprise Edition 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 requirements are met, the shadow master host facility is activated for this host. No restart of Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition 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 327) 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 183).

About Daemons and Hosts

Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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 55 describes how to initially set up the master host, and the section, “About Master and Shadow Master Configuration” on page 168 shows how dynamic master host changes can be configured. The master host usually runs the Sun Grid Engine, Enterprise Edition 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, Enterprise Edition jobs. Therefore, they are hosting Sun Grid Engine, Enterprise Edition queues and run the Sun Grid Engine, Enterprise Edition 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 56).
- **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, Enterprise Edition. 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, Enterprise Edition'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, Enterprise Edition 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, Enterprise Edition administrator.

The following sections explain how to configure the different host objects with the help of the Sun Grid Engine, Enterprise Edition 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.

Invalid Host Names

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

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

▼ 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.

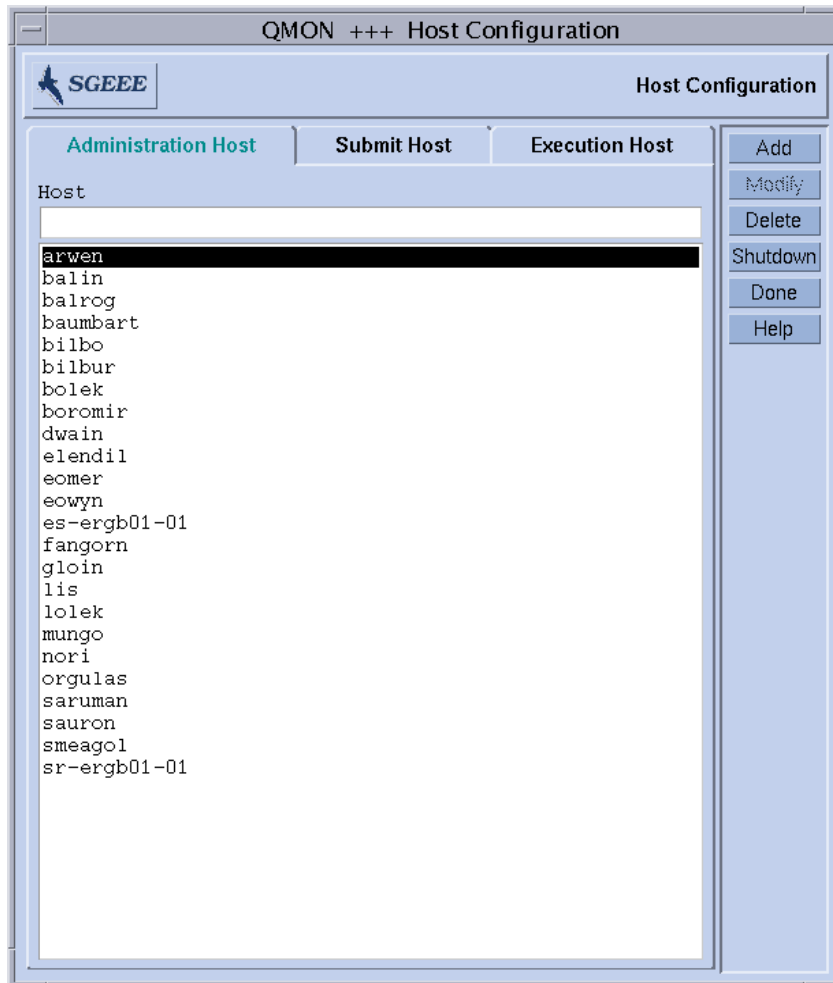


FIGURE 6-1 Administration Host Configuration Dialogue Box

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, Enterprise Edition 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.

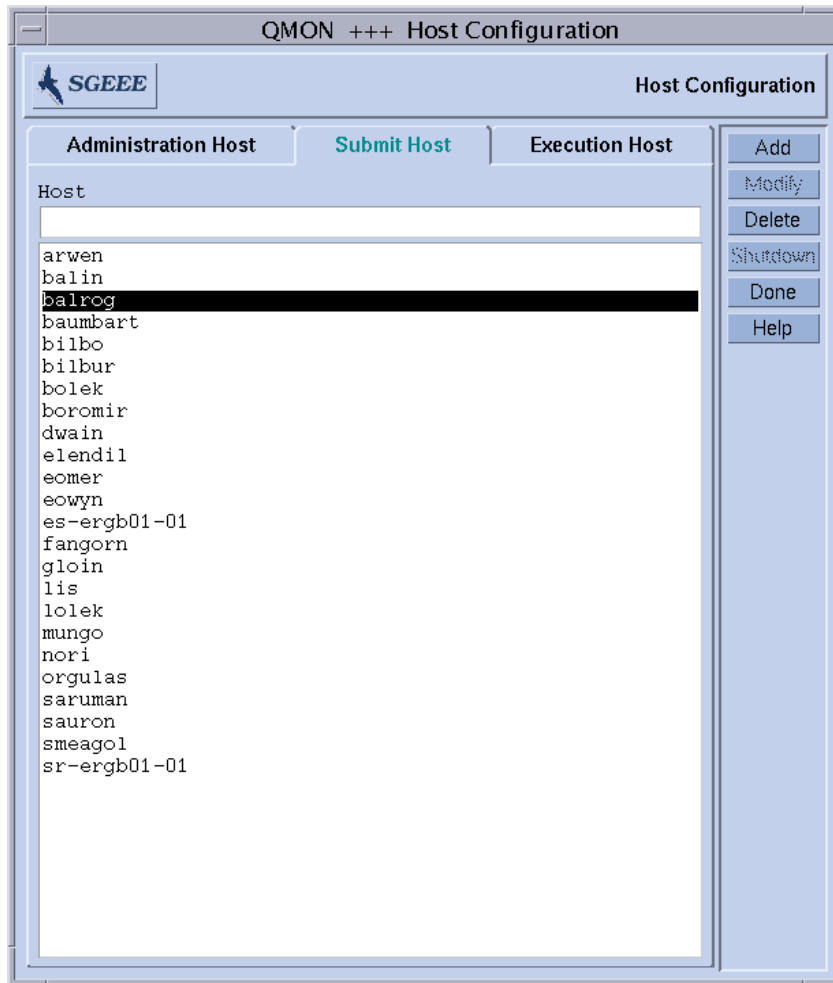


FIGURE 6-2 Submit Host Configuration

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, Enterprise Edition 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 171). 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.

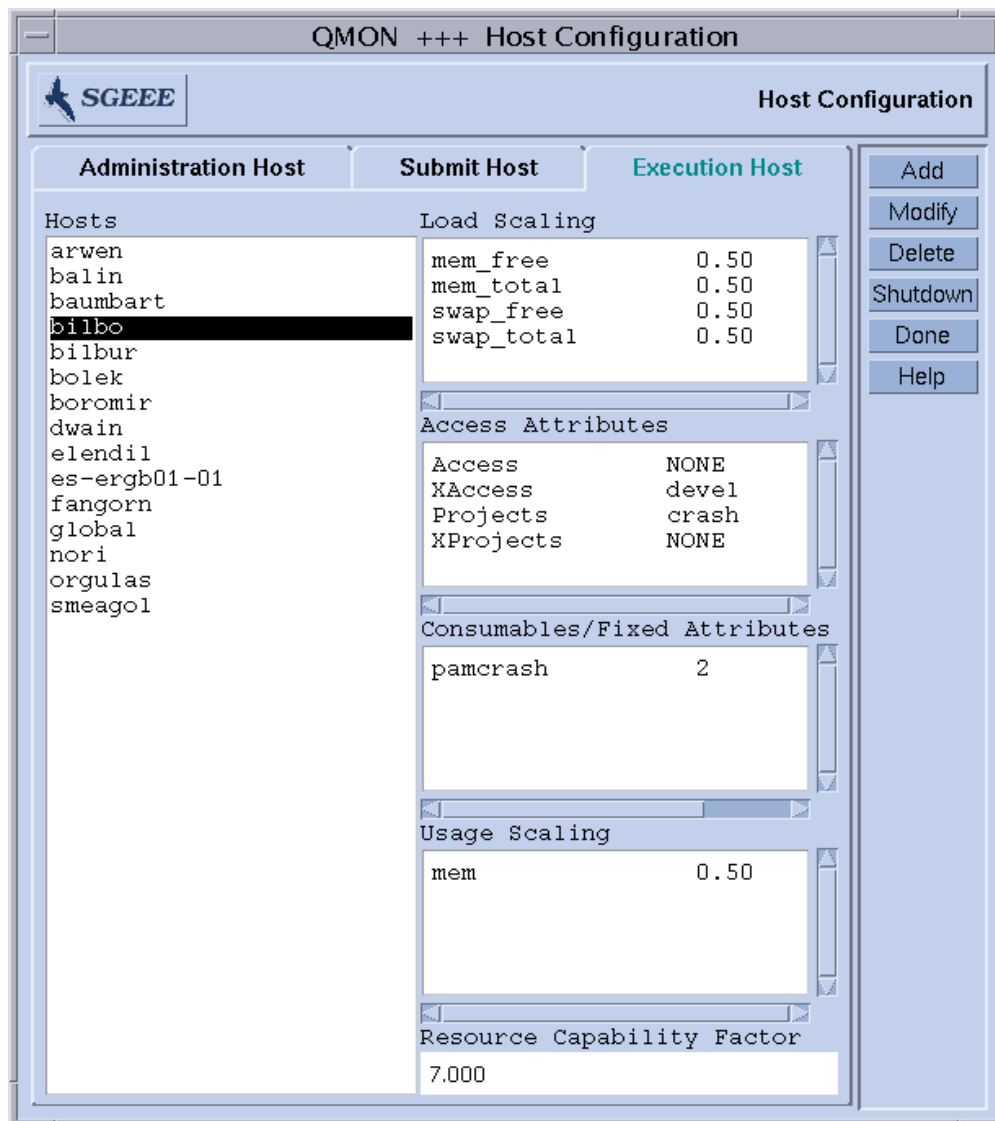


FIGURE 6-3 Execution Host Configuration

2. Depending on how you want to configure the host, proceed according to the guidance in the following sections.

Sun Grid Engine, Enterprise Edition 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 171 and “How To Configure Submit Hosts with QMON” on page 173).

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 213, “User Access Permissions” on page 88, and “Load Parameters” on page 237 for details on complex attributes, user access permissions, and load parameters.

For Sun Grid Engine, Enterprise Edition the additional Usage Scaling display window contains the current scaling factors for the individual usage metrics CPU, memory, and I/O for different machines. Resource usage is reported by `sge_execd` periodically for each job currently running. The scaling factors indicate the relative cost of resource usage on the particular machine for the user or project running a job. They could be used, for instance, to compare the cost of a second of CPU time on a 400 MHz processor to that of a 600 MHz CPU. Metrics not being displayed in the Usage Scaling window have a scaling factor of “1.”

The Resource Capability Factor field is also additional in Sun Grid Engine, Enterprise Edition and is used by the scheduler during job placement. It is a single number associated with the host which indicates its overall relative power for scheduling purposes. Factors which might contribute to the value chosen for the resource capability factor include number of CPUs, CPU clock speed, type of CPU, amount of available memory, speed of devices connected, and so forth.

▼ 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.

Add/Modify Exec Host

Host: bilbo

Ok Cancel

Scaling Consumables/Fixed Attributes User Access Project Access

Load Scaling

Load	Scale Factor
logins	1.00
mem_free	0.50
mem_total	0.50
mem_used	1.00
nastran	1.00
np_load_avg	1.00
np_load_long	1.00
np_load_medium	1.00
np_load_short	1.00

Usage Scaling

Usage	Scale Factor
cpu	1.00
mem	0.50
io	1.00

Resource Capability Factor

7

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.

For Sun Grid Engine, Enterprise Edition, the current scaling factors for the usage metrics CPU, memory and I/O are displayed in the Usage column of the Usage 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.

In addition, a resource capability factor can be assigned to the host in the Resource Capability Factor input field for Sun Grid Engine, Enterprise Edition. Valid factors are again positive floating point numbers in fixed point or scientific notation.

If 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 213) 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.

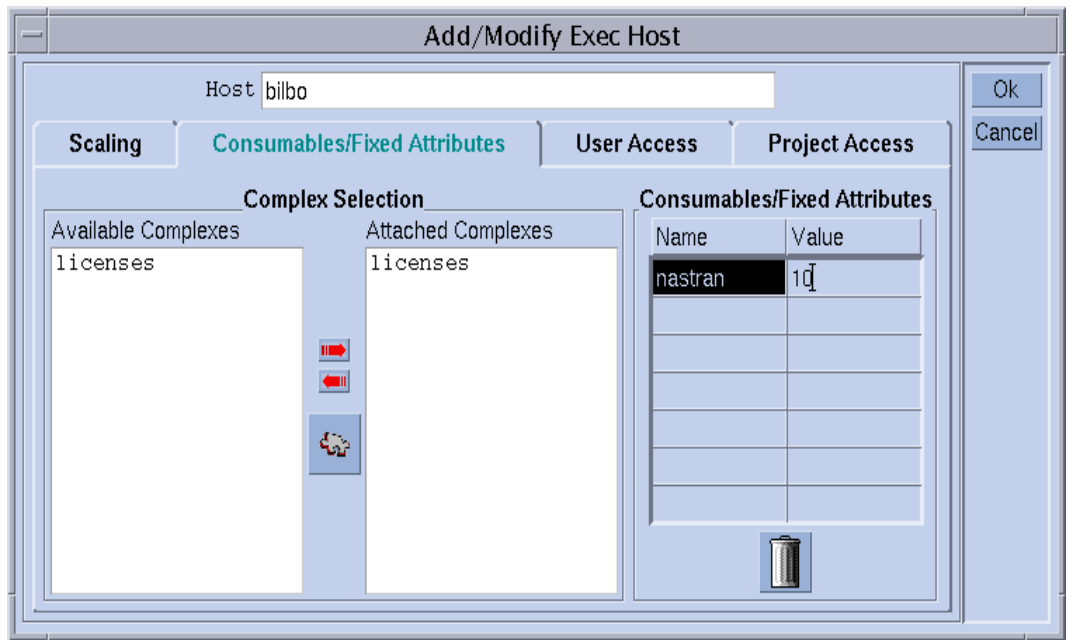


FIGURE 6-5 Modify Consumable/Fixed Attributes

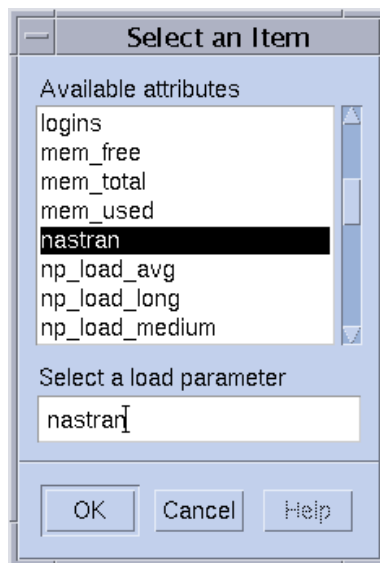


FIGURE 6-6 Available Complex Attributes

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.

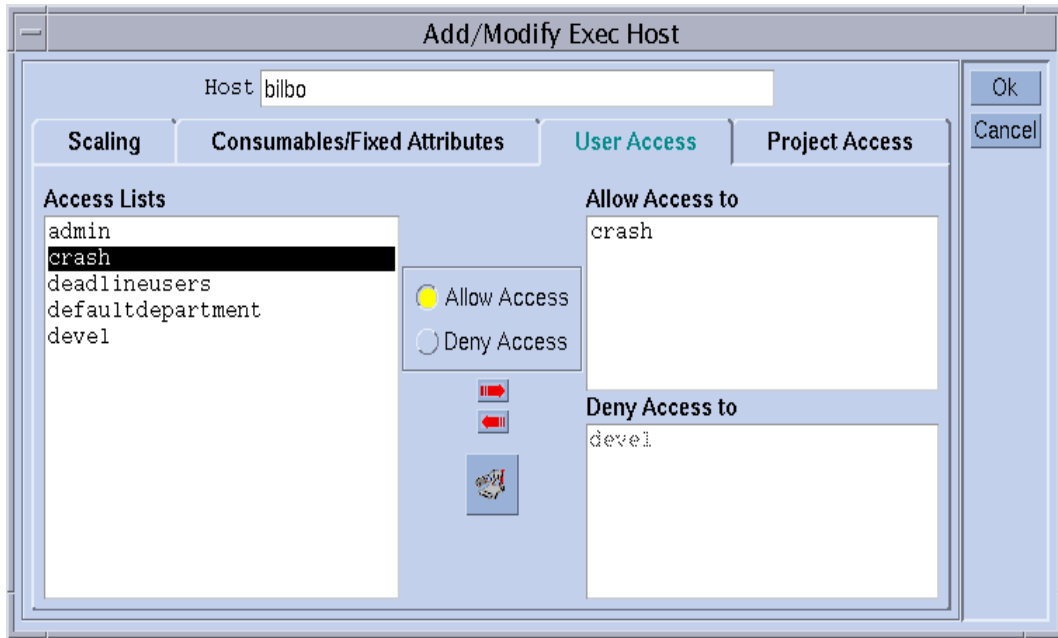


FIGURE 6-7 Modify User Access

By selecting the Project Access tab (FIGURE 6-8), you can define access permissions to the execution host based on previously configured projects.

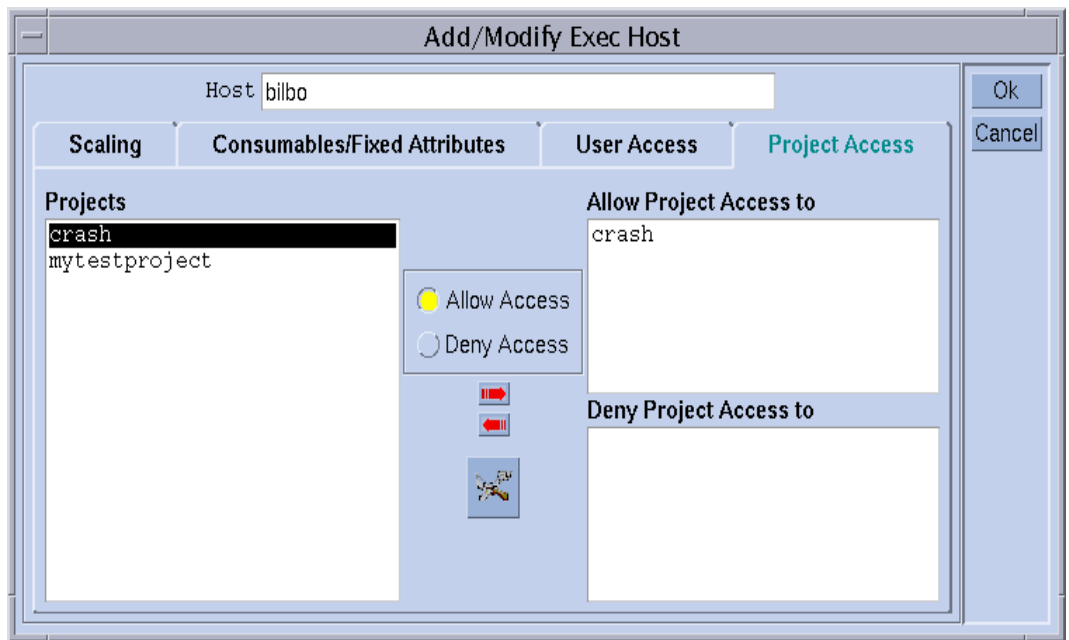


FIGURE 6-8 Modify Project Access

▼ 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 `ghost`

The `ghost` command provides a convenient way to retrieve a quick overview of the execution host status.

- Enter the following command.

```
% ghost
```

The command produces output similar to the following.

TABLE 6-1 Sample qghost 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 qghost 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, Enterprise Edition manager or operator privileges for these operations (see Chapter 9, “Managing User Access and Policies” on page 243).

```
% qconf -kej
% qconf -ks
% qconf -km
```

- The first command will kill all currently active jobs and bring down all Sun Grid Engine, Enterprise Edition execution daemons.

Note – If replacing that command by `qconf -ke`, the Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition behavior. There is a global configuration, which is provided by for the Sun Grid Engine, Enterprise Edition master host as well as every host in the Sun Grid Engine, Enterprise Edition pool. In addition, the Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition command—`qconf`—to change the cluster configurations may be used by Sun Grid Engine, Enterprise Edition 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 6-9, is displayed.

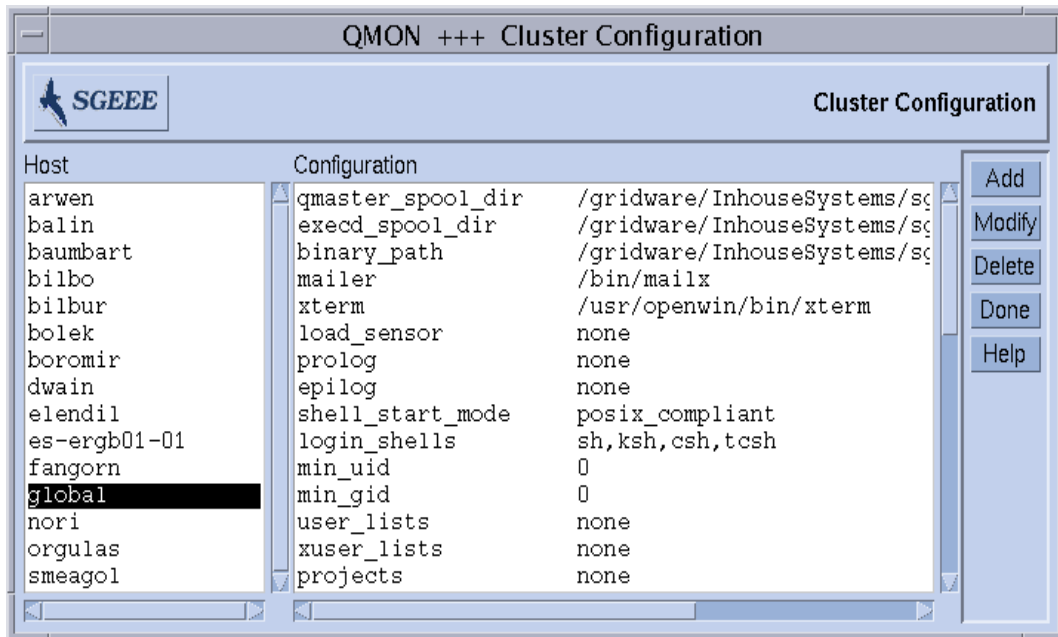


FIGURE 6-9 Cluster Configuration Dialog Box

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, `global`.

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 186), click either the Add button or the Modify button.

The Cluster Settings dialogue box, similar to the example in FIGURE 6-10, is opened.

Cluster Settings
Configuration for Host

global

General Settings

Master Spool Dir: /gridware/inhouseSystems/sgeee
Execd Spool Dir: /gridware/inhouseSystems/sgeee
Binary Path: /gridware/inhouseSystems/sgeee
Mailer: /bin/mailx
Xterm: /usr/openwin/bin/xterm
Load Sensor: none
Admin User: cccadmin
Admin Mail: none
Prolog: none
Epilog: none
Login Shells: sh,ksh,csh,tcsh
Maximal Array Job Instances: 2000
Maximal Array Job Tasks: 75000
Maximal Jobs Per User: 10
Shell Start Mode: posix_compliant
Log Level: log_info

Advanced Settings

Min UID: 0
Min GID: 0
Finished Jobs: 0
Loadreport Time: 00:00:15
Max Unheard: 00:05:00
Stat Log Time: 48:00:00
Reschedule Unknown: 00:00:00
User Lists:
Xuser Lists:
Projects:
XProjects:
Ignore FQDN: True
Enforce Project: False
Enforce User: False
GID Range for Job Monitoring: 33801-34000

Ok
Cancel

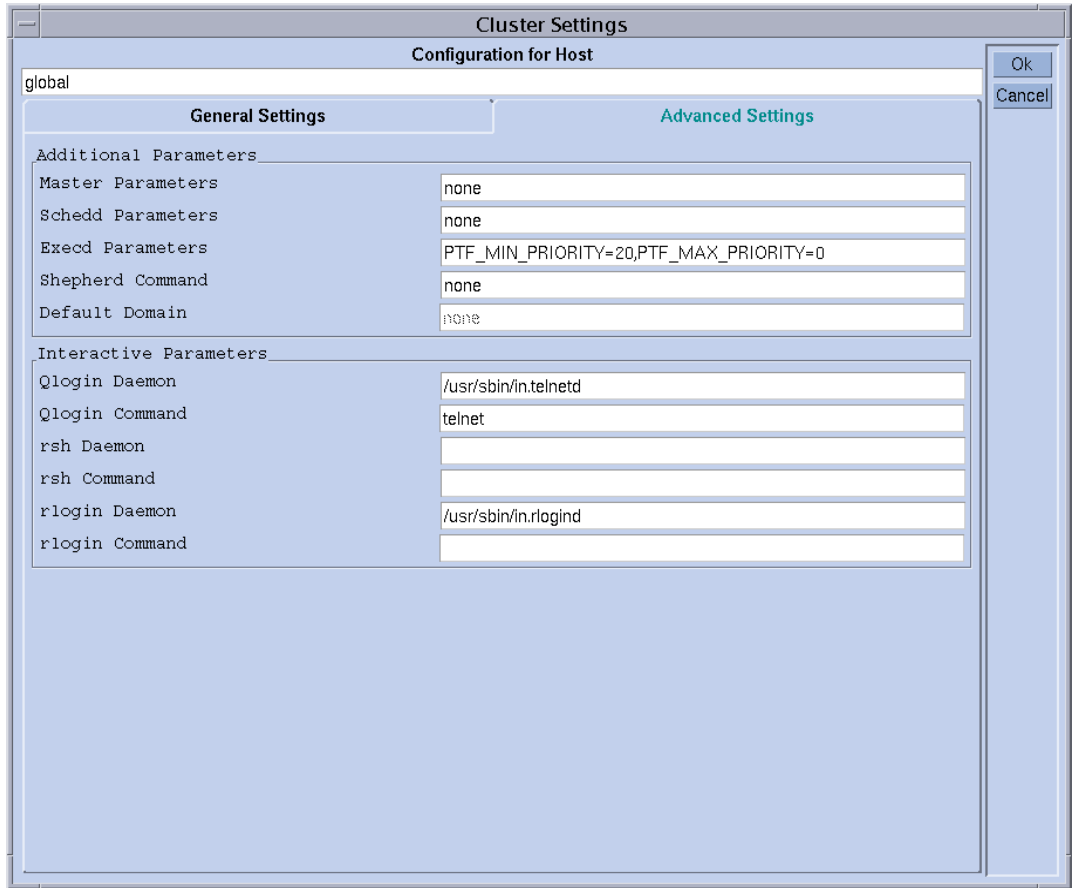
FIGURE 6-10 Cluster Settings Dialogue Box—General Settings

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 6-11) 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.



The image shows a window titled "Cluster Settings" with a sub-header "Configuration for Host". Below this is a dropdown menu currently set to "global". To the right of the dropdown are "Ok" and "Cancel" buttons. The main area is divided into two tabs: "General Settings" (selected) and "Advanced Settings". Under "General Settings", there is a section "Additional Parameters" with five rows: "Master Parameters" (none), "Schedd Parameters" (none), "Execd Parameters" (PTF_MIN_PRIORITY=20,PTF_MAX_PRIORITY=0), "Shepherd Command" (none), and "Default Domain" (none). Below this is a section "Interactive Parameters" with five rows: "Qlogin Daemon" (/usr/sbin/in.telnetd), "Qlogin Command" (telnet), "rsh Daemon" (empty), "rsh Command" (empty), "rlogin Daemon" (/usr/sbin/in.rlogind), and "rlogin Command" (empty).

Additional Parameters	
Master Parameters	none
Schedd Parameters	none
Execd Parameters	PTF_MIN_PRIORITY=20,PTF_MAX_PRIORITY=0
Shepherd Command	none
Default Domain	none

Interactive Parameters	
Qlogin Daemon	/usr/sbin/in.telnetd
Qlogin Command	telnet
rsh Daemon	
rsh Command	
rlogin Daemon	/usr/sbin/in.rlogind
rlogin Command	

FIGURE 6-11 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.

Configuring Queues and Queue Calendars

This chapter provides background information about, and instructions for, configuring Sun Grid Engine, Enterprise Edition queues and queue calendars.

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

- “How To Configure Queues with QMON” on page 192
- “How To Configure General Parameters” on page 193
- “How To Configure Execution Method Parameters” on page 195
- “How To Configure Checkpointing Parameters” on page 196
- “How To Configure Load and Suspend Thresholds” on page 197
- “How To Configure Limits” on page 198
- “How To Configure User Complexes” on page 200
- “How To Configure Subordinate Queues” on page 202
- “How To Configure User Access” on page 203
- “How To Configure Project Access” on page 204
- “How To Configure Owners” on page 205
- “How To Configure Queues from the Command Line” on page 206
- “How To Configure Queue Calendars With QMON” on page 207
- “How To Configure Calendars From the Command Line” on page 210

About Configuring Queues

Sun Grid Engine, Enterprise Edition *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,

Enterprise Edition queues, but start running immediately as soon as they are dispatched. The Sun Grid Engine, Enterprise Edition scheduler's job pending list is the only waiting area for Sun Grid Engine, Enterprise Edition jobs.

Configuring Sun Grid Engine, Enterprise Edition 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, Enterprise Edition users on all hosts belonging to the Sun Grid Engine, Enterprise Edition 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 158. 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 193).

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 200 and "How To Configure User Access" on page 203 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.

Ten parameter sets are available to define a queue.

- General (see "How To Configure General Parameters" on page 193)
- Execution Method (see "How To Configure Execution Method Parameters" on page 195)

- Checkpointing (see “How To Configure Checkpointing Parameters” on page 196)
- Load/Suspend Thresholds (see “How To Configure Load and Suspend Thresholds” on page 197)
- Limits (see “How To Configure Limits” on page 198)
- Complexes (see “How To Configure User Complexes” on page 200)
- Subordinates (see “How To Configure Subordinate Queues” on page 202)
- User Access (see “How To Configure User Access” on page 203)
- Project Access (see “How To Configure Project Access” on page 204)
- Owners (see “How To Configure Owners” on page 205)

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.

The screenshot shows the 'Queue Configuration: Modify' dialog box. At the top, there's a title bar and a logo. Below the logo, the text 'Queue Configuration: Modify' is displayed. The main area contains several input fields and buttons. The 'Queue' field is set to 'bilbur.q' and the 'Hostname' field is set to 'bilbur'. There are buttons for 'Clone', 'Reset', and 'Refresh'. On the right side, there are buttons for 'Ok', 'Cancel', and 'Help'. Below these fields, there are five tabs: 'Complexes', 'Subordinates', 'User Access', 'Project Access', and 'Owners'. The 'General Configuration' tab is selected, showing various parameters. The 'Sequence Nr' is 0, 'Processors' is 'UNDEFINED', 'tmp Directory' is '/tmp', 'Shell' is '/bin/csh', 'Shell Start Mode' is 'posix_compliant', 'Initial State' is 'default', 'Rerun Jobs' is a checkbox, 'Calendar' is 'weekend-night', 'Notify Time' is '00:00:60', 'Job's Nice' is 10, 'Slots' is 10, and 'Type' is a list with checkboxes for 'Batch', 'Interactive', 'Checkpointing', 'Parallel', and 'Transfer'.

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, Enterprise Edition 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 `sgc_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 4-9).

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.

The screenshot shows a window titled "Queue Configuration: Modify". Inside, there's a header bar with the SGE logo and the title. Below the header, there are input fields for "Queue" (containing "bilbur.q") and "Hostname" (containing "bilbur"). To the right of these fields are "Ok", "Cancel", and "Help" buttons. Below the input fields are three buttons: "Clone", "Reset", and "Refresh". The main area of the window is divided into several tabs: "Complexes", "Subordinates", "User Access", "Project Access", "Owners", "General Configuration", "Execution Method" (which is selected and highlighted in green), "Checkpointing", "Load/Suspend Thresholds", and "Limits". Under the "Execution Method" tab, there are six rows of text labels followed by empty input fields: "Prolog", "Epilog", "Starter Method", "Suspend Method", "Resume Method", and "Terminate Method".

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, Enterprise Edition'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.

The screenshot shows a window titled "Queue Configuration: Modify". At the top left is the SGE logo and the text "SGEEE". Below this, the title "Queue Configuration: Modify" is repeated. The main area is divided into two sections. The top section contains input fields for "Queue" (with the value "bilbur.q") and "Hostname" (with the value "bilbur"). To the right of these fields are three buttons: "Ok", "Cancel", and "Help". Below the input fields are three buttons: "Clone", "Reset", and "Refresh". The bottom section is a tabbed interface with five tabs: "Complexes", "Subordinates", "User Access", "Project Access", and "Owners". The "User Access" tab is selected and highlighted in green. Within this tab, there are five sub-tabs: "General Configuration", "Execution Method", "Checkpointing", "Load/Suspend Thresholds", and "Limits". The "Checkpointing" sub-tab is selected and highlighted in green. Below the sub-tabs, there is a label "MinCpuTime" and a text input field containing the value "00:05:00". To the right of the input field is a small icon of a power button.

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 this parameter.

▼ 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.

Queue Configuration: Modify

Queue: bilbur.q

Hostname: bilbur

Clone Reset Refresh

Complexes Subordinates User Access Project Access Owners

General Configuration Execution Method Checkpointing **Load/Suspend Thresholds** Limits

Load Thresholds

Load	Value
np_load_av	11.00
h_vmem	100M

Suspend Thresholds

Load	Value
np_load_av	40.0

Suspend interval: 00:05:00

Jobs suspended per interval: 1

Ok Cancel Help

FIGURE 7-4 Queue Configuration—Load/Suspend 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 213).

Overload in the case of load thresholds results in preventing the queue from receiving further jobs by Sun Grid Engine, Enterprise Edition. 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.

Queue Configuration: Modify

Queue: bilbur.q

Hostname: bilbur

Clone Reset Refresh

Complexes Subordinates User Access Project Access Owners

General Configuration Execution Method Checkpointing Load/Suspend Thresholds **Limits**

Double click on values to change limits

Hard Limit	Value
Wallclock Time (sec)	24:00:00
CPU Time (sec)	12:00:00
File Size (Byte)	INFINITY
Data Size (Byte)	INFINITY
Stack Size (Byte)	INFINITY
Corefile Size (Byte)	INFINITY
Resident Set Size (Byte)	500M

Soft Limit	Value
Wallclock Time (sec)	INFINITY
CPU Time (sec)	08:00:00
File Size (Byte)	INFINITY
Data Size (Byte)	INFINITY
Stack Size (Byte)	INFINITY
Corefile Size (Byte)	INFINITY
Resident Set Size (Byte)	INFINITY

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).

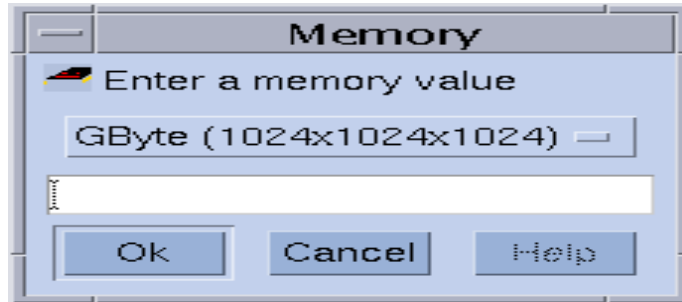


FIGURE 7-6 Memory Input Dialogue Box

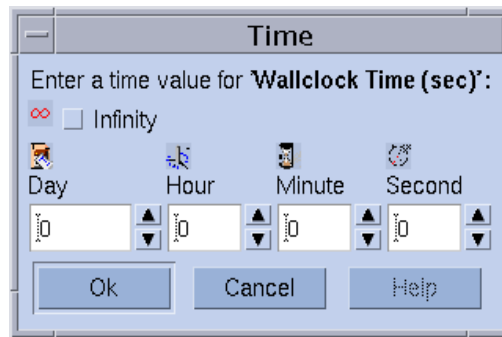


FIGURE 7-7 Time Input Dialogue Box

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.

Queue Configuration: Modify

Queue: bilbur.q
Hostname: bilbur

Clone Reset Refresh

General Configuration Execution Method Checkpointing Load/Suspend Thresholds Limits

Complexes Subordinates User Access Project Access Owners

Complex Selection

Available Complexes: licenses
Attached Complexes: licenses

Consumables/Fixed Attributes

Load	Value
nastran	5

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 220) 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 213 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 213 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.

Queue Configuration: Modify

Queue: bilbur.q

Hostname: bilbur

Clone Reset Refresh

Ok Cancel Help

General Configuration Execution Method Checkpointing Load/Suspend Thresholds Limits

Complexes Subordinates User Access Project Access Owners

Queue	Max Slots
arwen.q	2

If 'Max Slots' are filled in the current queue, the queues in column one are suspended.

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.

Refer 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.

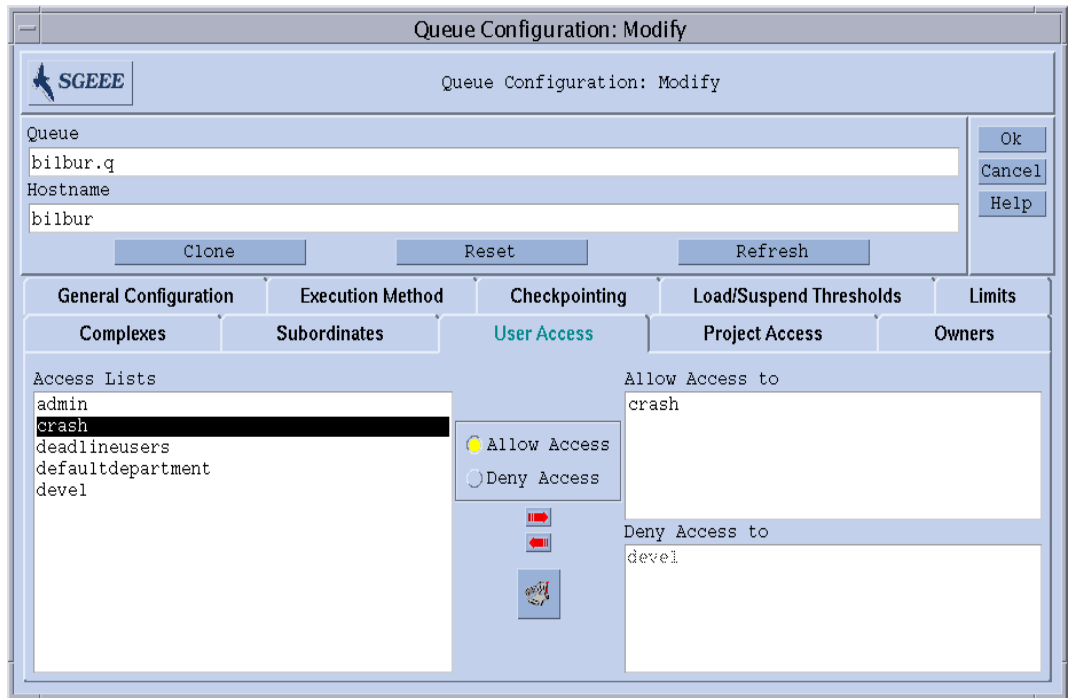


FIGURE 7-10 Queue Configuration—User Access

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 88) by clicking the button in the middle bottom of the screen.

▼ How To Configure Project Access

- Select the Project Access parameter set.

A screen similar to the example in FIGURE 7-11 is displayed.

Queue Configuration: Modify

SGEEE Queue Configuration: Modify

Queue: bilbur.q

Hostname: bilbur

Clone Reset Refresh

General Configuration Execution Method Checkpointing Load/Suspend Thresholds Limits

Complexes Subordinates User Access **Project Access** Owners

Projects: crash, devel

Allow Project Access to: crash

Deny Project Access to:

Allow Access (selected) Deny Access

FIGURE 7-11 Queue Configuration—Project Access

The fields offered allow for setting the following parameters:

- The projects allowed or denied access to the queue

Jobs being submitted to a project belonging to the list of allowed projects have access to the queue. Jobs submitted to denied projects will not be dispatched to the queue.

Refer to the `queue_conf` manual page for detailed information on these parameters.

Open the Project Configuration dialogue box (see “About Projects” on page 258) 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 7-12 is displayed.

The screenshot shows a window titled "Queue Configuration: Modify" with the SGEEE logo. It contains fields for "Queue" (bilbur.q) and "Hostname" (bilbur). Below these are buttons for "Clone", "Reset", and "Refresh". On the right are "Ok", "Cancel", and "Help" buttons. The main area has tabs for "General Configuration", "Execution Method", "Checkpointing", "Load/Suspend Thresholds", and "Limits". The "Limits" tab is active, showing sub-tabs for "Complexes", "Subordinates", "User Access", "Project Access", and "Owners". The "Owners" sub-tab is selected, displaying an "Owner List" with "charu" and an "Enter Owner" field. A trash bin icon is located in the bottom right of the "Owner List" area.

FIGURE 7-12 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 a 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, Enterprise Edition 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, Enterprise Edition 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 Configuration.

The Queue Calendar Configuration dialogue box, similar to FIGURE 7-13, is displayed.

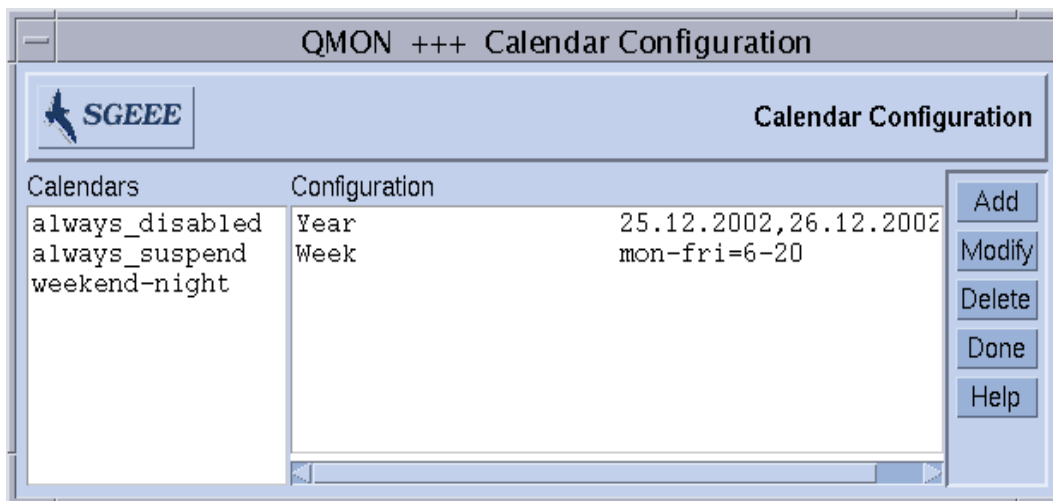


FIGURE 7-13 Calendar Configuration

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 7-14, is opened and provides the means to delete, modify, or add.

Add/Modify Calendar

Calendar Name
weekend-night

Example

Year
1.1.1997,30.3.1997-31.3.1997,26.12.1997[=suspended]

Week
mon-fri=20-6[=suspended] sat-sun[=suspended]

Year
25.12.2002,26.12.2002=on

Week
mon-fri=6-20

Ok
Cancel

FIGURE 7-14 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 7-15. 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 the section, “About Configuring Queues” on page 191 for further details on configuring queues.

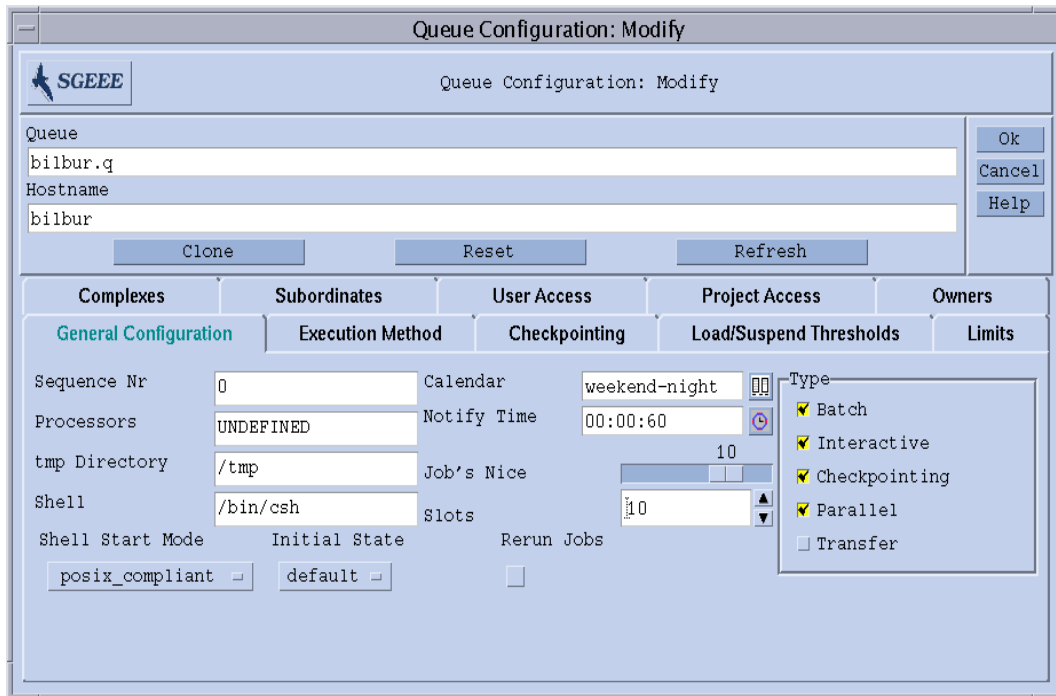


FIGURE 7-15 Calendar Configuration in General Parameters Queue Configuration Menu

▼ 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, Enterprise Edition cluster. The calendar to be added is either read from file (`-Acal`), 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`).

The Complexes Concept

This chapter explains the important Sun Grid Engine, Enterprise Edition 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.

- “How To Add Or Modify a Complex Configuration” on page 214
- “How To Set Up Consumable Resources” on page 224
- “How To Modify Complex Configurations From the Command Line” on page 235
- “How to Write Your Own Load Sensors” on page 238

About Complexes

The definition of complexes provides all pertinent information concerning the resource attributes a user may request for a Sun Grid Engine, Enterprise Edition job via the `qsub` or `qalter -l` option and for the interpretation of these parameters within the Sun Grid Engine, Enterprise Edition system.

Complexes also build the framework for Sun Grid Engine, Enterprise Edition 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, Enterprise Edition jobs will be taken into account during the scheduling process. Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition 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 216
 - "The Host Complex" on page 217
 - "The Global Complex" on page 219
 - "User-Defined Complexes" on page 220

The Complex Configuration dialogue box provides the means for changing the definition of the existing complexes and for defining new user complexes.

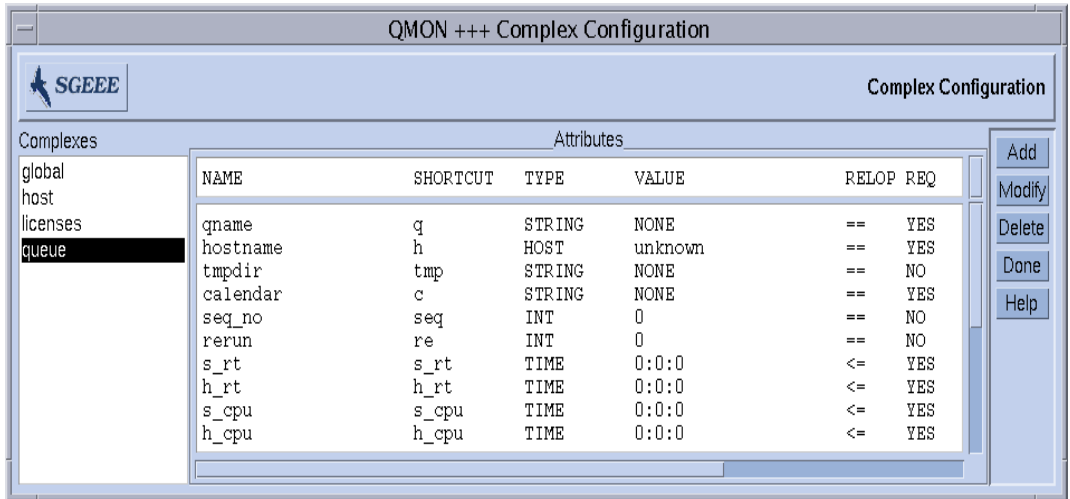


FIGURE 8-1 Complex Configuration Dialogue Box—Queue

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.

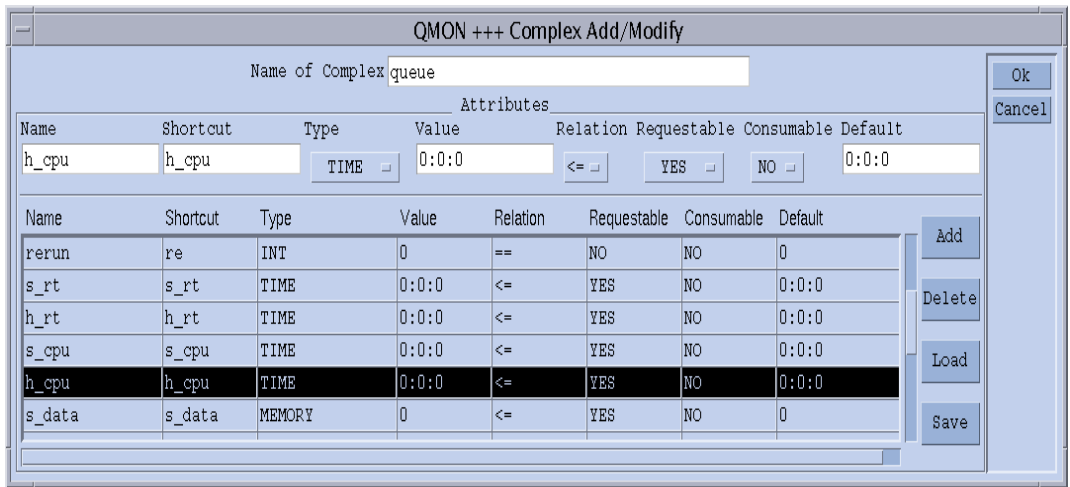


FIGURE 8-2 Complex Add/Modify Dialogue Box

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 `sgc_qmaster`.

Complex Types

The Sun Grid Engine, Enterprise Edition 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 191). This is usually done for *Consumable Resources* (see the section, “Consumable Resources” on page 224). 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`
- `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, Enterprise Edition load values such as the CPU load average or load values defined by the Sun Grid Engine, Enterprise Edition administration (see the section, “Load Parameters” on page 237).

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 170)

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 224), 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.

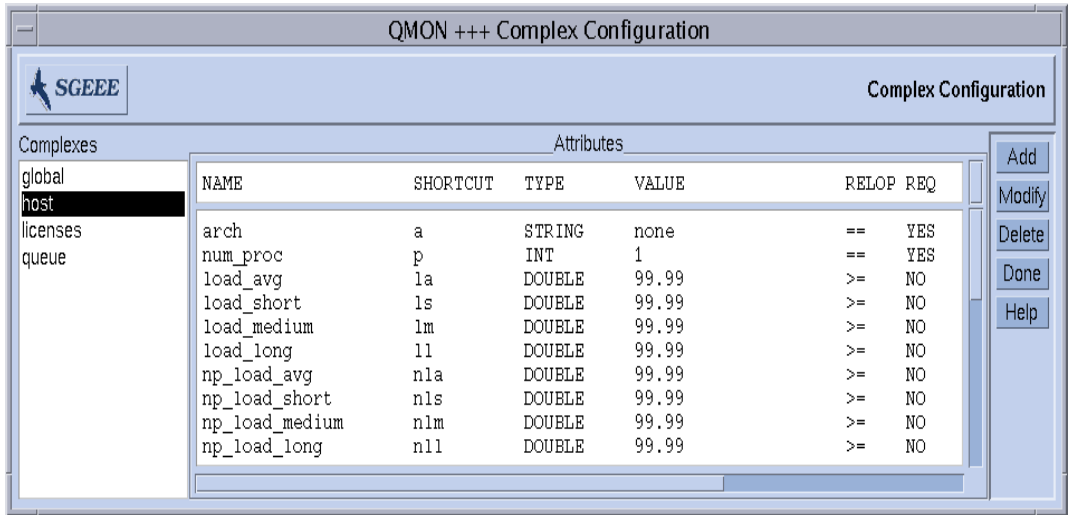


FIGURE 8-3 Complex Configuration Dialogue Box—Host

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 237). Global load values can be reported from any host in the cluster. There are no global load values reported by Sun Grid Engine, Enterprise Edition 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 170) 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.

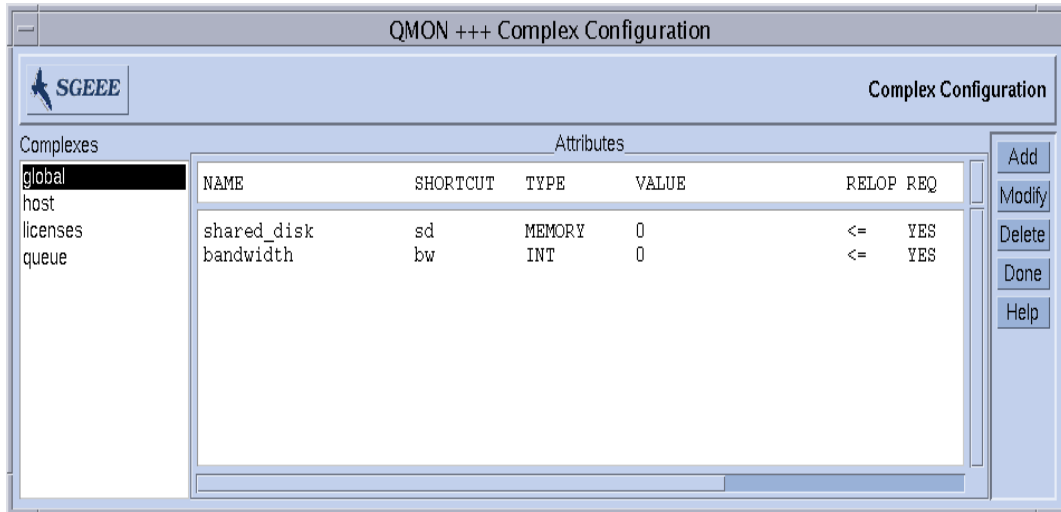


FIGURE 8-4 Complex Configuration Dialogue Box—Global

User-Defined Complexes

By setting up user-defined complexes, the Sun Grid Engine, Enterprise Edition administration has the ability to extend the set of attributes managed by Sun Grid Engine, Enterprise Edition 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, Enterprise Edition. 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 191 and “About Configuring Hosts” on page 170). 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.

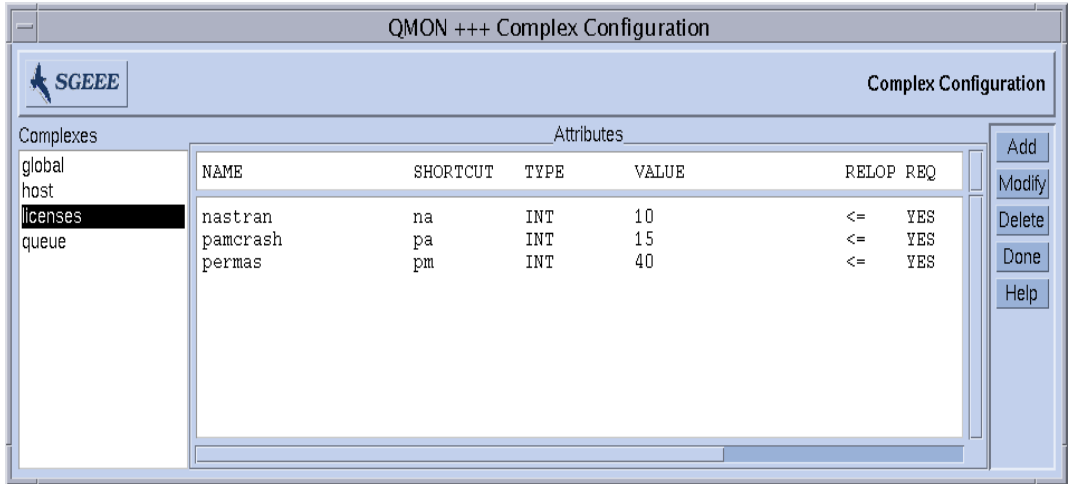


FIGURE 8-5 Complex Configuration Dialogue Box—Licenses

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 191 and its related sections for details on how to configure queues).

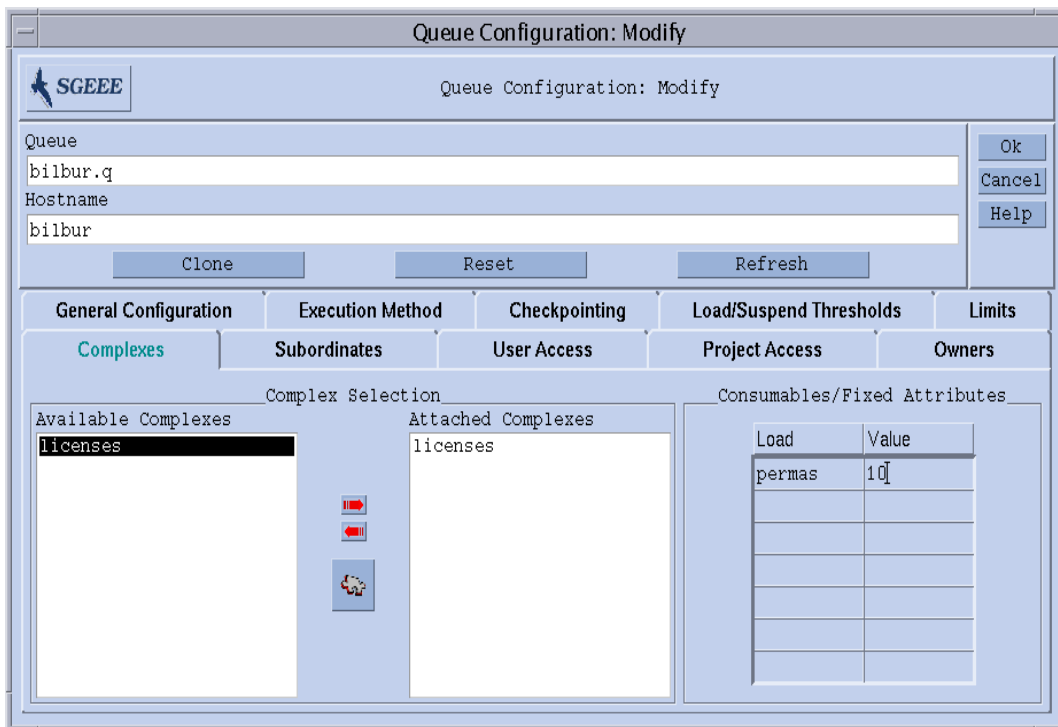


FIGURE 8-6 User-Defined Complexes Queue Configuration

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, Enterprise Edition 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, “Submitting Jobs” on page 91 for details on how to submit jobs).

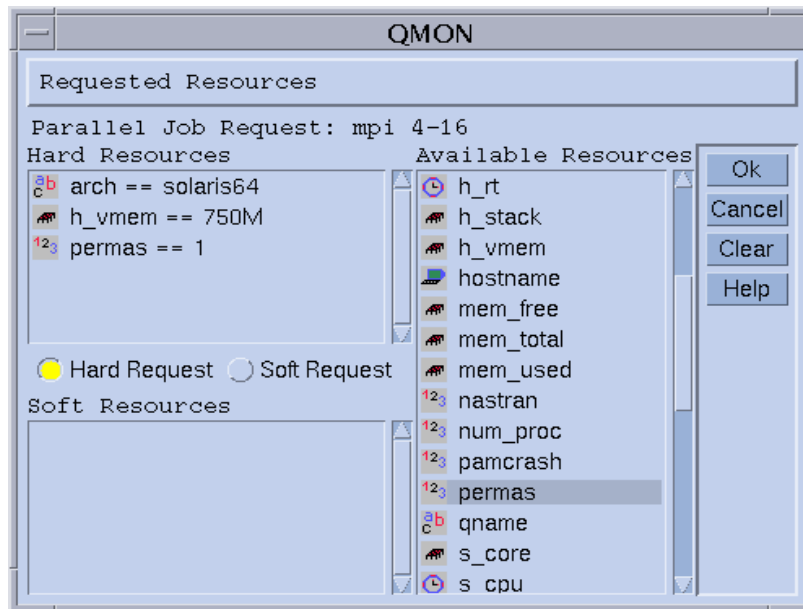


FIGURE 8-7 Requested Resources Submit Sub-Dialogue Box

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, Enterprise Edition administrator and the consumption of the corresponding resource is monitored by Sun Grid Engine, Enterprise Edition internal bookkeeping. Sun Grid Engine, Enterprise Edition accounts for the consumption of this resource for all running jobs and ensures that jobs are only dispatched if the Sun Grid Engine, Enterprise Edition internal bookkeeping indicates enough available consumable resources.

Consumables can be combined with default or user-defined load parameters (see “Load Parameters” on page 237); 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, Enterprise Edition 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 191 and “About Configuring Hosts” on page 170). 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, Enterprise Edition 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 226
 - “Example 2: Space Sharing for Virtual Memory” on page 230
 - “Example 3: Managing Available Disk Space” on page 233

QMON +++ Complex Add/Modify

Name of Complex:

Attributes

Name	Shortcut	Type	Value	Relation	Requestable	Consumable	Default
virtual_free	vf	MEMORY	0	<=	YES	YES	1G

Name	Shortcut	Type	Value	Relation	Requestable	Consumable	Default
swap_free	sf	MEMORY	0	<=	YES	NO	0
swap_total	st	MEMORY	0	<=	YES	NO	0
virtual_free	vf	MEMORY	0	<=	YES	NO	0
virtual_total	vt	MEMORY	0	<=	YES	NO	0
mem_used	mu	MEMORY	INFINITY	>=	YES	NO	0
swap_used	su	MEMORY	INFINITY	>=	YES	NO	0

Buttons: Add, Delete, Load, Save, Ok, Cancel

FIGURE 8-8 Complex Configuration Dialogue Box—`virtual_free`

Then, for each queue or for each host you want Sun Grid Engine, Enterprise Edition 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.

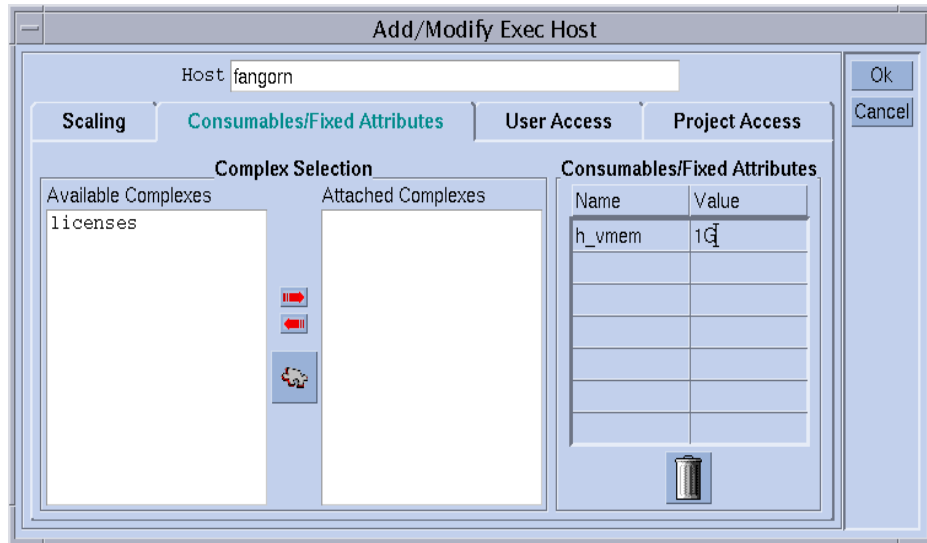


FIGURE 8-9 Execution Host Configuration—virtual_free

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, Enterprise Edition 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, Enterprise Edition 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.

QMON +++ Complex Add/Modify

Name of Complex: global

Name	Shortcut	Type	Value	Relation	Requestable	Consumable	Default
pam-crash	pc	INT	0	<=	FORCED	YES	1
shared_disk	sd	MEMORY	0	<=	YES	YES	100M
bandwidth	bw	INT	0	<=	YES	YES	10M
pam-crash	pc	INT	0	<=	FORCED	YES	1

Buttons: Add, Delete, Load, Save, Ok, Cancel

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.

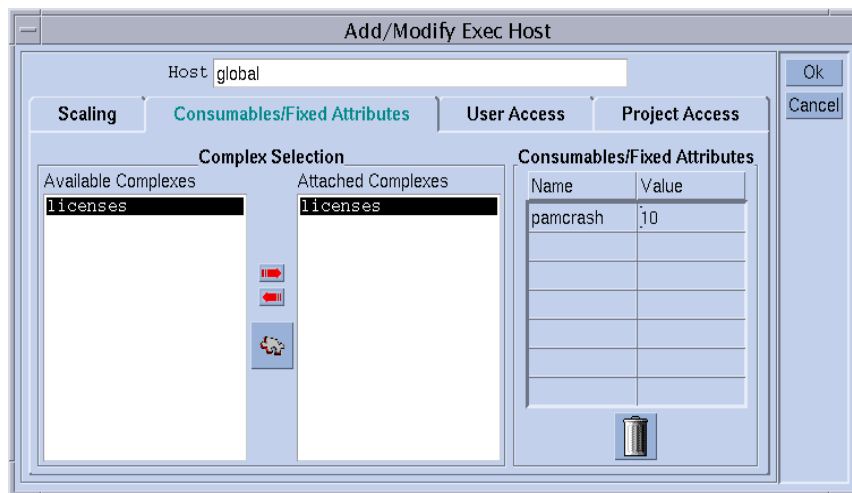


FIGURE 8-11 Global Host Configuration—pam-crash

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 the host in FIGURE 8-12.

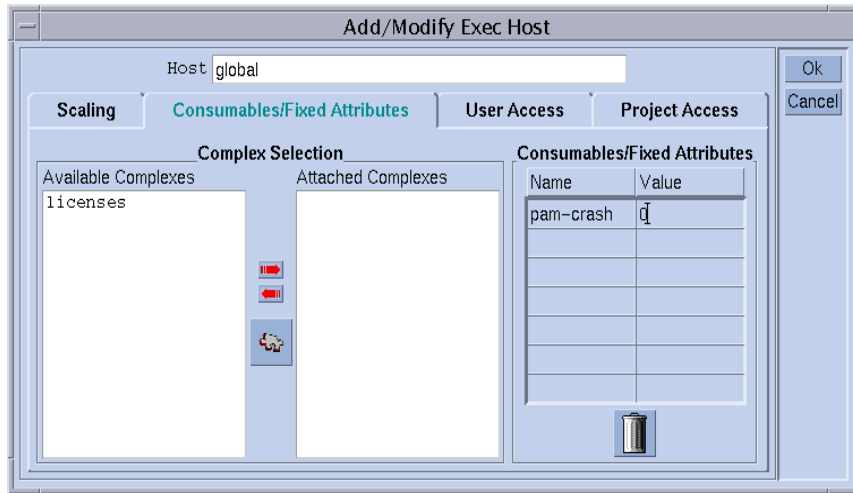


FIGURE 8-12 Execution Host Configuration—pam-crash

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.

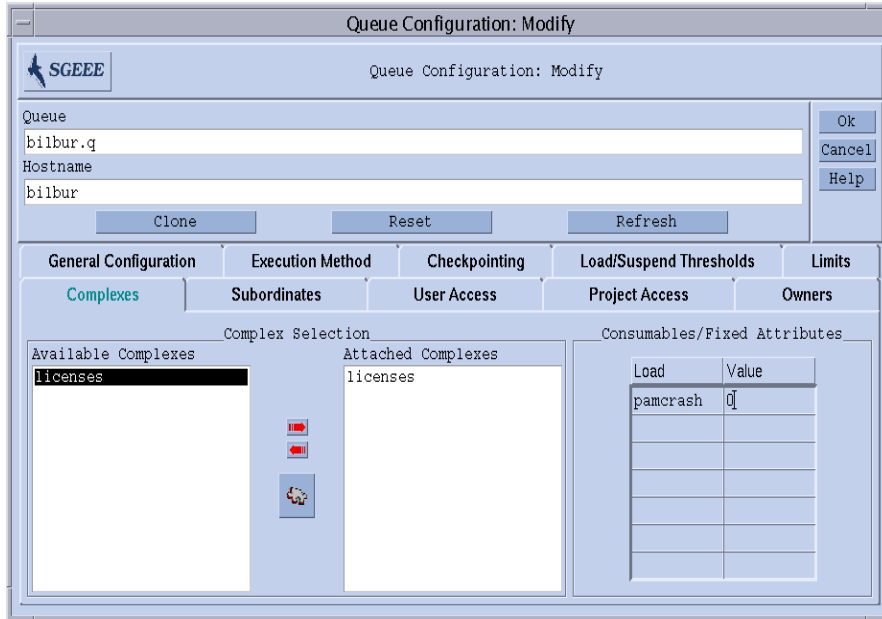


FIGURE 8-13 Queue Configuration—pam-crash

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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition. The additional availability of recent memory statistics will be taken into account automatically by Sun Grid Engine, Enterprise Edition in the virtual memory capacity planning. If the load report for free virtual memory falls below the value obtained by Sun Grid Engine, Enterprise Edition-internal bookkeeping, the load value will be used to avoid memory oversubscription. Differences in the reported load values and the Sun Grid Engine, Enterprise Edition internal bookkeeping may occur easily if jobs are started without using Sun Grid Engine, Enterprise Edition.

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 `bilbur`—500 Megabytes, to the queue `bilbur.q`. Hence the accumulated memory consumption of all jobs executing in queue `bilbur.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 `bilbur` may still not exceed 1 Gigabyte.

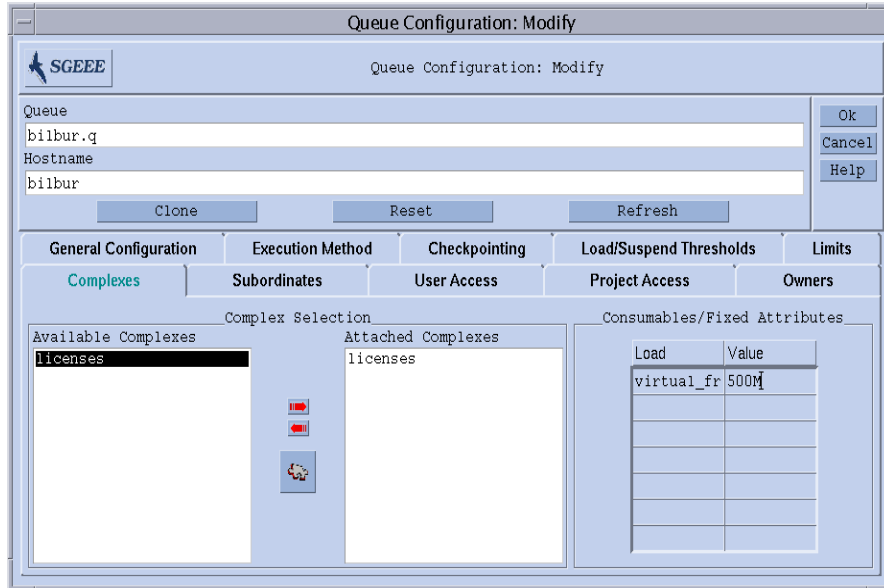


FIGURE 8-14 Queue Configuration—virtual_free

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 `bilbur.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, Enterprise Edition 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, Enterprise Edition 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 consumable resource, as shown, for example, in FIGURE 8-15.

QMON +++ Complex Add/Modify

Name of Complex:

Attributes

Name	Shortcut	Type	Value	Relation	Requestable	Consumable	Default
h_fsize	h_fsize	MEMORY	0	<=	YES	NO	0
slots	s	INT	0	<=	YES	YES	1
s_vmem	s_vmem	MEMORY	0	<=	YES	NO	0
h_vmem	h_vmem	MEMORY	0	<=	YES	NO	0
s_fsize	s_fsize	MEMORY	0	<=	YES	NO	0
h_fsize	h_fsize	MEMORY	0	<=	YES	NO	0
cpu	cpu	DOUBLE	0	>=	YES	NO	0

Buttons: Add, Delete, Load, Save, Ok, Cancel

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.

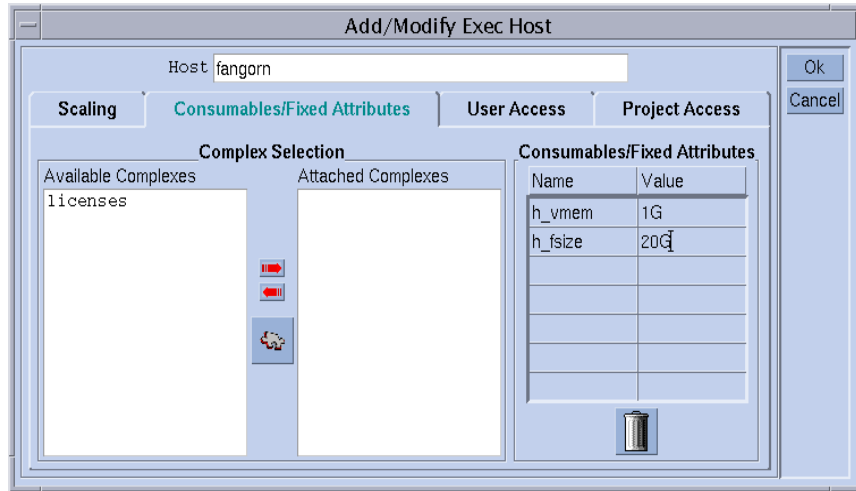


FIGURE 8-16 Execution Host Configuration—`h_fsize`

Submitting jobs to a Sun Grid Engine, Enterprise Edition 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 the 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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition to occupy disk space concurrently, the Sun Grid Engine, Enterprise Edition 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, Enterprise Edition.

The Sun Grid Engine, Enterprise Edition load sensor interface (see “Adding Site-Specific Load Parameters” on page 237) allows you to enhance the set of standard Sun Grid Engine, Enterprise Edition 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, Enterprise Edition will compare job requirements for disk space with the available capacity derived from the Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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 214 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	value	relop	requestable	consumable	default
#-----							
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, Enterprise Edition 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 169). 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 216 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, Enterprise Edition 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, Enterprise Edition 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 237) 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 host-specific configuration (see the section, “The Basic Cluster Configuration” on page 184 or the `sge_conf` manual page).

The corresponding QMON screen might look like the example in FIGURE 8-17.

Cluster Settings
Configuration for Host

dwain

General Settings

Master Spool Dir
Execd Spool Dir
Binary Path
Mailer: /usr/sbin/Mail
Xterm: /usr/bin/X11/xterm
Load Sensor: /usr/local/bin/load.sh
Admin User
Admin Mail
Prolog
Epilog
Login Shells
Maximal Array Job Instances
Maximal Array Job Tasks
Maximal Jobs Per User
Shell Start Mode: posix_compliant
Log Level: log_info

Advanced Settings

Min UID: 0
Min GID: 0
Finished Jobs: 0
Loadreport Time
Max Unheard
Stat Log Time
Reschedule Unknown
User Lists
Xuser Lists
Projects
XProjects
Ignore PQDN: False
Enforce Project: False
Enforce User: False
GID Range for Job Monitoring

Ok
Cancel

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.

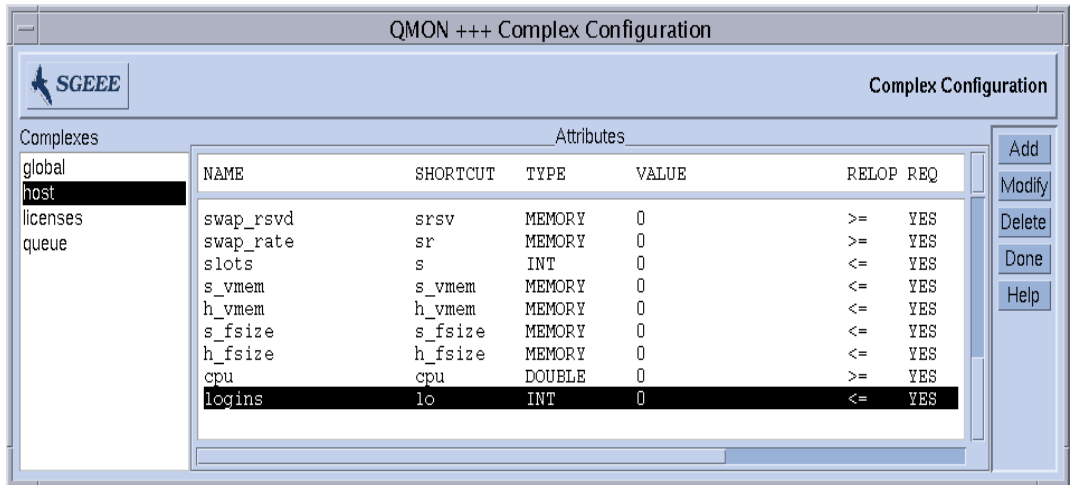


FIGURE 8-18 Complex Configuration Dialogue Box—logins

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, Enterprise Edition system. Topics in this chapter include user access, projects, 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.

- “How To Configure Accounts with QMON” on page 246
- “How To Configure Manager Accounts with QMON” on page 246
- “How To Configure Manager Accounts from the Command Line” on page 247
- “How To Configure Operator Accounts with QMON” on page 248
- “How To Configure Operator Accounts from the Command Line” on page 249
- “How To Configure User Access Lists with QMON” on page 251
- “How To Configure User Access Lists from the Command Line” on page 253
- “How To Configure the User Object with QMON” on page 254
- “How To Assign a Default Project” on page 256
- “How To Configure the User Object from the Command Line” on page 257
- “How To Define Projects with QMON” on page 258
- “How To Define Projects from the Command Line” on page 262
- “How To Change the Scheduler Configuration with QMON” on page 271
- “How To Administer Policy/Ticket Based Advanced Resource Management with QMON” on page 273
- “How To Edit the Share Tree Policy From QMON” on page 278
- “How To Configure the Share-Based Policy from the Command Line” on page 284
- “How To Configure the Functional Share Policy From QMON” on page 287
- “How To Configure the Functional Share Policy from the Command Line” on page 290
- “How To Configure the Override Policy” on page 296
- “How To Configure the Override Policy from the Command Line” on page 298
- “How To Configure Checkpointing Environments with QMON” on page 307
- “How To Configure the Checkpointing Environment from the Command Line” on page 310

About Setting Up a User

The following list describes the necessary/available tasks in order to set up a user for Sun Grid Engine, Enterprise Edition:

- 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 `sges_qmaster` runs.

- Setting Sun Grid Engine, Enterprise Edition Access Permissions

Sun Grid Engine, Enterprise Edition 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 250 for a detailed description.

In addition, a Sun Grid Engine, Enterprise Edition system user may get the permission to suspend or enable certain queues (see “How To Configure Owners” on page 205 for more information).

- Declaration of a Sun Grid Engine, Enterprise Edition User

If you intend to include a node in the share tree for the users or to define a functional or override policy for the user (see the section, “How To Administer Policy/Ticket Based Advanced Resource Management with QMON” on page 273), the user must be declared to the Sun Grid Engine, Enterprise Edition system. See “How To Configure the User Object with QMON” on page 254 for details.

- Sun Grid Engine, Enterprise Edition Project Access

If Sun Grid Engine, Enterprise Edition projects are used for the definition of share-based, functional or override policies (see the section, “How To Administer Policy/Ticket Based Advanced Resource Management with QMON” on page 273), the user should be given access to one or multiple projects. Otherwise, the user’s jobs may end up in the lowest possible priority class and will hardly receive access to resources.

- File Access Restrictions

Sun Grid Engine, Enterprise Edition users need to have read access to the directory `<sges_root>/cell/common`.

Before a Sun Grid Engine, Enterprise Edition job is started, the Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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 “Verifying the Installation” on page 70.

As Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition users.

■ Sun Grid Engine, Enterprise Edition Definition Files

You can set up the following definition files for Sun Grid Engine, Enterprise Edition users.

- `qmon` (the resource file for the Sun Grid Engine, Enterprise Edition GUI; see the section, “Customizing QMON” on page 35)
- `sge_aliases` (current working directory path aliases; see the section, “About Path Aliasing” on page 300)
- `sge_request` (default request definition file; see the section, “About Configuring Default Requests” on page 302).

About User Access

Four user categories exist in the Sun Grid Engine, Enterprise Edition system.

- **Managers** – Managers have full capabilities to manipulate Sun Grid Engine, Enterprise Edition. 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.
- **Users** – Users have certain access permissions, as described in “About User Access Permissions” on page 250, but no cluster or queue management capabilities.

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)
 - Userset access/department list configuration (see FIGURE 9-3)
 - User configuration (see FIGURE 9-5)
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, Enterprise Edition 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.

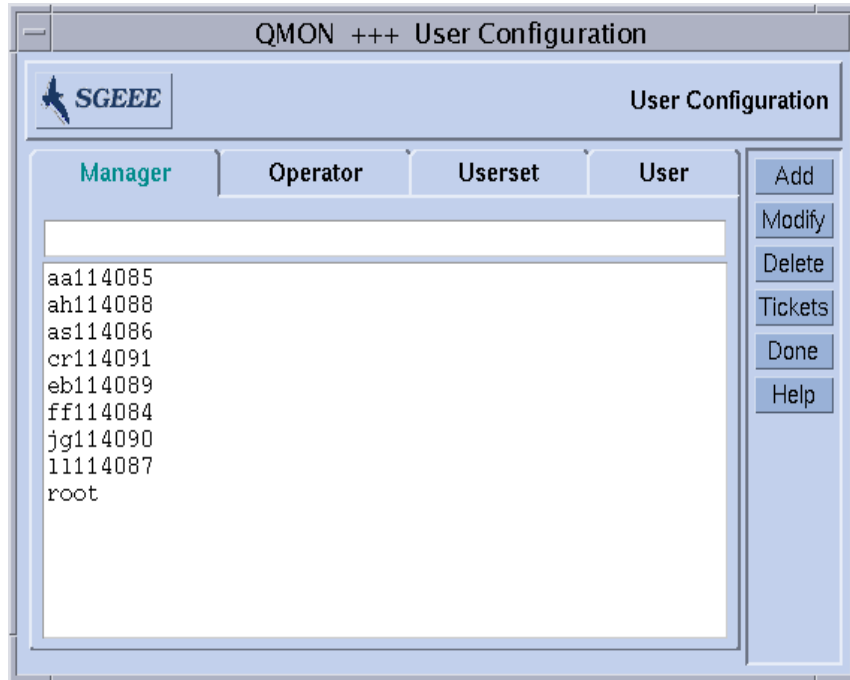


FIGURE 9-1 Manager Configuration Dialogue Box

▼ 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, Enterprise Edition managers. By default the root accounts of all Sun Grid Engine, Enterprise Edition trusted hosts (see the section, “About Daemons and Hosts” on page 169) are Sun Grid Engine, Enterprise Edition managers.

- `qconf -dm user_name[,...]`

Delete manager – This command deletes the specified users from the list of Sun Grid Engine, Enterprise Edition managers.

- `qconf -sm`

Show managers – This command shows the list of all Sun Grid Engine, Enterprise Edition managers.

▼ How To Configure Operator Accounts with QMON

When 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, Enterprise Edition command permission (unless they are also declared to be manager accounts—see “How To Configure Manager Accounts with QMON” on page 246). 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.

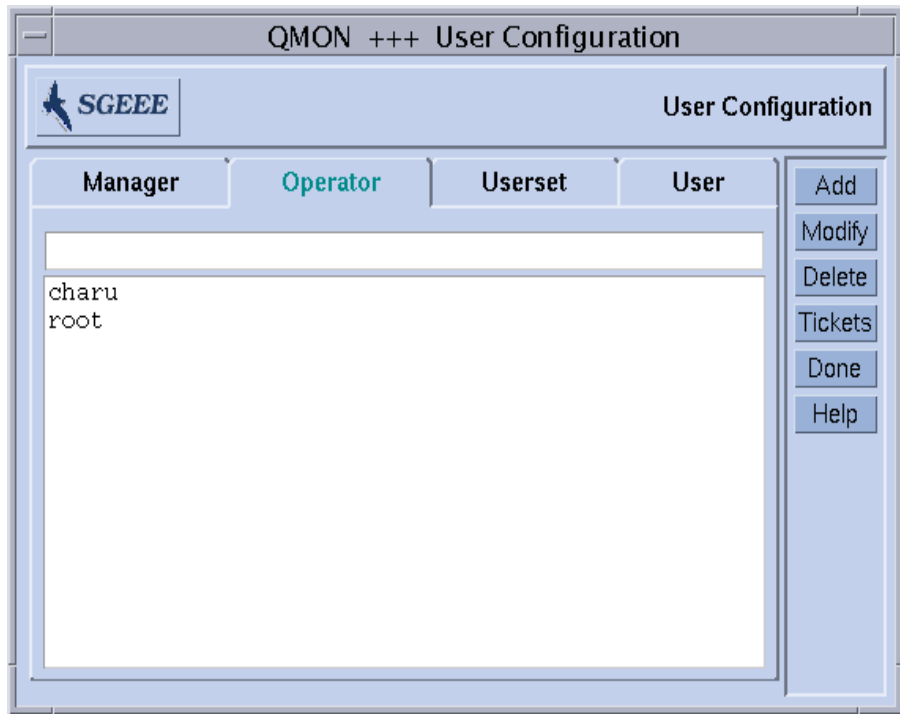


FIGURE 9-2 Operator Configuration Dialogue Box

▼ 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, Enterprise Edition operators.

- `qconf -do user_name[,...]`

Delete operator – This command deletes the specified users from the list of Sun Grid Engine, Enterprise Edition operators.

- `qconf -so`

Show operators – This command shows the list of all Sun Grid Engine, Enterprise Edition operators.

About Queue Owner Accounts

Queue owners are defined during configuration or modifications of a Sun Grid Engine, Enterprise Edition queue. Refer to sections, “How To Configure Queues with QMON” on page 192 and “How To Configure Queues from the Command Line” on page 206. 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, Enterprise Edition 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, Enterprise Edition system. However, Sun Grid Engine, Enterprise Edition 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 313) 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 184), in the queue configuration (see the section,

“How To Configure Subordinate Queues” on page 202) or in the process of configuring parallel environment interfaces (see the section, “How To Configure PEs with QMON” on page 314) 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.

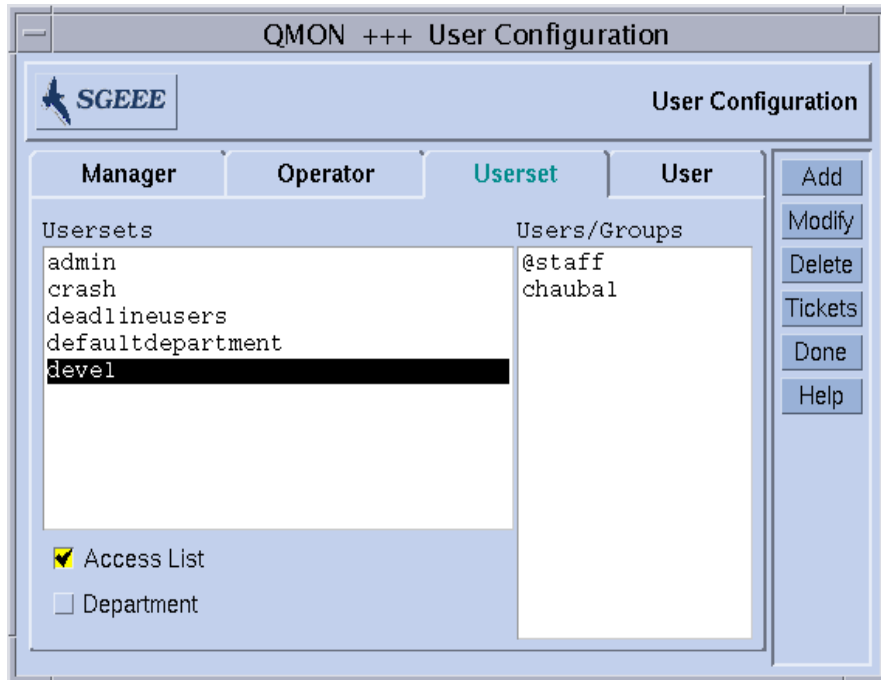


FIGURE 9-3 Userset Configuration Dialogue Box

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.

In Sun Grid Engine, Enterprise Edition, a Userset can be either an Access List, a Department, or both. The two corresponding flags below the Usersets selection list indicate the type. This section assumes that all Usersets are access lists. Departments are explained in the section, “About Using Usersets To Define Projects and Departments” on page 254.

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 userset by pressing the Add button.
- **Modification** – Modify a selected access list by pressing the Modify button.

In the cases of addition and modification, the Access List Definition dialogue box, similar to the one displayed in FIGURE 9-4, is opened and provides the corresponding means.

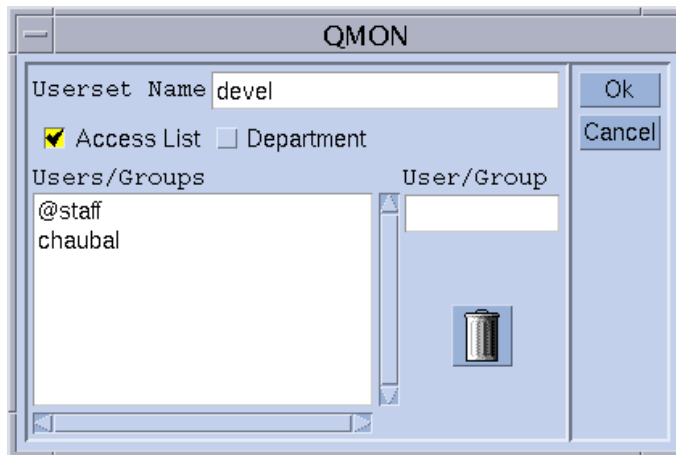


FIGURE 9-4 Access List Definition Dialogue Box

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.

For the definition of access lists in Sun Grid Engine, Enterprise Edition, ensure that the Access List flag is selected. See the section, “About Using Usersets To Define Projects and Departments” on page 254 for an explanation of the Department flag.

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 -Au filename`
Add a user access list from file—This command uses a configuration file, *filename*, to add an access list.
- `qconf -du user_name[,...] access_list_name[,...]`
Delete user—This command deletes one or more users from the specified access list(s).
- `qconf -dul access_list_name [...]`
Delete user list—This command completely removes userset lists.
- `qconf -mu access_list_name`
Modify user access list—This command is used to modify the specified access lists.
- `qconf -Mu filename`
Modify user access list from file—This command uses a configuration file, *filename*, to modify the specified access lists.

- `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 Using Usersets To Define Projects and Departments

Usersets are also used to define Sun Grid Engine, Enterprise Edition projects (see “About Projects” on page 258) and departments. Departments are used for the configuration of the Sun Grid Engine, Enterprise Edition policies, *Functional* (see “About the Functional Policy” on page 285) and *Override* (see “About the Override Policy” on page 294). They differ from access lists in that a user can only be a member of a single department, while the same user can be contained in multiple access lists. In addition, the Userset with the preserved name, *deadlineusers*, contains all users who are allowed to submit deadline jobs through the Sun Grid Engine, Enterprise Edition software (see “About the Deadline Policy” on page 291).

A Userset is identified as a department by the Department flag shown in FIGURE 9-3 and FIGURE 9-4. If a Userset is a department it can be used and defined as an access list at the same time. However, the restriction of only a single appearance by any user in any department applies.

About User Object Configuration

If share-based, functional or override policies (see “How To Administer Policy/Ticket Based Advanced Resource Management with QMON” on page 273) are intended to be defined for users, Sun Grid Engine, Enterprise Edition software needs to have these user names declared before the policies can be defined. Users are declared via the User Configuration dialogue box.

▼ How To Configure the User Object with QMON

1. In the QMON Main menu, press the User Configuration button.

2. Select the User tab on the top of the screen.

The User Configuration dialogue box, similar to that shown in FIGURE 9-5, is presented.

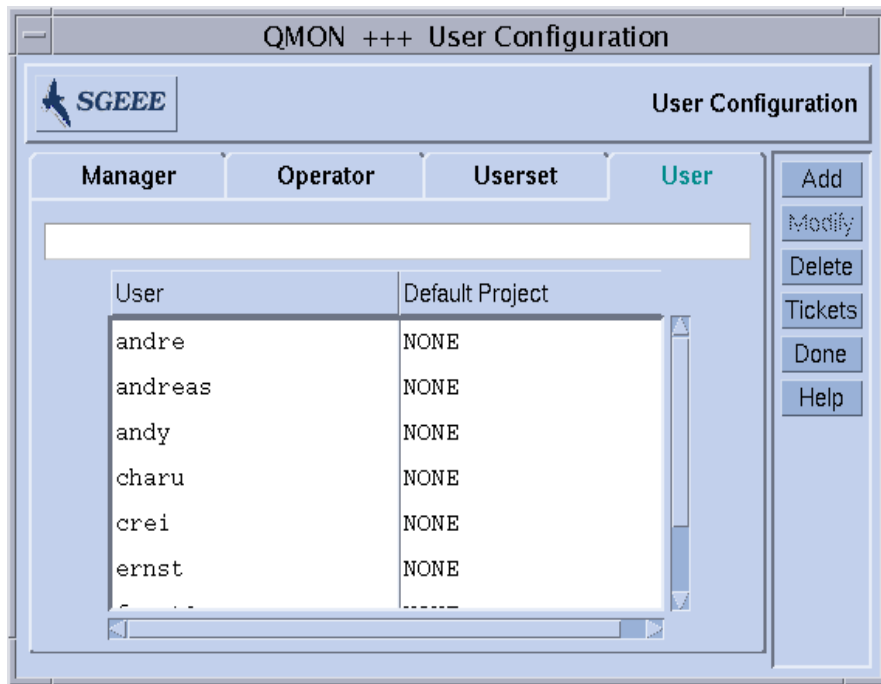


FIGURE 9-5 User Configuration Dialogue Box

3. Depending on what you want to accomplish, enter user names on the input line at the top of the dialogue box—or select the name if it is already listed in the box—and then do one of the following.

Add or Delete

- **Add a new user name** – After entering the name, press the Add button or press Return on your keyboard.
- **Delete a user name** – After selecting the name, press the Delete button.

▼ How To Assign a Default Project

You can assign a default project (see “About Projects” on page 258) to each user. The default project will be attached to each job, which the user submits without requesting another project to which he or she has access.

1. To assign a default project, highlight a user entry by clicking on it.
2. Press the Default Project button at the top of the list.

The Project Selection dialogue box, similar to that shown in FIGURE 9-6, is presented.



FIGURE 9-6 Project Selection Dialogue Box

3. Select an appropriate project for the highlighted user entry.
4. Press OK to assign the default project and close the dialogue box.

▼ How To Configure the User Object from the Command Line

- Enter the following command with appropriate options.

```
# qconf options
```

Available Options

- **qconf -auser**

Add user—This command opens a template user configuration (see the user entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual*) in the editor specified via `$EDITOR` or (by default) `vi` and enables you to modify it. After saving your changes and exiting the editor, the changes are registered with `sgc_qmaster`.

- **qconf -Auser filename**

Add user from file—This command parses the specified file—which must have the user configuration template format—and adds the user configuration.

- **qconf -duser user_name[,...]**

Delete user—This command deletes one or more user objects.

- **qconf -muser user_name**

Modify user—This command modifies an existing user entry. It loads the user configuration in the editor specified via `$EDITOR` or (by default) `vi` and allows you to modify it. After saving your changes and exiting the editor, the changes are registered with `sgc_qmaster`.

- **qconf -Muser filename**

Modify user from file—This command parses the specified file—which must have the user configuration template format—and modifies the user configuration.

- **qconf -suser user_name**

Show user—This command displays the configuration of a particular user.

- **qconf -suserl**

Show user list—This command prints a listing of all users currently defined.

About Projects

Sun Grid Engine, Enterprise Edition projects provide a means to organize joint computational tasks from multiple users and to define resource utilization policies for all jobs belonging to such a project. Projects are used in three scheduling policy areas:

- share-based, when shares are assigned to projects (see section “About the Share-Based Policy” on page 275).
- functional, when projects receive a per-cent of the functional tickets (see section “About the Functional Policy” on page 285)
- override, when an administrator grants override tickets to a project (see section “About the Override Policy” on page 294)

Note – Projects have to be declared before they can be used in any of the three policies.

Sun Grid Engine, Enterprise Edition managers define Sun Grid Engine, Enterprise Edition projects by giving them a name and some attributes. Sun Grid Engine, Enterprise Edition users may attach a project to a job during job submission. Associating a job with a project influences the job’s dispatching depending on the project’s share of share-based, functional and/or override tickets.

▼ How To Define Projects with QMON

Sun Grid Engine, Enterprise Edition managers may define and update definitions of Sun Grid Engine, Enterprise Edition projects by using the Project Configuration dialogue box.

1. From the QMON Main menu, click the Project Configuration icon.

The Project Configuration dialogue box, which is similar to the example in FIGURE 9-7, is presented.

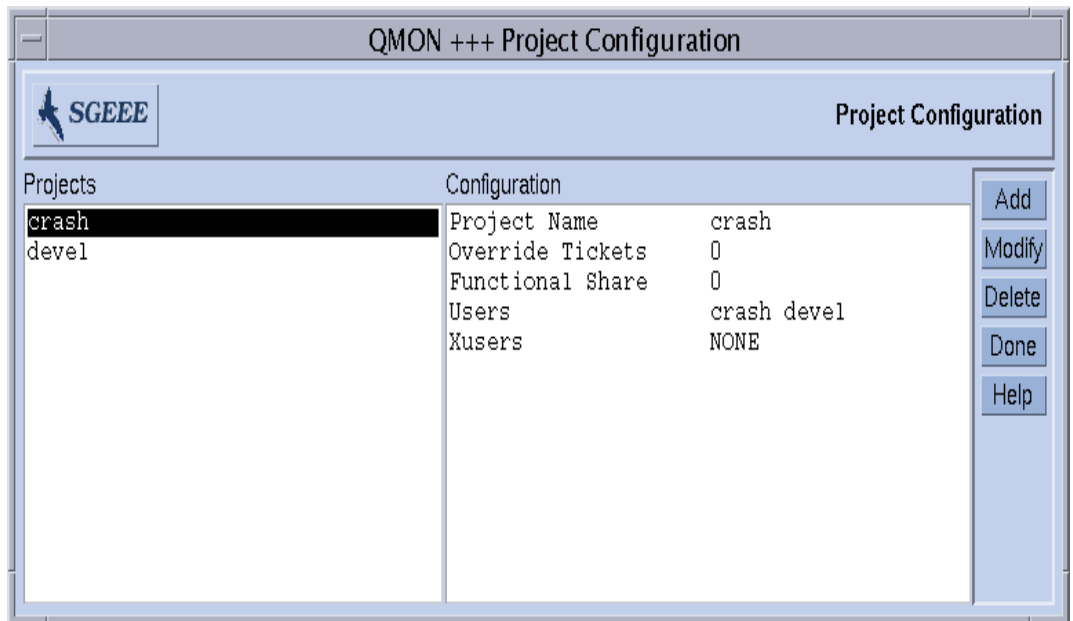


FIGURE 9-7 Project Configuration Dialogue Box

The already defined projects are shown in the Projects selection list on the left side of the screen.

2. Click the name of any listed project.

The project definition is displayed in the Configuration window.

3. Depending on what you want to accomplish, do one of the following.

- a. Press Delete to remove the highlighted project immediately.

b. Press Add to add a new project or Modify to modify the highlighted project.

Pressing Add or Modify will each cause the Add/Modify Project dialogue box, similar to the example in FIGURE 9-8, to be displayed.

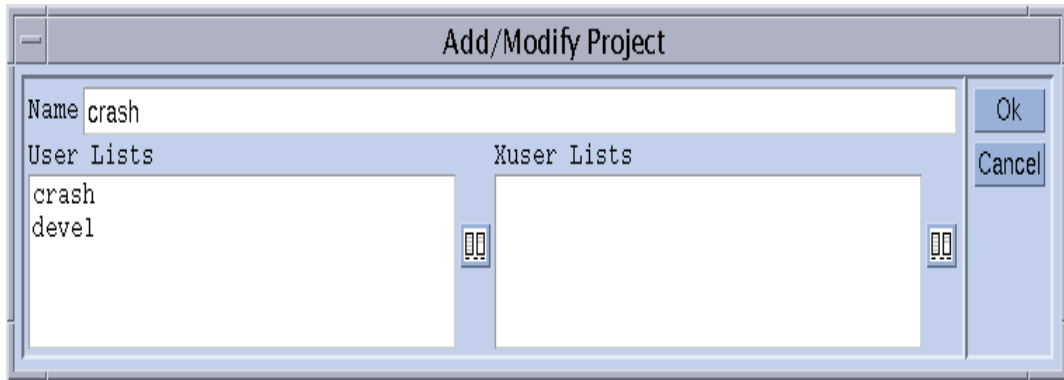


FIGURE 9-8 Add/Modify Project Dialogue Box

c. In the Add/Modify Project dialogue box, proceed according to the following guidelines.

- **When adding or modifying a project**, the Name input field at the top of the Add/Modify Project dialogue box denotes the project name. The project is defined by the users who are permitted or denied access to the project.
- **Specify permission or denial** by attaching user access lists (see the section, "About User Access Permissions" on page 250) to User Lists (access permitted) or Xuser Lists (access denied). Users or user groups contained in access lists attached to User Lists are permitted to submit jobs to the project. Users or user groups listed in Xuser Lists are denied permission to use the project. *If both lists are empty, any user can access the project.* If a user is contained in different access lists that are attached to *both* the User List and the Xuser List, the user is *denied* access.
- **To add users to, or remove them from, User Lists and Xuser Lists**, click the icon button on the right of the User Lists and Xuser Lists windows. This opens the Select Access Lists dialogue box, similar to the example shown in FIGURE 9-9.

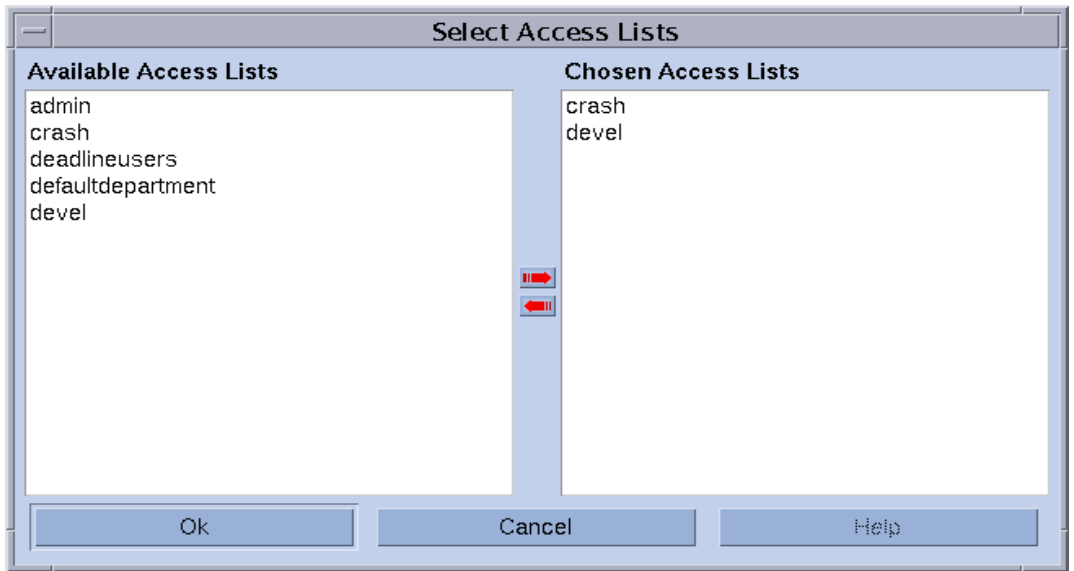


FIGURE 9-9 Select Access Lists Dialogue Box

The Select Access Lists dialogue box displays all defined access lists in the Available Access Lists window and the attached lists in the Chosen Access Lists window. You can select access lists in both windows and move between them via the arrow icon buttons.

d. Click the OK button to commit the changes and close the dialogue box.

▼ How To Define Projects from the Command Line

- Enter the following command with appropriate options.

```
# qconf options
```

Available Options

- `qconf -apr j`

Add project—This command opens a template project configuration (see the project entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual*) in the editor specified via `$EDITOR` or (by default) `vi` and enables you to modify it. After saving your changes and exiting the editor, the changes are registered with `sge_qmaster`.

- `qconf -Apr j filename`

Add project from file—This command parses the specified file—which must have the project configuration template format—and adds the new project configuration.

- `qconf -dpr j project_name[,...]`

Delete project—This command deletes one or more projects.

- `qconf -mpr j project_name`

Modify project—This command modifies an existing user entry. Loads the project configuration in the editor specified via `$EDITOR` or (by default) `vi` and enables you to modify it. After saving your changes and exiting the editor, the changes are registered with `sge_qmaster`.

- `qconf -mpr j filename`

Modify project from file—This command parses the specified file—which must have the project configuration template format—and modifies the existing project configuration.

- `qconf -spr j project_name`

Show project—This command displays the configuration of a particular project.

- `qconf -spr jl`

Show project list—This command prints a listing of all projects currently defined.

About Scheduling

The Sun Grid Engine, Enterprise Edition 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,
- **Post-dispatch monitoring**—These activities involve adjusting a job's relative importance as it gets resources and as other jobs with their own relative importance enter or leave the system.

Sun Grid Engine, Enterprise Edition software schedules jobs across a heterogeneous cluster of computers based on the following.

- The cluster's current load
- The jobs' relative importance
- The hosts' relative performance
- 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, Enterprise Edition 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, Enterprise Edition scheduling tasks.

- **Dynamic resource management**—The Sun Grid Engine, Enterprise Edition system dynamically controls and adjusts the resource entitlements allocated to running jobs (i.e., it modifies their CPU share).
- **Queue sorting**—The software ranks the queues in the cluster according to the order in which the queues should be filled up.
- **Job sorting**—This determines the order in which the Sun Grid Engine, Enterprise Edition system attempts to schedule jobs.

Dynamic Resource Management

Sun Grid Engine, Enterprise Edition software implements automated job scheduling strategies using a weighted combination of four policies.

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

You can set up the Sun Grid Engine, Enterprise Edition system to routinely use either a share-based policy, a functional policy, or both. These policies can be combined in any proportion, from giving zero weight to one and using only the second to giving both equal weight.

Along with the routine policies, jobs may be submitted with an initiation deadline. Deadline jobs perturb routine scheduling. Administrators may also override share-based, functional and initiation deadline scheduling temporarily or, for certain purposes such as express queues, permanently. An override may be applied to an individual job, or all jobs associated with a user, a department, a project, or a job class (i.e., queue).

In addition to the four policies for mediating among all jobs, Sun Grid Engine, Enterprise Edition sometimes lets users set priorities among their own jobs. A user submitting several jobs may say, for example, that job 3 is the most important and jobs 1 and 2 are equally important but less important than job 3. This is possible if the Sun Grid Engine, Enterprise Edition system's combination of policies includes the share-based policy, the functional policy, or both, with functional tickets granted to jobs.

Scheduling policies are implemented with tickets. Each policy has a pool of tickets from which it allocates tickets to jobs entering the multi-machine Sun Grid Engine, Enterprise Edition system. Each routine policy that is in force allocates some tickets to each new job and possibly reallocates tickets to executing jobs at each scheduling interval. The criteria each policy uses to allocate tickets are explained below.

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

Tickets are allocated to the routine policies at system configuration by Sun Grid Engine, Enterprise Edition managers. Managers and operators may change ticket allocations at any time with immediate effect. Additional tickets are injected into the system temporarily to indicate a deadline or an override. Policies are combined by assignment of tickets — when tickets are allocated to multiple policies a job gets a portion the tickets of each policy, which indicates its importance in each policy in force.

Sun Grid Engine, Enterprise Edition grants tickets to jobs entering the system to indicate their importance under each policy in force. Each executing job may gain (for example, from an override or because a deadline is approaching), lose (for example, because it is getting more than its fair share of resources) or keep the same number of tickets at each scheduling interval. The number of tickets a job holds represent the resource share Sun Grid Engine, Enterprise Edition tries to grant that job during each scheduling interval.

A site's dynamic resource management strategy is configured during Sun Grid Engine, Enterprise Edition installation by allocating tickets to the share-based and functional scheduling policies, by defining the share tree and functional shares, and by setting a maximum number of initiation deadline tickets. The share-based and functional ticket allocations and the initiation deadline ticket maximum may change automatically at any time. The override tickets are manually assigned or removed by the administrator.

Queue Sorting

The following means are provided to determine the order in which Sun Grid Engine, Enterprise Edition attempts to fill up queues.

- **Load reporting**—Sun Grid Engine, Enterprise Edition 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 237.
- **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 175).
- **Load adjustment**—Sun Grid Engine, Enterprise Edition 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.
- **Host capacity**—Hosts and the queues located on them can be sorted based on a capacity indicator, defining the relative power of the machines in the cluster.

Job Sorting

Before Sun Grid Engine, Enterprise Edition starts dispatching, jobs are brought into an order of highest priority first. Sun Grid Engine, Enterprise Edition 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.

- **Ticket-based job priority**—In Sun Grid Engine, Enterprise Edition, jobs are always treated corresponding to their relative importance defined by the number of tickets they possess. Therefore, pending jobs are sorted in ticket order and any ticket policy change the administrator applies, also changes the sorting order.
- **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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition system does the following.

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

If share-based scheduling is used in the Sun Grid Engine, Enterprise Edition system, the calculation takes into account the usage that has already occurred for that user, or project. If scheduling is not (at least in part) share based, the calculation simply ranks all the jobs executing and waiting to execute and takes the most important until it utilizes the resources (CPU, memory, and I/O bandwidth) in the cluster as fully as possible.

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, Enterprise Edition 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 271.

To retrieve even further detail about the decisions of the Sun Grid Engine, Enterprise Edition 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.

Scheduler Configuration

Refer to “How To Administer Policy/Ticket Based Advanced Resource Management with QMON” on page 273 for details on the scheduling administration of ticket bases resource sharing policies of Sun Grid Engine, Enterprise Edition. The remainder of this section focuses on administering the scheduler configuration, `sched_conf` and related issues.

Default Scheduling

The default Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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.

Scaling System Load

The Sun Grid Engine, Enterprise Edition 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 237 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, Enterprise Edition administrator should define load scaling factors for the concerning execution hosts and load parameters (see “How To Configure Execution Hosts with QMON” on page 175, 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, Enterprise Edition 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 237) and consumable resources (see the section, “Consumable Resources” on page 224) 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, Enterprise Edition 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, Enterprise Edition allows the administrator to specify *load adjustment* factors which are used in the Sun Grid Engine, Enterprise Edition 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, Enterprise Edition cluster configuration parameter *queue_sort_method* to *seqno* 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 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 – Which queue is selected is *not* defined if the considered queues all share the same sequence number.

Selecting Queue by Share

The goal of this method is to place jobs so as to attempt to meet the targeted share of global system resources for each job. This method takes into account the resource capability represented by each host in relation to all the system resources and attempts to balance the percentage of Sun Grid Engine, Enterprise Edition tickets for each host (i.e., the sum of Sun Grid Engine, Enterprise Edition tickets for all jobs running on a host) with the percentage of the resource capability which that particular host represents for the system. Refer to “How To Configure Execution Hosts with QMON” on page 175 for instructions on how to define the capacity of a host.

The hosts’ load is also taken into account in the sorting although it is of secondary importance. This should be the sorting method of choice for a site using the share tree policy.

Restricting the Number of Jobs per User or Group

The Sun Grid Engine, Enterprise Edition 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 9-10.

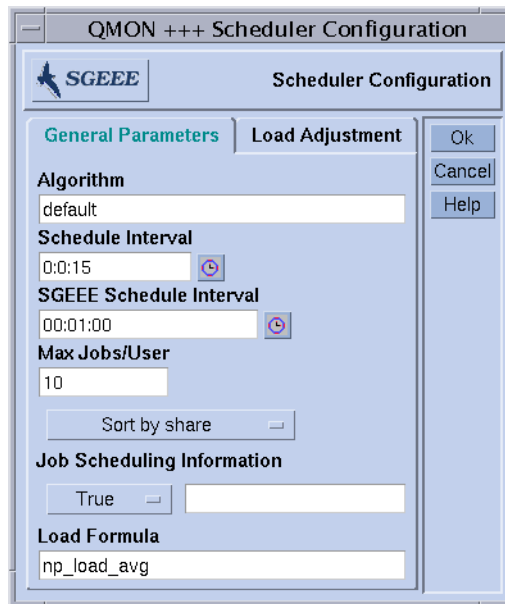


FIGURE 9-10 Scheduler Configuration Dialogue Box—General Parameters

From the General Parameters dialogue box, you can set the following parameters.

- The scheduling algorithm (see “Changing the Scheduling Algorithm” on page 268)
- The regular time interval between scheduler runs
- The regular time interval between Sun Grid Engine, Enterprise Edition scheduler runs, that is re-distribution of tickets based on the resource sharing policies
- 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 270).

- The queue sorting scheme—either sorting by load or sorting by sequence number (see “Selecting Queue by Sequence Number” on page 269) or sorting by share (see “Selecting Queue by Share” on page 270).
- 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 9-11.

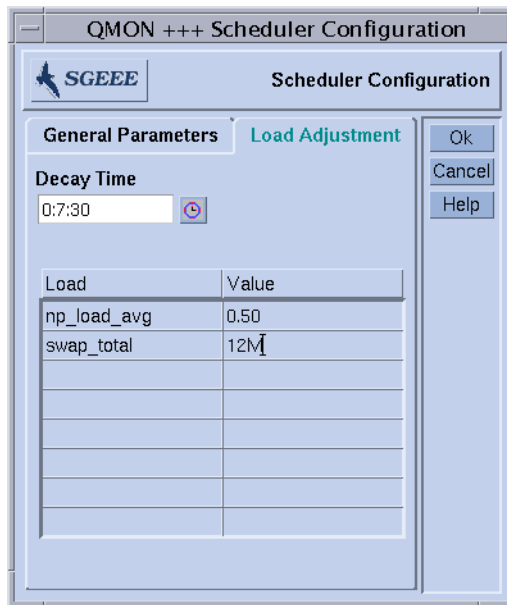


FIGURE 9-11 Scheduler Configuration Dialogue Box—Load Adjustment

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 268 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.

▼ How To Administer Policy/Ticket Based Advanced Resource Management with QMON

1. In the QMON Main menu, click the Ticket Configuration button.

The Ticket Overview dialogue box, similar to the example in FIGURE 9-12, is presented.


QMON +++ Ticket Overview	
<div>  Ticket Overview </div>	
Current Active Tickets Share Tree Tickets 1000000 Functional Tickets 1000 Deadline Initiation Tickets 0 Override Tickets 0	Edit Tickets Total Share Tree Tickets 1000000 Total Functional Tickets 1000 Maximum Deadline Initiation Tickets 10000
<div> <div>Refresh</div> <div>Apply</div> <div>Done</div> <div>Help</div> </div>	
Policy <div> <div>Users Allowed To Submit Deadline Jobs</div> <div>Share Tree Policy</div> <div>Functional Policy</div> <div>Override Policy</div> </div>	

FIGURE 9-12 Ticket Overview Dialogue Box

2. Proceed according to guidance in the following sections.

The Ticket Overview dialogue box shows the current ticket distribution among ticket based policies, allows to readjust the policy related tickets, and provides the gateway to specific configuration dialogues for all ticket based policies.

The tickets currently assigned to individual policies are shown in the Current Active Tickets display region on the left. The numbers reflect the relative importance of the policies and indicate whether a certain policy currently dominates the cluster or whether policies are in balance. Tickets provide a quantitative measure, meaning that twice the tickets assigned to the share-based policy than to the functional policy, for example, allocate twice the resource entitlement to the share-based policy than allocated to the functional policy. In this sense, tickets behave very much like stock shares.

The total amount of all tickets has no particular meaning. Only the relations between policies counts. Hence, total ticket numbers are usually quite high, to allow for fine-grain adjustment of the relative importance of the policies.

Edit Tickets Region

The Edit Tickets region allows to modify the tickets allocated to each policy except for the override policy. Override tickets are directly assigned through the override policy configuration whereas the other ticket pools are distributed among jobs associated with the policies automatically and with respect to the actual policy configuration.

Note – Always all share-based and functional tickets are distributed among the jobs associated with these policies. Deadline tickets are distributed only as deadline jobs approach their deadline. Override tickets may not be applicable to the currently active jobs, so active override tickets may be 0 while the override policy has tickets defined.

Policy Button Region

This region provides the following.

- A button to open the User Configuration dialogue box for easy access to the Deadlineusers Userset configuration
- A button to open the share-based, functional, and override policy configuration dialogue boxes—no configuration dialogue box is required for the deadline policy

The buttons on the right side of the panel enable you to Refresh the screen, Apply, or discard (Done) changes.

About the Share-Based Policy

Share-based (also called *share tree*) scheduling is a scheme that works toward granting each user and project its allocated share of system resources during an accumulation period such as a week, a month, or a quarter. It does this by constantly adjusting each user's and project's potential resource share for the near term (until the next scheduling interval). Share based scheduling is defined by user or by project or by both.

By giving each user/project its targeted share as far as possible, agglomerations of users/project such as departments or divisions also get their targeted share. Fair share for all entities is attainable only when every entity that is entitled to resources contends for them during the accumulation period. If a user/project or agglomeration does not submit jobs during some period, the resources will be shared among those who do.

Share-based scheduling is a *feedback scheme*. The share of the system to which any user/user-group and project/project-group is entitled is a Sun Grid Engine, Enterprise Edition configuration parameter. The share of the system to which any job is entitled is based on the following factors.

- The share allocated to the job's user or project
- The accumulated past usage for each user and user group, and project and project group, adjusted by a *decay factor* (i.e., "old" usage has less impact)

Sun Grid Engine, Enterprise Edition keeps track of how much usage users/projects have already received. At each scheduling interval, the Scheduler adjusts all jobs' share of resources to insure that all users/user groups and projects/project groups get very close to their fair share of the system over the accumulation period. In other words, resources are granted or denied in order to keep everyone more or less at their targeted share of usage.

The Half-life Factor

Half-life is how fast the system "forgets" about a user's resource consumption. The system administrator can decide whether to or how to penalize a user for high resource consumption, be it six months ago or six days ago. On each node of the share tree, Sun Grid Engine, Enterprise Edition software maintains a record of users' resource consumption.

With this record, the system administrator can decide how far to look back to determine a user's under-utilization or over-utilization when setting up a share-based policy. The resource usage in this context is a mathematical integral (sum) of all the computer resources consumed over a "sliding window of time."

The length of this window is determined by a "half-life" factor, which in the Sun Grid Engine, Enterprise Edition system is an internal decay function. This decay function reduces the impact of accrued resource consumption over time. A short half-life quickly lessens the impact of resource over-consumption; a longer half-life gradually lessens the impact of resource over-consumption.

In the Sun Grid Engine, Enterprise Edition system, this half-life decay function is a specified unit of time. For example, a half-life of seven days applied to a resource consumption of 1,000 units results in the following usage "penalty" adjustment over time.

- 500 after 7 days
- 250 after 14 days
- 125 after 21 days
- 62.5 after 28 days

The half-life based decay diminishes the impact of a user's resource consumption over time, until the penalization effect is very small and negligible. Note that if a user receives *override tickets*, these are not subjected to a past usage penalty, as they belong to a different policy system. The decay function is a characteristic of the share-tree policy only.

Compensation Factor

When the comparison shows that actual usage is well below targeted usage, adjusting a user's/project's share of resources might allow a user to dominate the system based on the goal of reaching target share. This domination may not be desirable. The *compensation factor* allows an administrator to limit how much a user/project with very little accumulated usage can dominate the resources in the near term in attempting to reach the specified usage target.

For example, a compensation factor of 2 limits a user's/project's current share to twice its targeted share. That is, if a user/project is supposed to get 20 percent of the system resources over the accumulation period and is currently getting much less, it can get only 40 percent in the near term.

In combination with the share-based policy, where long-term resource entitlements of users or projects are defined as per the share tree, the compensation factor makes automatic adjustments in entitlements.

If a particular user or project is either *below* or *over* the defined target entitlement, the Sun Grid Engine, Enterprise Edition system compensates by either *raising* or *lowering* that user's or project's entitlement for a short term over or under the long-term target. This compensation is performed by the Sun Grid Engine, Enterprise Edition system's share tree algorithm calculations.

The compensation factor provides an additional mechanism on top to control the amount of compensation that the Sun Grid Engine, Enterprise Edition system assigns. The additional compensation factor (CF) calculation is only carried out if the following are true.

- $\text{Short-term-entitlement} > \text{long-term-entitlement} * \text{CF}$
- $\text{CF} > 0$

If one or both of the above are not true, the compensation as defined and implemented by the share-tree algorithm is used.

A general rule for setting the compensation factor is that the smaller the value of CF, the greater will be its effect. If the value is greater than 1, then the Sun Grid Engine, Enterprise Edition system will compensate—but the compensation will be limited. The upper limit for compensation is calculated as $\text{long-term-entitlement} * \text{CF}$. Note also that, as defined above, the short-term entitlement must exceed this limit before anything happens based on the compensation factor.

If the value = 1, then the Sun Grid Engine, Enterprise Edition system compensates in the same way as with the raw share-tree algorithm. So a value of 1 has a similar effect as a value of 0. The only difference is an implementation detail that the CF calculations are carried out (without an effect) while they are suppressed if $\text{CF} = 0$.

If the value is < 1 , then the Sun Grid Engine, Enterprise Edition system “overcompensates.” Jobs receive much more compensation than they are entitled to based on the share tree algorithm. They also receive this overcompensation earlier because the $\text{short-term-entitlement} > \text{long-term-entitlement} * \text{CF}$ criterion for activating it is met at lower short-term entitlement values.

Hierarchical Share Tree

The share-based policy is implemented through a *hierarchical share tree* that specifies, for a moving accumulation period, how system resources are to be shared among all users/projects. The length of the accumulation period is determined by a configurable decay constant. Sun Grid Engine, Enterprise Edition bases a job’s share entitlement on the degree to which each parent node in the share tree has reached its accumulation limit. A job’s share entitlement is based on its leaf node share allocation which in turn depends on the allocations of its parent nodes. All jobs associated with a leaf node split the associated shares.

The entitlement derived from the share tree is combined with other entitlements (e.g., entitlement from a deadline or a functional policy) in determining a job’s net entitlement. The share tree is allotted the total number of tickets for share-based scheduling. This number determines the weight of share-based scheduling among the four scheduling policies.

The share tree is defined during Sun Grid Engine, Enterprise Edition installation and may be altered at any time. When the share tree is edited, the new share allocations take effect at the next scheduling interval.

▼ How To Edit the Share Tree Policy From QMON

1. At the bottom of the QMON Ticket Overview dialogue box, click Share Tree Policy.

The Share Tree Policy dialogue box, similar to the example in FIGURE 9-13, is presented.

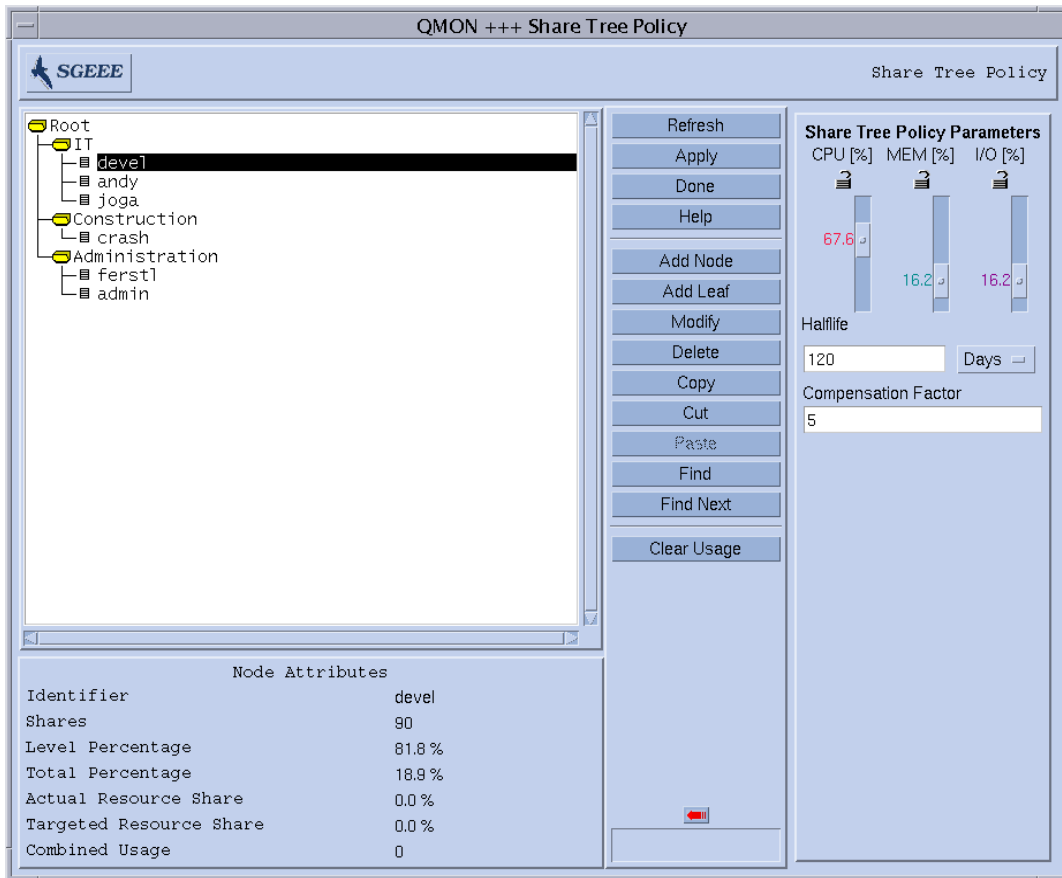


FIGURE 9-13 Share Tree Policy Dialogue Box

2. Proceed to edit the policy, according to guidance from the following sections.

Node Attributes Display

This region shows the attributes of the selected node:

- **Identifier**—a user, project or agglomeration name.
- **Shares**—the number of shares allocated to this user or project.

Note – Shares define relative importance and are no percentages. They also do not have quantitative meaning. Picking numbers in the hundreds or even thousands is generally a good idea as it allows fine tuning of importance relationships.

- **Level Percentage** – This node's portion of the total shares at its level (same parent node) in the tree; its shares divided by the sum of his and his sibling's shares.
- **Total Percentage** – This node's portion of the total shares in the entire share tree. This is the long term targeted resource share of the node concerning the share-based policy.
- **Actual Resource Usage** – The percentage of all the resources in the system which this node has consumed so far in the accumulation period. The percentage is expressed in relation to all nodes in the share tree.
- **Targeted Resource Usage** – Same as above, but only taking the currently active nodes in the share tree into account. Active nodes have jobs in the system. In the short term, Sun Grid Engine, Enterprise Edition attempts to balance the entitlement among active nodes.
- **Combined Usage** – The total usage for the node. Combined Usage is the sum of the usage accumulated at this node. Leaf nodes accumulate the usage of all jobs running under them. Inner nodes accumulate the usage of all descendant nodes. Combined Usage is composed of CPU, Memory and I/O usage according to the ratio specified in the Share Tree Policy Parameters dialogue section and is decayed at the half-life decay rate specified there.

When a user or project node (as a leaf node) is removed and then added back either at the same place or at a different place in the share tree, the user's or project's usage is retained. If you wish to zero out that usage before adding the user or project node back, then the user/project should be removed from and added back to the users/projects configured in Sun Grid Engine, Enterprise Edition.

Even if a user or project has never been included in the share tree, but has been running jobs that user or project will have non-zero usage upon being added to the share tree. Again, if it is desired that the user or project have zero usage upon being added to the tree, then it should be removed from the users or projects configured in Sun Grid Engine, Enterprise Edition before being added to the tree.

Refresh

The graphical user interface periodically updates the information which it displays. This button forces an immediate display refresh.

Apply

Clicking this button applies all the additions, deletions and node modifications you have made but keeps the window open.

Done

Clicking this button closes the window without applying the additions, deletions and node modifications you have made.

Help

Clicking this button opens on-line help.

Add Node

Click this button to add an interior node under the selected node. Clicking this button opens a blank Node Info screen where you may enter the node's name and number of shares. The nodes name and the shares you may enter are arbitrary.

Add Leaf

Click this button to add a leaf node under the selected node. Clicking this button opens a blank Node Info screen where you may enter the node's name and number of shares. The nodes name has to be an existing Sun Grid Engine, Enterprise Edition user ("How To Configure the User Object with QMON" on page 254) or Sun Grid Engine, Enterprise Edition project ("About Projects" on page 258).

The following rules apply:

- All nodes have a unique path in share tree.
- A project is not referenced more than once in share tree.
- A user appears only once in a project sub-tree.
- A user appears only once outside of a project sub-tree.
- A user does not appear as a non-leaf node.

- All leaf nodes in a project sub-tree reference a known user or the reserved name, “default.” (See a detailed description of this special user in the section, “About the Special User, default” on page 283.)
- There are no sub-projects within a project sub-tree.
- All leaf nodes not in a project sub-tree reference a known user or project.
- All user leaf nodes in a project sub-tree have access to the project.

Modify

Click this button to edit the selected node. Clicking this button opens a Node Info screen that displays the name of the selected node and its number of shares.

Delete

Clicking this button deletes the selected node and all its descendants.

Copy

Clicking this button copies the selected node together with its descendants into a paste buffer.

Cut

Clicking this button cuts the selected node together with its descendants off the share tree. The cut off part is copied into the paste buffer.

Paste

Clicking this button pastes the most recently copied node under the selected node.

Find

This button opens an input box for entering the search string and then searches in the share tree for a corresponding name. Node names are indicated which begin with the case sensitive search string.

Find Next

Finds the next occurrence of the search string.

Clear Usage

By pressing this button, you set back to 0 all of the accumulated in the entire share-tree hierarchy. This button is particularly useful in cases where the share-based policy is aligned to a budget and needs to start from scratch at the beginning of each budget term. The Clear Usage facility also is handy when setting up or modifying test Sun Grid Engine, Enterprise Edition environments.

Large Arrow Navigator

Single-click on this arrow to open the Share Tree Policy Parameters portion of this window.

Share Tree Policy Parameters

- **CPU (%) slider**—This slider's setting indicates what percentage of Combined Usage CPU is. When you change this slider, the MEM and I/O sliders change to compensate for the change in CPU percentage.
- **MEM (%) slider**—This slider's setting indicates what percentage of Combined Usage memory is. When you change this slider, the CPU and I/O sliders change to compensate for the change in MEM percentage.
- **I/O (%) slider**—This slider's setting indicates what percentage of Combined Usage I/O is. When you change this slider, the CPU and MEM sliders change to compensate for the change in I/O percentage.

Note – CPU(%), MEM(%), and I/O(%) always add to 100%

- **Lock Symbol**—When a lock is open the slider it guards may change freely, either because it was moved or because another slider was moved and this one must change to compensate.

When a lock is closed the slider it guards may not change. If two locks are closed and one is open, none of the sliders may be changed.

- **Half-life**—Use this type-in field to specify the half-life for usage. Usage will be decayed each scheduling interval in a way, that any particular contribution to accumulated usage will have half the value after a duration of half-life.
- **Days/Hours selection menu**—Select whether half-life is measured in days or hours.

- **Compensation Factor**—This type-in field accepts a positive integer-valued compensation factor. Reasonable values are in the range [2 ... 10].

The compensation factor prevents a user/project whose actual usage is far below its targeted usage from dominating resources when it first gets them (see explanation above).

About the Special User, `default`

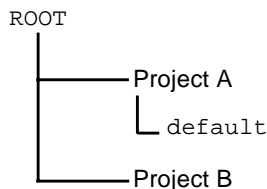
The user `default` can be used to reduce the amount of share-tree maintenance for sites with many users. It is applicable only for so called “hybrid” share-trees, where users are subordinated under Sun Grid Engine, Enterprise Edition projects in the share tree, and for cases where the same share entitlement is assigned to most users under the same project (equal share scheduling).

The user `default` can appear only as a leaf node under a project node in the share tree (where the project node refers to an existing Sun Grid Engine, Enterprise Edition project). If it is present, then it is interpreted as a shortcut for configuring all existing Sun Grid Engine, Enterprise Edition user entries underneath the corresponding project node while giving them the same share amount. Each user who has access to the project and submits jobs to it receives the same share entitlement configured for the corresponding `default` user entry. To activate the facility for a particular user you have to add this user to the list of Sun Grid Engine, Enterprise Edition system users.

Note that the users’ short-term entitlements will vary due to differences in the amount of resources they consume. Their long-term entitlements are the same, however.

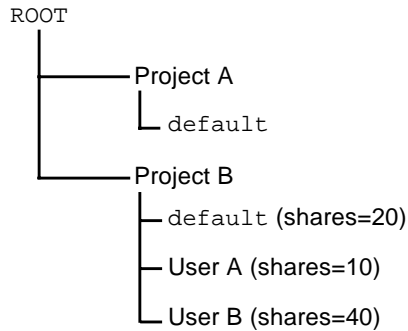
If you intend to assign special (lower or higher) rights to some users while maintaining the same long term entitlement for all other users then you can configure a share tree with individual user entries next to the `default` user for those users with special entitlements.

The following is Example A.



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

Contrast this with Example B.



In Example B, treatment for Project A is the same as it is in Example A. But for Project B, all users submitting to it receive equal long-term resource entitlements—except for User A, who receives half the entitlement of most other users, and User B, who receives twice the entitlement.

▼ How To Configure the Share-Based Policy from the Command Line

Note – Share tree configuration is recommended to be done via QMON because an hierarchical tree by its nature is well suited for graphical display and editing. However, if the need arises to integrate share tree modifications in shell scripts, for example, you can use the `qconf` command and its options.

- Use the `qconf` command, according to guidance in the following list.
 - The `qconf` options, `-astree`, `-mstree`, `-dstree`, and `-sstree`, provide the means to add an entire new share tree, to modify an existing share tree configuration, to delete a share tree, and to display the share tree configuration. Refer to the `qconf` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for details on these options. The `share_tree` manual page contains a description of the share tree configuration format.
 - The `-astnode`, `-mstnode`, `-dstnode`, and `-sstnode` options to `qconf` will not address the entire share tree, but only a single node. The node is referenced as path through all parent nodes down the share tree, similar to a directory path. The options allow you to add, modify, delete and display a node. The information contained in a node consists of its name and the attached shares.
 - The weighting of the usage parameters CPU, memory and I/O, the half-life and the compensation factor are contained in the scheduler configuration as `usage_weight_list`, `halftime`, and `compensation_factor`. The

scheduler configuration is accessible from the command line via the `-msconf` and the `-ssconf` options of `qconf`. Refer to the `sched_conf` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for details on the format.

About the Functional Policy

Functional scheduling, sometimes called priority scheduling, is a non-feedback scheme for determining a job's importance by its association with the submitting user, project, department and job class. The entitlement to system resources derived from the functional policy is combined with other entitlements (e.g., entitlement from a deadline or share-based policy) in determining a job's net entitlement.

The total number of tickets allotted to the functional policy determines the weight of functional scheduling among the four scheduling policies. The total number of functional tickets is divided among the functional categories of user, department, project, job, and job class by the administrator during Sun Grid Engine, Enterprise Edition installation.

Functional Shares

Functional shares are assigned to every member of the functional categories (user, department, project, job, and job class). These shares indicate what proportion of the tickets for a category each job associated with a member of the category is entitled to. If user `davidson` has 200 shares and user `donlee` has 100, a job submitted by `davidson` is entitled to twice the number of user-functional-tickets `donlee`'s job can get, no matter how many tickets that is.

The functional tickets allotted to each category are shared among all the jobs associated with a particular category.

The `share_functional_shares` Parameter

The functional policy defines entitlement shares for the categories user, project, department, job class (queue) and job and then shares for all members underneath each of these categories. The functional policy is thus similar to a two level share tree, but with the difference that a job can be associated with several of those categories at the same time. It belongs to a particular user, for instance, but also may belong to a project, a department and a job class.

However, as in the share tree, the share entitlement which a job receives from a functional category is determined by the shares defined for its corresponding category member (e.g., its project) and the shares given to the category (project vs.

user, department, and so on) as such. The `share_functional_shares` parameter (under `schedd_params` in cluster configuration) defines how the category member shares are used to determine the shares of a job. The shares assigned to the category members (e.g. a particular user or project) can be replicated for each job or they can be distributed among the jobs under the category member.

- `share_functional_shares=false` means replication.
- `share_functional_shares=true` defines distribution.

Those shares are comparable with stock shares. They do not have an effect for the jobs belonging to the same category member. All jobs under the same category member have the same amount of shares in both cases. But the share number has an effect when comparing the share amounts within the same category. Jobs with many siblings belonging to the same category member receive relatively small share portions if `share_functional_shares` is set to `true`. This is not the case if `share_functional_shares` is `false` and, thus, all sibling jobs have the same share amount as their category member.

Use `share_functional_shares=true` if you want a category member to receive a constant functional entitlement level for the sum of all its jobs independently of how many there are in the system. The entitlement of the individual job may get negligibly small, however, if it has many siblings. Use `share_functional_shares=false` to give each job the same entitlement level based on its category member's entitlement no matter how many siblings are in the system. Yet note that a category member with many jobs underneath may dominate the functional policy.

Be aware that the setting of share functional shares does not determine how many functional tickets in total are distributed. The total amount is always as defined by the administrator for the functional policy ticket pool. The share functional shares parameter just influences how functional tickets are distributed within the functional policy.

▼ How To Configure the Functional Share Policy From QMON

1. At the bottom of the QMON Ticket Overview dialogue box, click Functional Policy. The Functional Policy dialogue box, similar to the example in FIGURE 9-14, is presented.

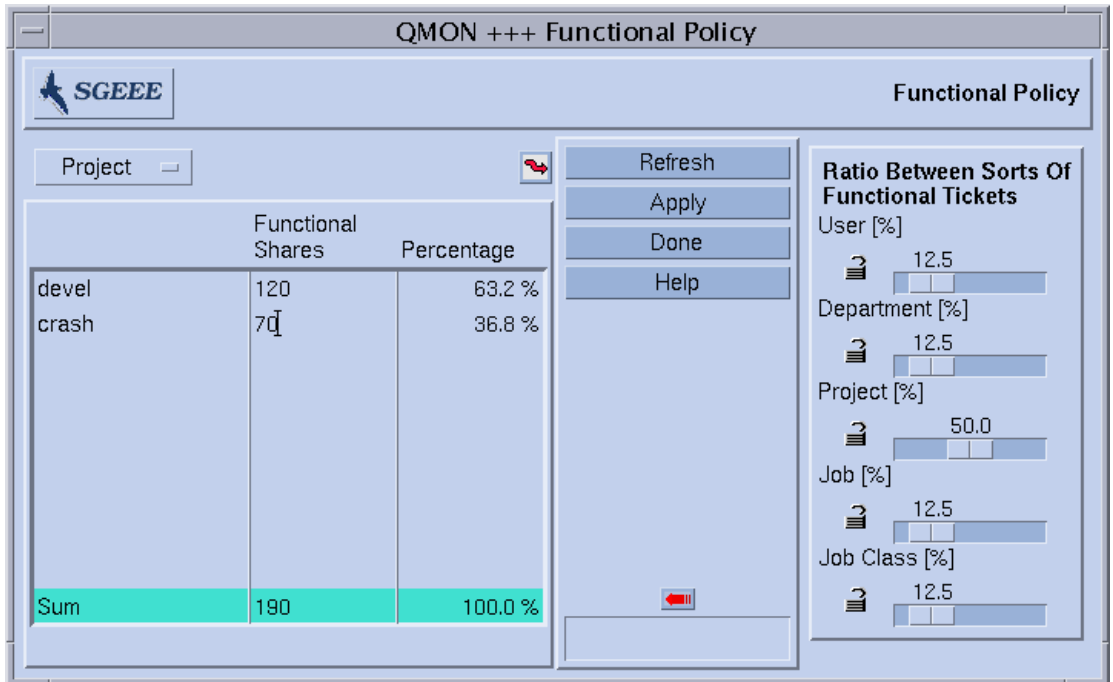


FIGURE 9-14 Functional Policy Dialogue Box

2. Proceed according to guidance in the following sections.

Functional Selection Menu

Select the category for which you are defining functional shares: user, project, department, job, or job class (defined by a queue).

Functional Display

This scrollable region shows the following.

- A list of the members of the category (user, project, department, job, or job class) for which you are defining functional shares.
- The number of functional shares for each member of the category. Shares are used as a convenient indication of the relative importance of each member of the functional category. *This field is editable.*
- The percentage of the functional share allocation for this category of functional ticket (user, Userset, etc.) that this number of functional shares represents. This field is a feedback device and is not editable.

Jagged Arrow Navigator

Single-click on this arrow to open a configuration dialogue box.

- For User functional shares, the User Configuration dialogue opens. You may use the User tab to switch to the appropriate mode for changing the configuration of Sun Grid Engine, Enterprise Edition users.
- For Department functional shares, again the User Configuration dialogue opens. You may use the Userset tab to switch to the appropriate mode for changing the configuration of departments represented as Sun Grid Engine, Enterprise Edition usersets.
- For Project functional shares, the Project Configuration dialogue opens.
- For Job functional shares, the Job Control dialogue opens.
- For Job Class functional shares, the Queue Control dialogue opens.

Refresh

The graphical user interface periodically updates the information which it displays. This button forces an immediate refresh of the display.

Apply

Clicking this button applies all the additions, deletions and modifications you have made but keeps the window open.

Done

Clicking this button closes the window. Changes will not be applied.

Help

Clicking this button opens on-line help.

Large Arrow Navigator

Single-click on this arrow to open the `Ratio Between Sorts of Functional Tickets` portion of this window.

Ratio Between Sorts of Functional Tickets

`User(%)`, `Department(%)`, `Project(%)`, `Job(%)` and `Job Class (%)` always add to 100%.

User (%) slider

This slider's setting indicates what percentage of the total functional tickets are to be allocated to the users category. When you change this slider, the other unlocked sliders change to compensate for the change in User percentage.

Departments (%) slider

This slider's setting indicates what percentage of the total functional tickets are to be allocated to the departments category. When you change this slider, the other unlocked sliders change to compensate for the change in Department percentage.

Project (%) slider

This slider's setting indicates what percentage of the total functional tickets are to be allocated to the projects category. When you change this slider, the other unlocked sliders change to compensate for the change in Project percentage.

Job (%) slider

This slider's setting indicates what percentage of the total functional tickets are to be allocated to the jobs category. When you change this slider, the other unlocked sliders change to compensate for the change in Job percentage.

Job Class (%) slider

This slider's setting indicates what percentage of the total functional tickets are to be allocated to the job class category. When you change this slider, the other unlocked sliders change to compensate for the change in Job Class percentage.

Lock Symbol

When a lock is open the slider it guards may change freely, either because it was moved or because another slider was moved and this one must change to compensate.

When a lock is closed the slider it guards may not change.

If four locks are closed and one is open, none of the sliders may be changed.

▼ How To Configure the Functional Share Policy from the Command Line

- **Use the `qconf` command and its options, according to guidance in the following list.**
 - For the user category via the `qconf -muser` command, modifying the `fshare` parameter (see the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for details on the `user` file format).
 - For the department category via the `qconf -mu` command, modifying the `fshare` parameter (see the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for details on the `access_list` file format which is used to represent departments).
 - For the project category via the `qconf -mprj` command, modifying the `fshare` parameter (see the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for details on the `project` file format).
 - For the job class category via the `qconf -mq` command, modifying the `fshare` parameter (see the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for details on the `queue` file format which is used to represent job classes).
 - The weighting between the different categories is defined in the scheduler configuration `sched_conf` and can be changed via `qconf -msconf`. The parameters to change are `weight_user`, `weight_department`, `weight_project`, `weight_job`, and `weight_jobclass`. The parameter values range between 0 and 1 and need to add up to 1.

Note – Functional shares can be assigned to jobs *only* via QMON. No command-line interface is available for this function.

About the Deadline Policy

Deadline scheduling ensures that a job is completed by a certain time by starting it soon enough and giving it enough resources to finish on time. The submitter specifies the following about the job.

- **Start time**—This is the time at which the job becomes eligible for execution. The start time is usually right after job submission, but can be delayed via the QMON Job Submission dialogue box parameter `Start At` or the `-a` option to `qsub` (see Chapter 4, “Submitting Jobs” on page 91 for details).
- **Initiation deadline**—This is the time at which the job reaches its highest importance, getting all of its potential deadline tickets thereby gaining its largest potential share of system resources. The user submitting the job has to determine whether the deadline initiation time is suitable for the job to meet its deadline.

Deadline Tickets

Sun Grid Engine, Enterprise Edition may exploit available system resources by starting deadline jobs, at a low level of importance, ahead of their initiation deadline. Jobs with a deadline receive additional tickets automatically as they approach their initiation deadlines. Deadline tickets are given to deadline jobs linearly from the time the job is eligible for execution until the initiation deadline is reached. If more than one deadline job reaches its initiation deadline, the deadline tickets are distributed proportionally to all the jobs based on their initiation deadlines.

The `share_deadline_tickets` Parameter

The administrator assigns a certain number of tickets to the deadline policy. This ticket number determines the amount of tickets assigned to each deadline job together with the job’s relative position between submission and deadline initiation time. The `share_deadline_tickets` parameter (under `schedd_params` in cluster configuration) is a third influence factor in the calculation of the deadline tickets for the deadline jobs.

A setting of `share_deadline_tickets=true` means that the total amount of tickets assigned to the deadline policy is distributed across all deadline jobs and then the portion for each job is reduced corresponding to where it stands in its

approach to its deadline initiation time. A `share_deadline_tickets=false` setting means that each deadline job will get the full ticket amount assigned to the deadline policy as it reaches its deadline initiation time and proportionally less as it approaches it.

Use `share_deadline_tickets=true` if you want to control the total ticket amount distributed by the deadline policy, especially in relation to the share-based and functional policy, which only have a fixed ticket amount to distribute. Note that the ticket amounts assigned to an individual job can get too small for reaching a deadline if too many deadline jobs are in the system at the same time.

Use `share_deadline_tickets=false` to control the importance of individual deadline jobs relative to the ticket pools available for the other policies. With this setting it doesn't matter how many deadline jobs are in the system. The jobs always can get up to the maximum deadline ticket amount. With many deadline jobs in the system, however, other policies may lose importance.

Deadline Tickets Configuration

The system administrator sets the maximum number of deadline tickets available to all deadline jobs. This number indicates the weight of deadline scheduling among the four policies. Configure by way of the Ticket Overview screen (FIGURE 9-12), which also shows the current number of deadline tickets active in the system.

Deadlineusers Configuration

The policy regarding users who are permitted to submit deadline jobs is also under the control of the cluster administration. Only users who are part of the user access list, “deadlineusers,” are granted deadline tickets. Figure FIGURE 9-15 shows the Initiation Deadline section of the Deadline Job Submission dialogue box.

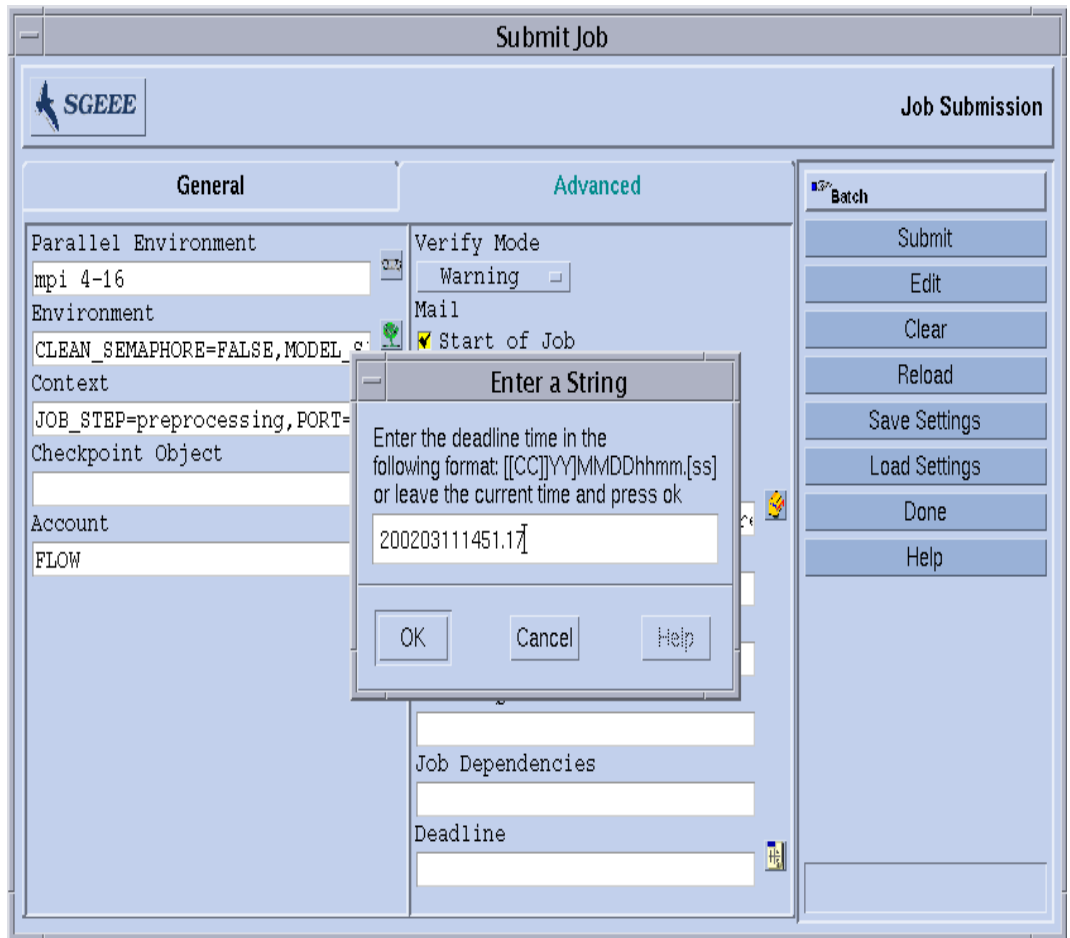


FIGURE 9-15 Deadline Job Submission Dialogue Box

From the command line, you can pass the initiation deadline to the Sun Grid Engine, Enterprise Edition system via the `-dl` option to `qsub`. See Chapter 4 for details on how to submit deadline jobs.

About the Override Policy

Override scheduling allows a Sun Grid Engine, Enterprise Edition manager or operator to dynamically adjust the relative importance of an individual job or of all the jobs associated with a user, department, project, or job class by adding tickets to that job, user, department, project, or job class. Adding override tickets increases the total number of tickets, thus the overall share of resources, that a user, department, project, job class, or job has.

Adding override tickets also increases the total number of tickets in the system. These additional tickets “deflate” the value of every job’s tickets.

Override tickets are mainly intend to be used for two purposes.

- To temporarily override the automated ticket assignment policies—share-based, functional, and deadline—without a need to change the configuration of these policies
- To establish resource entitlement levels with an associated fixed amount of tickets. This is appropriate for scenarios like high/medium/low job or priority classes

Override tickets assigned directly to a job simply go away when the job finishes, and all other tickets are “inflated” back to their original value. Override tickets assigned to users, departments, projects, and job classes remain in the system until they are explicitly removed by the administrator.

The Ticket Overview screen (FIGURE 9-12) shows the current number of override tickets active in the system.

Note – Override entries remain in the Override dialogue box and can influence subsequent work if they are not explicitly deleted by the operator when they are no longer needed.

The `share_override_tickets` Parameter

The administrator assigns tickets to the different members of the override categories; i.e., to the different users, projects, departments, job classes (queues) or jobs. Except for the “jobs” category, this means that the ticket value assigned to individual jobs under a particular category member is determined by the ticket amount defined for the corresponding member. So the number of tickets given to user A, for example, determines how many tickets are assigned to all jobs of user A.

The `share_override_tickets` parameter (under `schedd_params` in cluster configuration) controls how job ticket values are derived from their category member ticket value. A setting of `share_override_tickets=true` means that the tickets of the category members are distributed evenly among the jobs under this

member. A setting of `share_override_tickets=false` means that each job inherits the ticket amount defined for its category member; that is, the category member tickets are replicated for all jobs underneath.

Use `share_override_tickets=true` if you want to control the total ticket amount distributed by the override policy, especially in relation to the share-based and functional policy, which only have a fixed ticket amount to distribute. Note that the ticket amounts assigned to an individual job can get negligibly small if a lot of jobs are under one category member (e.g., belong to a certain user) and if `share_override_tickets` is set to `true`.

Use `share_override_tickets=false` to control the importance of individual jobs relative to the ticket pools available for the other policies and override categories. With this setting it doesn't matter how many jobs are under a category member. The jobs always get the same ticket amount, but the total number of override tickets in the system increases the more jobs with a right to receive override tickets are in the system. So other policies may lose importance in such a scenario.

▼ How To Configure the Override Policy

1. From the Ticket Overview dialogue box, click **Override Policy**.

The Override Policy dialogue box, which is similar to the example in FIGURE 9-16, is presented.

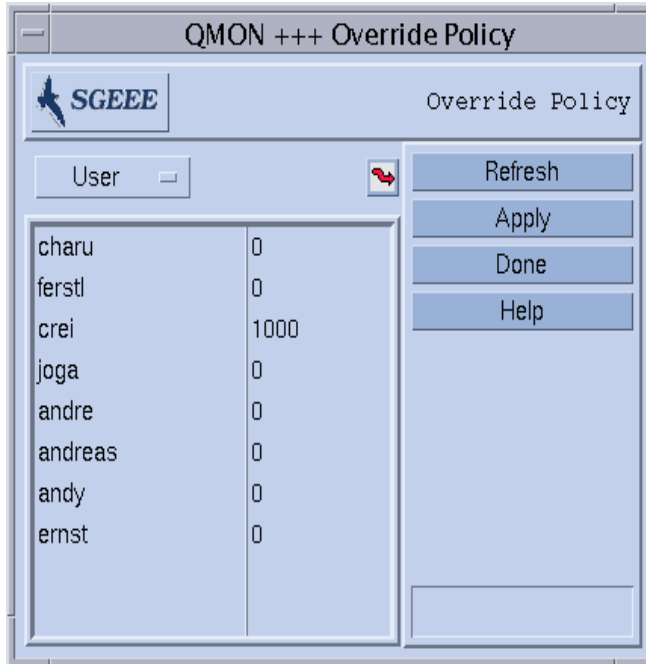


FIGURE 9-16 Override Policy Dialogue Box

2. Assign override tickets to jobs, users, departments, projects, or job classes, according to guidance in the following sections.

Override Selection Menu

Select the sort of entity for which you are defining override tickets: user, project, department, job, or job class.

Override Display

This scrollable region shows the following.

- A list of the members of the entity (user, project, department, job, or job class) for which you are defining tickets

- The integer number of override tickets for each member of the entity. This field is editable.

Jagged Arrow Navigator

Single-click on this arrow to open a configuration dialogue box.

- For User override tickets, the User Configuration dialogue box opens. You may use the User tab to switch to the appropriate mode for changing the configuration of Sun Grid Engine, Enterprise Edition users.
- For Department override tickets, again the User Configuration dialogue box opens. You may use the Userset tab to switch to the appropriate mode for changing the configuration of departments represented as Sun Grid Engine, Enterprise Edition usersets.
- For Project override tickets, the Project Configuration dialogue box opens.
- For Job override tickets, the Job Control dialogue box opens.
- For Job Class override tickets, the Queue Control dialogue box opens.

Refresh

The graphical user interface periodically updates the information which it displays. This button forces an immediate refresh of the display.

Apply

Clicking this button applies all the additions, deletions and modifications you have made but keeps the window open.

Done

Clicking this button closes the window without applying the additions, deletions and modifications you have made.

Help

Clicking this button opens on-line help.

▼ How To Configure the Override Policy from the Command Line

- Proceed according to guidance in the following list.
 - **For the user category, via the `qconf -muser` command**—modifies the `oticket` parameter (see the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for details on the user file format).
 - **For the department category, via the `qconf -mu` command**—modifies the `oticket` parameter (see the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for details on the `access_list` file format which is used to represent departments).
 - **For the project category, via the `qconf -mprj` command**—modifies the `oticket` parameter (see the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for details on the project file format).
 - **For the job class category, via the `qconf -mq` command**—modifies the `oticket` parameter (see the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual* for details on the queue file format which is used to represent job classes).

Note – Override tickets can be assigned to jobs *only* via QMON. No command line interface is available for this function at present.

About Policy Hierarchy

Policy hierarchy provides the means to resolve certain cases of conflicting policies for pending jobs in particular. Those cases can occur in combination with the share-based and functional policies. Both policies share the characteristic that jobs belonging to the same *leaf-level* entities are ordered first-come-first-served with respect to assigning priorities (share entitlements) to them. Leaf-level entities means user/project leaves in the share-tree, or any of the “members” of a functional category (a particular user, project, department or queue) in the functional policy, except for the “jobs” category. So, for instance, the first job of the same user gets the most, the second gets the next most, the third next, and so on.

A conflict can occur if another policy mandates an order which is different. So, for instance, the override policy might define that the third job is the most important one, while the first one submitted should come last.

A policy hierarchy putting the override policy in front of the share tree or functional policy will make sure that jobs most important to the override policy also will get the most entitlements in the share/functional policy as long as those jobs belong to the same leaf level share-tree entity (user or project).

The `policy_hierarchy` parameter (which is found under `schedd_params` in Cluster Configuration) can be a up to four-letter combination of the first letters of the four policies `S`(hare-based), `F`(unctional), `D`(eadline) and `O`(verride). This way you establish a chain of policies with the first letter defining the top policy and the last letter the bottom of the hierarchy. Policies that are not listed in the policy hierarchy do not influence the hierarchy. They still may be a source for tickets of jobs though. Those tickets just do not influence the ticket calculations in other policies. Still, all tickets of all policies are added up for each job to define its overall entitlement.

The following are examples of two settings and a description of how they influence the order of the pending jobs.

```
policy_hierarchy=OS
```

- First, the override policy assigns the appropriate number of tickets to each pending job.
- This number of tickets is then used to influence the entitlement assignment in the share tree in case two jobs belong to the same user or to the same leaf level project.. Then the share tree tickets are calculated for the pending jobs.
- The tickets from the override policy and the share tree policy are added up together with all other active policies not in the hierarchy. The job with the highest resulting number of tickets has the highest entitlement.

```
policy_hierarchy=OS
```

- The deadline tickets for all pending deadline jobs are calculated.
- Then the override policy assigns the appropriate number of tickets to each pending job and the tickets from the deadline and override policy are added up.
- The resulting ticket values influence the entitlement assignment in the functional policy in case two jobs belong to the same functional category member. Based on this, the functional tickets are calculated for the pending jobs. The resulting value is added to the ticket amount from the deadline and override policy.
- These ticket values then influence the entitlement assignment in the share tree in case two jobs belong to the same user or to the same leaf level project. The corresponding share tree tickets are calculated for the pending jobs and are added to the previous sum from the deadline, override, and functional policies.
- The job with the highest resulting number of tickets has the highest entitlement.

Again, all combinations of the four letters are feasible, but only a subset are meaningful or have practical relevance. The last letter always should be a S or F because those are the only two policies that can be influenced due to their characteristics described in the examples. If D and O are next to each other, they can be interchanged without changing the behavior.

More generally, the following form is recommended for `policy_hierarchy` settings.

`[O|D][O|D][S|F][S|F]`

So, if present, then the policies which only can influence (the deadline and the override policy) should occur in the first or second letter only, while the last or last two letters should represent policies which can be influenced (share-based and functional).

A setting such as OFD is perfectly valid, but is equivalent to OF. Settings such as OFDS are also valid and have a somewhat different result than ODFS, for instance, but examples that demand an OFDS as opposed to an ODFS setting appear to be quite contrived.

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, Enterprise Edition 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, Enterprise Edition 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 is 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

CODE EXAMPLE 9-1 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/      *                *                /
```

CODE EXAMPLE 9-1 Example of Path-Aliasing File

About Configuring Default Requests

Batch jobs are normally assigned to queues by the Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition 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 9-2.

```
# 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 9-2 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, Enterprise Edition 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, Enterprise Edition 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 finish.
- **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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition 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 9-17, is presented.

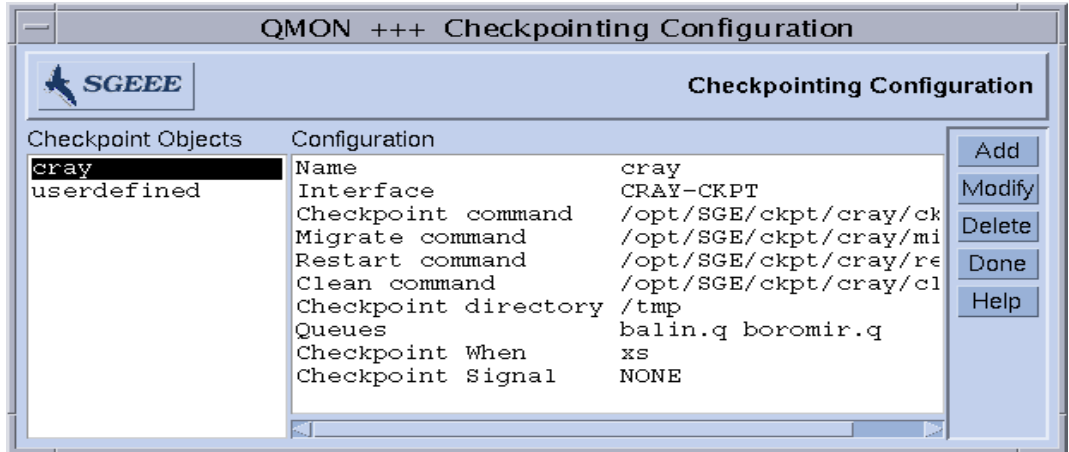


FIGURE 9-17 Checkpointing Configuration Dialogue Box

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.

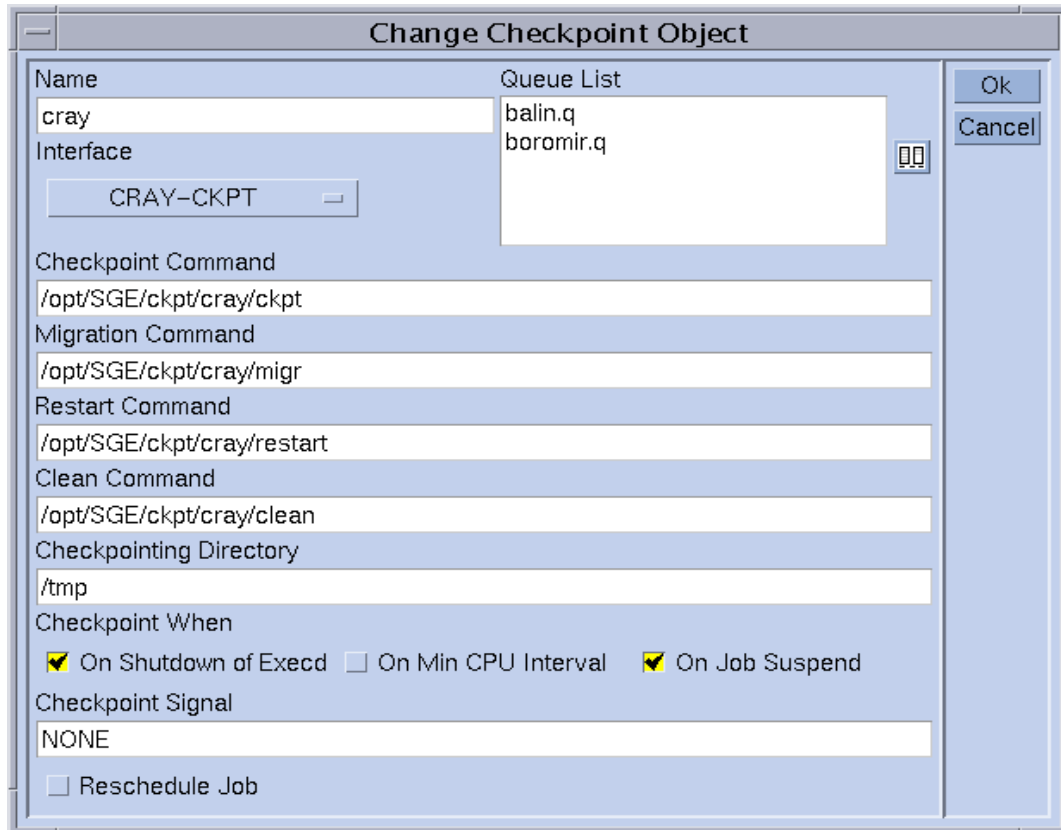
Delete Configured Checkpointing Environments

- 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 9-18, is presented, along with the current configuration of the selected checkpointing environment.



The image shows a graphical user interface window titled "Change Checkpoint Object". The window is divided into several sections. At the top, there are two labels: "Name" and "Queue List". Below "Name" is a text field containing "cray". Below "Queue List" is a list box containing "balin.q" and "boromir.q". To the right of the list box is a small icon of two overlapping windows. Below the "Name" label is a label "Interface" and a button labeled "CRAY-CKPT". Below the "Queue List" label is a label "Checkpoint Command" and a text field containing "/opt/SGE/ckpt/cray/ckpt". Below that is a label "Migration Command" and a text field containing "/opt/SGE/ckpt/cray/migr". Below that is a label "Restart Command" and a text field containing "/opt/SGE/ckpt/cray/restart". Below that is a label "Clean Command" and a text field containing "/opt/SGE/ckpt/cray/clean". Below that is a label "Checkpointing Directory" and a text field containing "/tmp". Below that is a label "Checkpoint When" and three checkboxes: "On Shutdown of Execd" (checked), "On Min CPU Interval" (unchecked), and "On Job Suspend" (checked). Below that is a label "Checkpoint Signal" and a text field containing "NONE". At the bottom is a checkbox labeled "Reschedule Job" which is unchecked. On the right side of the window are two buttons: "Ok" and "Cancel".

FIGURE 9-18 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, Enterprise Edition 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 9-18).

The Select Queues dialogue box, which is similar to the example in FIGURE 9-19, is presented.

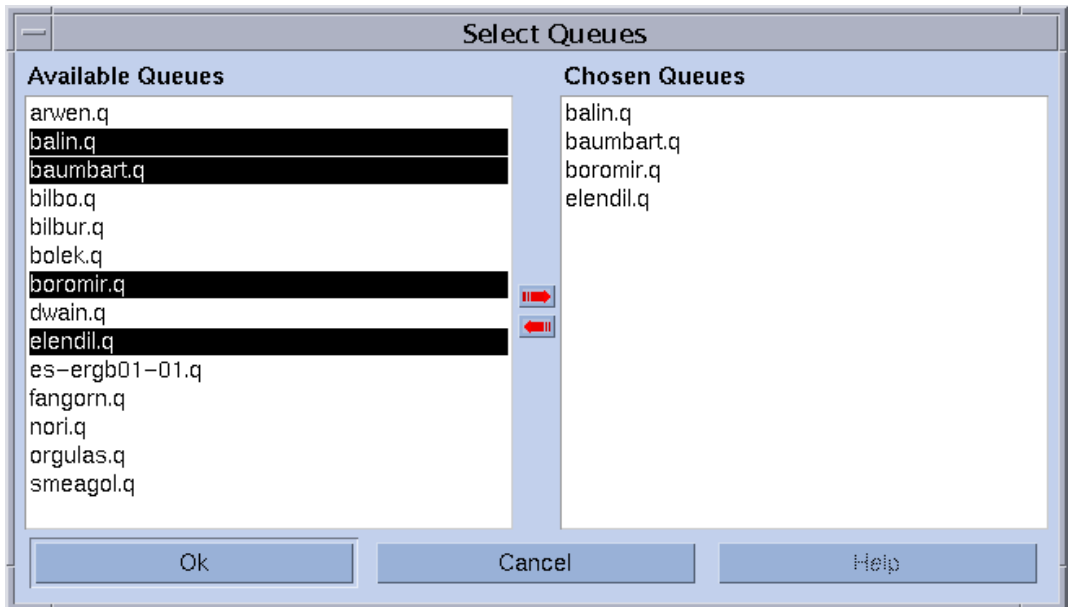


FIGURE 9-19 Checkpointing Queue Selection Dialogue Box

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 9-18, 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`

Add 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 -Ackpt filename`

Add checkpointing environment from file—This command parses the specified file—which must have the checkpointing environment configuration template format—and adds the new checkpointing environment configuration.

- `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 -mckpt filename`

Modify checkpointing environment from file—This command parses the specified file—which must have the checkpointing environment configuration template format—and modifies the existing checkpointing environment configuration.

- `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.

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.

- “How To Configure PEs with QMON” on page 314
 - “Display the Contents of a PE” on page 315
 - “Delete a PE” on page 315
 - “Modify a PE” on page 315
 - “Add a PE” on page 316
 - “How To Configure PEs from the Command Line” on page 319
 - “How To Display Configured PE Interfaces from the Command Line” on page 320
 - “How To Display Configured PE Interfaces with QMON” on page 320
-

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, Enterprise Edition system provides a flexible and powerful interface that satisfies the various needs.

The Sun Grid Engine, Enterprise Edition system 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 PE interfaces may be configured concurrently at the same time.

Arbitrary PEs can be interfaced by Sun Grid Engine, Enterprise Edition as long as suitable startup and stop procedures are provided as described in the section, "The PE Startup Procedure" on page 322 and in the section, "Termination of the PE" on page 324, 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.

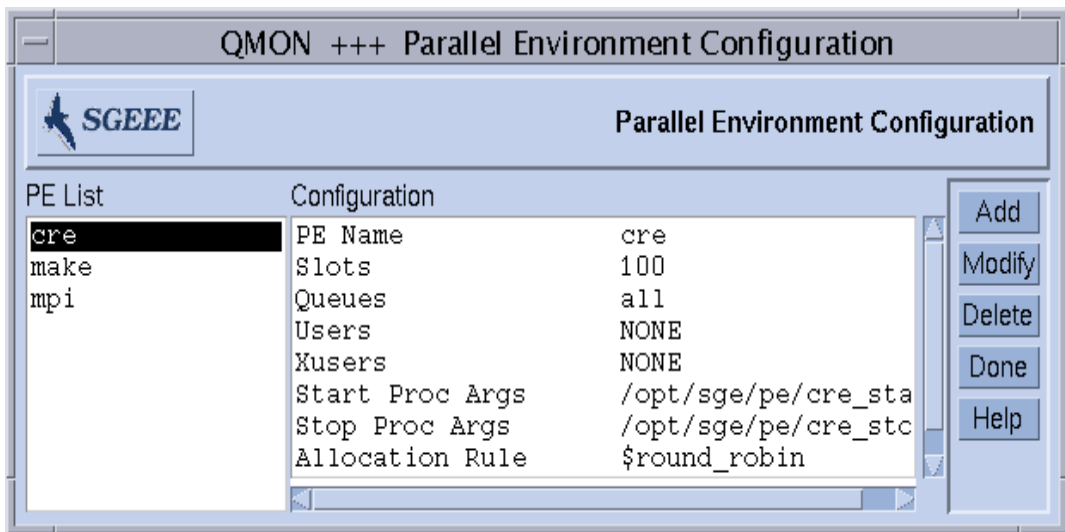


FIGURE 10-1 Parallel Environment Configuration Dialogue Box

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 316.**
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.

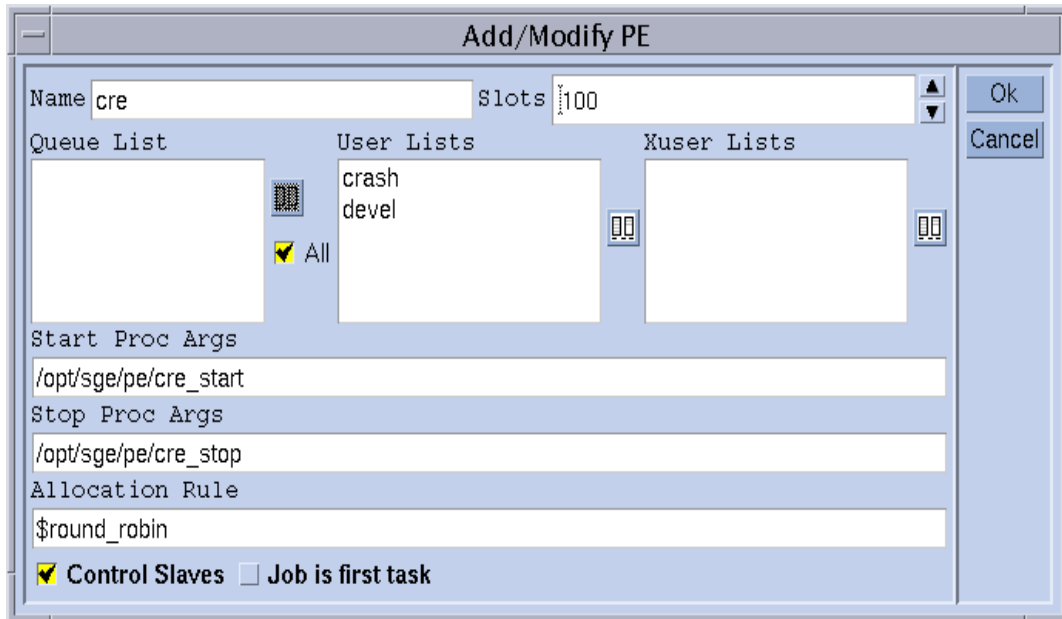


FIGURE 10-2 Parallel Environment Definition Dialogue Box

2. Add the PE definitions according to guidance in the section, “Explanation of the Parallel Environment Definition Parameters” on page 316.

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. (Alternatively, you can use the All checkbox to specify any parallel queue being used by the PE.)

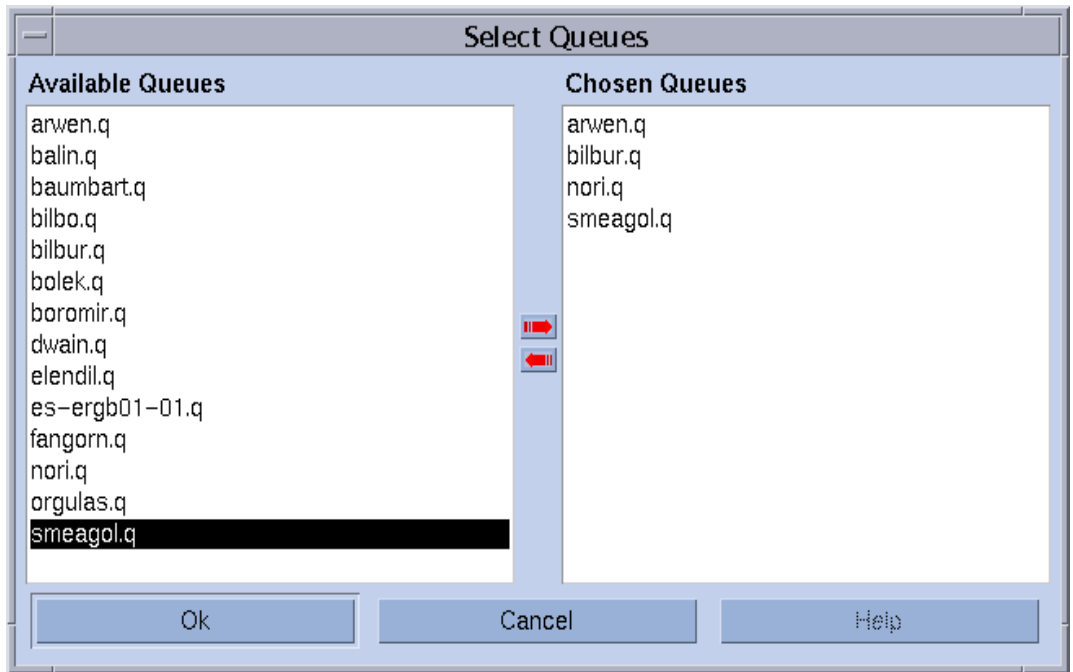


FIGURE 10-3 Select Queues Dialogue Box

- The *User Lists* display region contains the user access lists (see the section, “About User Access Permissions” on page 250) which 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.

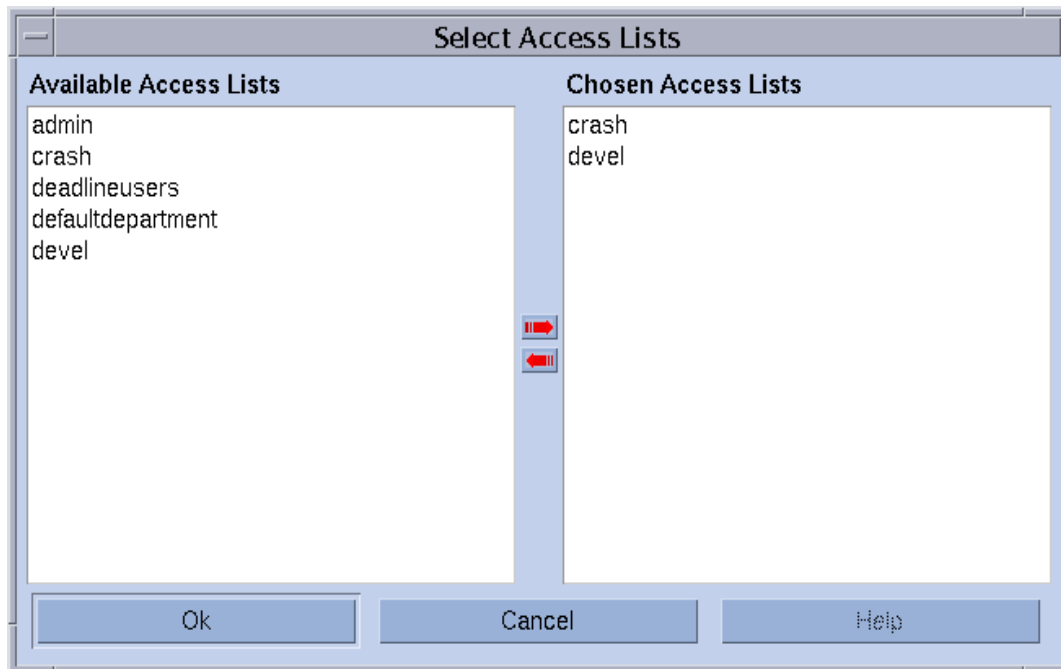


FIGURE 10-4 Select Access Lists Dialogue Box

- 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 322 and “Termination of the PE” on page 324 respectively). Note that specifying these parameters is optional. If no such procedures are required for a certain parallel environment, you can leave the fields empty.

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, Enterprise Edition internal run-time information to the procedures. The *sgc_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 that is used by a PE. A positive integer fixes the number of processes for each suitable host, the special denominator *\$pe_slots* can be used to cause the full range of processes of a job to be allocated on a single host (SMP), and the denominators *\$fill_up* and *\$round_robin* can be used to cause unbalanced distributions of processes at each host.

For more details on these allocation rules, see the `sge_pe` entry in the *Sun Grid Engine 5.3 and Sun Grid Engine, Enterprise Edition 5.3 Reference Manual*.

- The *Control Slaves* toggle button declares whether parallel tasks are generated via Sun Grid Engine, Enterprise Edition (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, Enterprise Edition 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, Enterprise Edition. Refer to the section, “Tight Integration of PEs and Sun Grid Engine, Enterprise Edition Software” on page 324 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 -Ap filename`

Add parallel environment from file – This command parses the specified file, `filename`—which must have the PE configuration template format—and adds the new PE configuration.

- `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 -Mp filename`

Modify parallel environment from file – This command parses the specified file, *filename*—which must have the PE configuration template format—and modifies the existing PE configuration.

- `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 314).

The example from the section, “Advanced Example” on page 105 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.

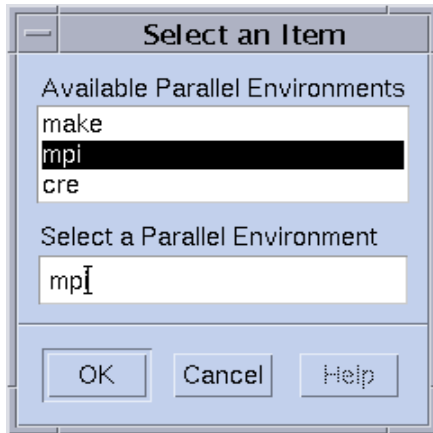


FIGURE 10-5 PE Selection

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 117 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, Enterprise Edition 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 110 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=ssf 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, Enterprise Edition PE interface facility is highly configurable. In particular, the Sun Grid Engine, Enterprise Edition 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. If you are unsure, ask the Sun Grid Engine, Enterprise Edition administrator if you are required to export certain environment variables.

The PE Startup Procedure

The Sun Grid Engine, Enterprise Edition 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, Enterprise Edition system. An example for such a startup procedure for the PVM environment is contained within the Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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 314. The parameters are among those provided to PE startup and stop scripts by Sun Grid Engine, Enterprise Edition during runtime. The required host file, as an example, is generated by Sun Grid Engine, Enterprise Edition 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 queue on which parallel processes are to be run.
- The first entry of each line specifies the host name of the queue.
- The second entry specifies the number of parallel processes to be run in this queue.
- The third entry denotes the queue.
- The fourth entry denotes a processor range to be used in case of a multiprocessor machine.

This file format is generated by Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition software reports 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, Enterprise Edition—to remove all errors that may be hard to trace if the procedure is integrated into the Sun Grid Engine, Enterprise Edition 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 314).

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, Enterprise Edition system may have no information about the processes running under PE control and thus cannot clean up. Sun Grid Engine, Enterprise Edition software, of course, cleans up the processes directly associated with the job script that it has launched.

The Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition—to remove all errors that may be hard to trace if the procedure is integrated into the Sun Grid Engine, Enterprise Edition framework.

Tight Integration of PEs and Sun Grid Engine, Enterprise Edition Software

The explanation of the Control Slaves parameter in the section, “How To Configure PEs with QMON” on page 314 mentions that PEs for which the creation of parallel tasks is performed by the Sun Grid Engine, Enterprise Edition 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, Enterprise Edition. To overcome this problem, the Sun Grid Engine, Enterprise Edition 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, Enterprise Edition software.

The Sun Grid Engine, Enterprise Edition 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 also contain README files describing the usage and any current restrictions. Refer to those README files for further detail.

In addition, for the purpose of comparison, the `<sge_root>/mpi/sunhpc/loose-integration` directory contains a *loose* integration sample with Sun HPC ClusterTools™ software, and the `<sge_root>/mpi` directory contain a *loosely* integrated variant of the interfaces for comparison.

Note – Performing a tight integration with a PE is an advanced task and may require expert knowledge of the PE and the Sun Grid Engine, Enterprise Edition PE interface. You may want to contact your Sun support representative distributor for assistance.

Error Messaging

This chapter describes Sun Grid Engine, Enterprise Edition error messaging procedures. Use this information to help you troubleshoot any problems you may encounter with the software.

How Sun Grid Engine, Enterprise Edition Software Retrieves Error Reports

Sun Grid Engine, Enterprise Edition 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_execds`. 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 46 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, Enterprise Edition 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, E for error (an error condition has been detected) or C for critical (may lead to a program abort).

You can use the `loglevel` parameter in the cluster configuration to bias on a global or local basis what message types you want to be logged.

- The fifth is the message text.

Note – If, for some reason, an error logfile is not accessible, Sun Grid Engine, Enterprise Edition 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 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, Enterprise Edition notifies users and/or administrators about error events via e-mail. The mail messages sent by Sun Grid Engine, Enterprise Edition 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, Enterprise Edition 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 successful; it did not happen.
Migrate	0	Success
	Rest	Success—For kernel checkpoint, however, special meaning: Checkpoint not successful; 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, Enterprise Edition 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, Enterprise Edition 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, Enterprise Edition distribution. For `csh` or `tcsh` users the file `<sg_e_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 `cs`h or `tc`sh 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, Enterprise Edition 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, Enterprise Edition 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.
