Altair®

# PBS Pro™

# Administrator Guide 5.4

---

for UNIX®, Linux, and Windows®

# Portable Batch System™ Administrator Guide

PBS-3BA01: Altair® PBS Pro™ 5.4, Updated: February 11, 2004

Edited by: James Patton Jones

Altair Grid Technologies is a subsidiary of Altair Engineering, Inc. For more information, and for product sales and technical support, contact Altair at:

URL: www.altair.com www.pbspro.com
Email: sales@pbspro.com, support@pbspro.com

| Location | Telephone | e-mail |
|---|---|---|
| North America | +1 248 614 2425 | pbssupport@altair.com |
| China | +86 (0)21 5393 0011 | support@altair.com.cn |
| France | +33 (0)1 4133 0990 | francesupport@altair.com |
| Germany | +49 (0)7031 6208 22 | support@altair.de |
| India | +91 80 658 8540 +91 80 658 8542 | support@altair-eng.soft.net |
| Italy | +39 0832 315573 +39 800 905595 | support@altairtorino.it |
| Japan | +81 3 5396 1341 | aj-support@altairjp.co.jp |
| Korea | +82 31 728 8600 | support@altair.co.kr |
| Scandinavia | +46 (0)46 286 2050 | support@altair.se |
| United Kingdom | +44 (0)1327 810 700 | support@uk.altair.com |

For online documentation purchases, visit: store.pbspro.com

# Table of Contents

# Acknowledgements

PBS Pro is the enhanced commercial version of the PBS software originally developed for NASA. The NASA version had a number of corporate and individual contributors over the years, for which the PBS developers and PBS community are most grateful. Below we provide formal legal acknowledgements to corporate and government entities, then special thanks to individuals.

The NASA version of PBS contained software developed by NASA Ames Research Center, Lawrence Livermore National Laboratory, and MRJ Technology Solutions. In addition, it included software developed by the NetBSD Foundation, Inc., and its contributors, as well as software developed by the University of California, Berkeley and its contributors.

Other contributors to the NASA version of PBS include Bruce Kelly and Clark Streeter of NERSC; Kent Crispin and Terry Heidelberg of LLNL; John Kochmar and Rob Pennington of *Pittsburgh Supercomputing Center*; and Dirk Grunwald of *University of Colorado, Boulder*. The ports of PBS to the Cray T3e and the IBM SP SMP were funded by *DoD USAERDC*, Major Shared Research Center; the port of PBS to the Cray SV1 was funded by DoD MSIC.

No list of acknowledgements for PBS would possibly be complete without special recognition of the first two beta test sites. Thomas Milliman of the *Space Sciences Center* of the *University of New Hampshire* was the first beta tester. Wendy Lin of *Purdue University* was the second beta tester and holds the honor of submitting more problem reports than anyone else outside of NASA.

# Preface

## Intended Audience

This document provides the system administrator with the information required to install, configure, and manage the Portable Batch System (PBS). PBS is a workload management system that provides a unified batch queuing and job management interface to a set of computing resources.

## Related Documents

The following publications contain information that may also be useful in the management and administration of PBS.

PBS-3QS01 **PBS Pro Quick Start Guide**: Provides a quick overview of PBS Pro installation and license key generation.

PBS-3BU01 **PBS Pro User Guide**: Provides an overview of PBS Pro and serves as an introduction to the software, explaining how to use the user commands and graphical user interface to submit, monitor, track, delete, and manipulate jobs.

PBS-3BE01 **PBS Pro External Reference Specification**: Discusses in detail the PBS application programming interface (API), security within PBS, and intra-daemon communication.

## Ordering Software and Publications

To order additional copies of this manual and other PBS publications, or to purchase additional software licenses, contact your reseller or the PBS Products Department. Contact information is included on the copyright page of this document.

## Document Conventions

PBS documentation uses the following typographic conventions.

abbreviation — If a PBS command can be abbreviated (such as subcommands to `qmgr`) the shortest acceptable abbreviation is underlined.

`command` — This fixed width font is used to denote literal commands, filenames, error messages, and program output.

**input** — Literal user input is shown in this bold, fixed-width font.

`manpage`(x) — Following UNIX tradition, manual page references include the corresponding section number in parentheses appended to the manual page name.

*terms* — Words or terms being defined, as well as variable names, are in italics.

# Chapter 1
# Introduction

This book, the **Administrator Guide** to the Portable Batch System, Professional Edition (PBS Pro) is intended as your knowledgeable companion to the PBS Pro software. This edition pertains to PBS Pro in general, with specific information for version 5.4.

## 1.1 Book Organization

This book is organized into 13 chapters, plus three appendices. Depending on your intended use of PBS, some chapters will be critical to you, and others can be safely skipped.

Chapter 1 **Introduction**: Gives an overview of this book, PBS, and the PBS team.

Chapter 2 **Concepts and Terms**: Discusses the components of PBS and how they interact, followed by definitions of terms used in PBS.

Chapter 3 **Pre-Installation Planning**: Helps the reader plan for a new installation of PBS.

Chapter 4 **Installation**: Covers the installation of the PBS Pro software and licenses.

Chapter 5 **Upgrading PBS Pro**: Provides important information for sites that are upgrading from a previous version of PBS.

Chapter 6 **Configuring the Server**: Describes how to configure the PBS Server, and set up queues and nodes.

Chapter 7 **Configuring MOM**: Describes how to configure the PBS MOM processes.

Chapter 8 **Configuring the Scheduler**: Describes how to configure the PBS Scheduler.

Chapter 9 **Customizing PBS Resources**: Describes how to configure custom resources and dynamic consumable resources.

Chapter 10 **Administration**: Discusses PBS day-to-day administration and and related activities.

Chapter 11 **Administrator Commands**: Describes all PBS commands intended to be used by the Administrator.

Chapter 12 **Example Configurations**: Provides examples and sample configurations.

Chapter 13 **Problem Solving**: Discusses trouble-shooting, and describes the tools provided by PBS to assist with problem solving.

Appendix A **Error Codes**: Provides a listing and description of the PBS error codes.

Appendix B **Request Codes**: Provides a listing and description of the PBS request codes.

Appendix C **File Listing**: Lists directories and files installed by this release of PBS Pro, with owner, permissions, and average size.

## 1.2 What is PBS Pro?

PBS Pro is the professional version of the Portable Batch System (PBS), a flexible resource and workload management system, originally developed to manage aerospace computing resources at NASA. PBS has since become the leader in supercomputer workload management and the *de facto* standard on Linux clusters.

Today, growing enterprises often support hundreds of users running thousands of jobs across different types of machines in different geographical locations. In this distributed heterogeneous environment, it can be extremely difficult for administrators to collect detailed, accurate usage data or to set system-wide resource priorities. As a result, many computing resource are left under-utilized, while others are over-utilized. At the same time, users are confronted with an ever expanding array of operating systems and platforms. Each year, scientists, engineers, designers, and analysts must waste countless hours learning the nuances of different computing environments, rather than being able to focus on their core priorities. PBS Pro addresses these problems for computing-intensive enterprises such as science, engineering, finance, and entertainment.

Now you can use the power of PBS Pro to better control your computing resources. This product enables you to unlock the potential in the valuable assets you already have. By reducing dependency on system administrators and operators, you will free them to focus on other actives. PBS Pro can also help you to efficiently manage growth by tracking real usage levels across your systems and by enhancing effective utilization of future purchases.

### 1.2.1 History of PBS

In the past, UNIX systems were used in a completely interactive manner. Background jobs were just processes with their input disconnected from the terminal. However, as UNIX moved onto larger and larger processors, the need to be able to schedule tasks based on available resources increased in importance. The advent of networked compute servers, smaller general systems, and workstations led to the requirement of a networked batch scheduling capability. The first such UNIX-based system was the Network Queueing System (NQS) funded by NASA Ames Research Center in 1986. NQS quickly became the *de facto* standard for batch queueing.

Over time, distributed parallel systems began to emerge, and NQS was inadequate to handle the complex scheduling requirements presented by such systems. In addition, computer system managers wanted greater control over their compute resources, and users wanted a single interface to the systems. In the early 1990’s NASA needed a solution to this problem, but found nothing on the market that adequately addressed their needs. So NASA led an international effort to gather requirements for a next-generation resource management system. The requirements and functional specification were later adopted as an IEEE POSIX standard (1003.2d). Next, NASA funded the development of a new resource management system compliant with the standard. Thus the Portable Batch System (PBS) was born.

PBS was quickly adopted on distributed parallel systems and replaced NQS on traditional supercomputers and server systems. Eventually the entire industry evolved toward distributed parallel systems, taking the form of both special purpose and commodity clusters. Managers of such systems found that the capabilities of PBS mapped well onto cluster computers. The PBS story continued when Veridian (the R&D contractor that developed PBS for NASA) released the Portable Batch System Professional Edition (PBS Pro), a commercial, enterprise-ready, workload management solution. Three years later, the Veridian PBS Products business unit was acquired by Altair Engineering, Inc. Altair set up the PBS Products unit as a subsidiary company named Altair Grid Technologies focused on PBS Pro and related Grid software.

## 1.3 About the PBS Team

The PBS Pro product is being developed by the same team that originally designed PBS for NASA. In addition to the core engineering team, Altair Grid Technologies includes individuals who have supported PBS on computers all around the world, including some of the largest supercomputers in existence. The staff includes internationally-recognized experts in resource- and job-scheduling, supercomputer optimization, message-passing programming, parallel computation, and distributed high-performance computing. In addition, the PBS team includes co-architects of the NASA Metacenter (the first full-production geographically distributed meta-computing environment), co-architects of the Department of Defense MetaQueueing (prototype Grid) Project, co-architects of the NASA Information Power Grid, and co-chair of the Global Grid Forum's Scheduling Group.

## 1.4 About Altair Engineering

Through engineering, consulting and high performance computing technologies, Altair Engineering increases innovation for more than 1,500 clients around the globe. Founded in 1985, Altair's unparalleled knowledge and expertise in product development and manufacturing extend throughout North America, Europe and Asia. Altair specializes in the development of high-end, open CAE software solutions for modeling, visualization, optimization and process automation.

# Chapter 2
# Concepts and Terms

PBS is a distributed workload management system. As such, PBS handles the management and monitoring of the computational workload on a set of one or more computers. Modern workload/resource management solutions like PBS include the features of traditional batch queueing but offer greater flexibility and control than first generation batch systems (such as the original batch system NQS).

Workload management systems have three primary roles:

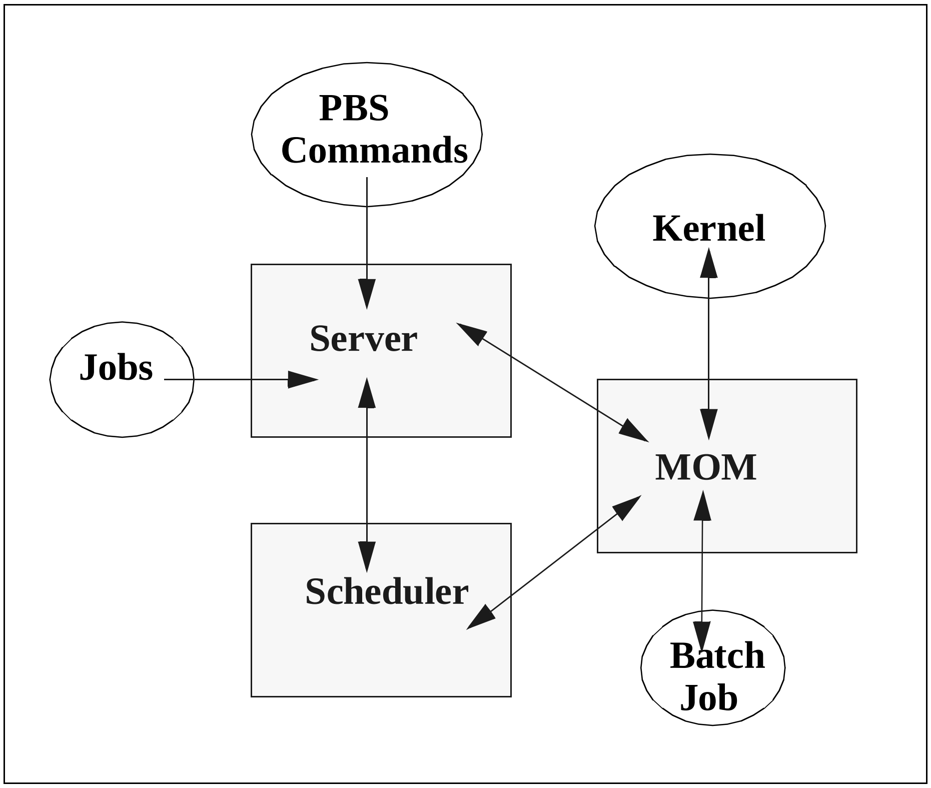
Queuing — The collecting together of work or tasks to be run on a computer. Users submit tasks or “jobs” to the resource management system where they are queued up until the system is ready to run them.

Scheduling — The process of selecting which jobs to run when and where, according to a predetermined policy. Sites balance competing needs and goals on the system(s) to maximize efficient use of resources (both computer time and people time).

Monitoring — The act of tracking and reserving system resources and enforcing usage policy. This covers both user-level and system-level monitoring as well as monitoring of the scheduling algorithms to see how well they are meeting the stated goals

## 2.1 PBS Components

PBS consist of two major component types: system daemons and user-level commands. A brief description of each is given here to help you make decisions during the installation process.



**Job Server** The *Job Server* daemon process is the central focus for PBS. Within this document, it is generally referred to as *the Server* or by the execution name *pbs_server.* All commands and daemons communicate with the Server via an *Internet Protocol* (IP) network. The Server's main function is to provide the basic batch services such as receiving/creating a batch job, modifying the job, protecting the job against system crashes, and running the job. Typically there is one Server managing a given set of resources.

**Job Executor (MOM)** The *Job Executor* is the daemon that actually places the job into execution. This daemon, *pbs_mom,* is informally called

*MOM* as it is the mother of all executing jobs. (MOM is a reverse-engineered acronym that stands for Machine Oriented Miniserver.) MOM places a job into execution when it receives a copy of the job from a Server. MOM creates a new session that is as identical to a user login session as is possible. For example, if the user's login shell is `csh`, then MOM creates a session in which `.login` is run as well as `.cshrc`. MOM also has the responsibility for returning the job's output to the user when directed to do so by the Server. One MOM daemon runs on each computer which will execute PBS jobs.

A special version of MOM, called the *Globus MOM*, is available if it is enabled during the installation of PBS. It handles submission of jobs to the Globus environment. Globus is a software infrastructure that integrates geographically distributed computational and information resources. Globus is discussed in more detail in the "Globus Support" section of the **PBS Pro User Guide**.

**Job Scheduler**

The *Job Scheduler* daemon, *pbs_sched*, implements the site's policy controlling when each job is run and on which resources. The Scheduler communicates with the various MOMs to query the state of system resources and with the Server to learn about the availability of jobs to execute. The interface to the Server is through the same API as used by the client commands. Note that the Scheduler communicates with the Server with the same privilege as the PBS Manager.

**Commands**

PBS supplies both command line programs that are POSIX 1003.2d conforming and a graphical interface. These are used to submit, monitor, modify, and delete jobs. These *client commands* can be installed on any system type supported by PBS and do not require the local presence of any of the other components of PBS.

There are three classifications of commands: user commands (which any authorized user can use), operator commands, and manager (or administrator) commands. Operator and Manager commands require specific access privileges, as discussed in section 10.7 "External Security" on page 178.

## 2.2 Defining PBS Terms

The following section defines important terms and concepts of PBS. The reader should review these definitions before beginning the planning process prior to installation of PBS. The terms are defined in an order that best allows the definitions to build on previous terms.

**Node** A *node* to PBS is a computer system with a single *operating system* (OS) image, a unified virtual memory space, one or more CPUs and one or more IP addresses. Frequently, the term *execution host* is used for node. A computer such as the SGI Origin 3000, which contains multiple CPUs running under a single OS, is one node. Systems like the IBM SP and Linux clusters, which contain separate computational units each with their own OS, are collections of nodes. Nodes can be defined as either *cluster nodes* or *time-shared nodes*, as discussed below.

**Cluster Node** A node whose purpose is geared toward running multi-node or parallel jobs is called a *cluster node*. If a cluster node has more than one virtual processor, the VPs may be assigned to different jobs (*job-shared*) or used to satisfy the requirements of a single job (*exclusive*). This ability to temporally allocate the entire node to the exclusive use of a single job is important for some multi-node parallel applications. Note that PBS enforces a one-to-one allocation scheme of cluster node VPs ensuring that the VPs are not over-allocated or over-subscribed between multiple jobs. (See also *node* and *virtual processors*.)

**time-shared Node** In contrast to cluster nodes are hosts that **always** service multiple jobs simultaneously, called *timeshared nodes*. Often the term *host* rather than node is used in conjunction with time-shared, as in *timeshared host.* A timeshared node will never be allocated exclusively or temporarily-shared. However, unlike cluster nodes, a timeshared node **can** be over-committed if the local policy specifies to do so. (See also *virtual processors*.)

**Cluster** This is any collection of nodes controlled by a single instance of PBS (i.e., by one PBS Server).

**Exclusive VP** An exclusive VP is one that is used by one and only one job at a time. A set of VPs is assigned exclusively to a job for the duration of that job. This is typically done to improve the performance of message-passing programs.

**Temporarily-shared VP** A *temporarily-shared node* is one where one or more of its VPs are temporarily shared by jobs. If several jobs request multiple temporarily-shared nodes, some VPs may be allocated commonly to both jobs and some may be unique to one of the jobs. When a VP is allocated on a temporarily-shared basis, it remains so until all jobs using it are terminated. Then the VP may be re-allocated, either again for temporarily-shared use or for exclusive use.

If a host is defined as time-shared, it will never be allocated exclusively or temporarily-shared.

**Load Balance** A policy wherein jobs are distributed across multiple time-shared hosts to even out the workload on each host. Being a policy, the distribution of jobs across execution hosts is solely a function of the Job Scheduler.

**Queue** A *queue* is a named container for jobs within a Server. There are two types of queues defined by PBS, *routing* and *execution*. A *routing queue* is a queue used to move jobs to other queues including those that exist on different PBS Servers. Routing queues are similar to the old NQS pipe queues. A job must reside in an *execution queue* to be eligible to run and remains in an execution queue during the time it is running. In spite of the name, jobs in a queue need not be processed in queue order (first-come first-served or *FIFO*).

**Node Attribute** Nodes have attributes associated with them that provide control information. The attributes defined for nodes are: state, type (ntype), the list of jobs to which the node is allocated, properties, max_running, max_user_run, max_group_run, and both assigned and available resources (“resources_assigned” and “resources_available”).

**Node Property** A set of zero or more *properties* may be given to each node in order to have a means of grouping nodes for allocation. The property is nothing more than a string of alphanumeric characters (first character must be alphabetic) without meaning to PBS. The PBS Administrator may assign to nodes whatever property names desired. Your choices for property names should be relayed to the users.

**Portable Batch System** PBS consists of one Job Server (pbs_server), one or more Job Scheduler (pbs_sched), and one or more execution servers (pbs_mom). The PBS System can be set up to distribute the workload to one large time-shared system, multiple time shared systems, a cluster of nodes to be used exclusively or temporarily-shared, or any combination of these.

The remainder of this chapter provides additional terms, listed in alphabetical order.

**Account** An *account* is arbitrary character string, which may have meaning to one or more hosts in the batch system. Frequently, account is used as a grouping for charging for the use of resources.

**Administrator** See Manager.

**API** PBS provides an *Application Programming Interface (API)* which is used by the commands to communicate with the Server. This API is described in the **PBS Pro External Reference Specification**. A site may make use of the API to implement new commands if so desired.

**Attribute** An *attribute* is an inherent characteristic of a parent object (Server, queue, job, or node). Typically, this is a data item whose value affects the operation or behavior of the object and can be set by the owner of the object. For example, the user can supply values for attributes of a job.

**Batch or Batch Processing** This refers to the capability of running jobs outside of the interactive login environment.

**Complex** A *complex* is a collection of hosts managed by one batch system. It may be made up of nodes that are allocated to only one job at a time or of nodes that have many jobs executing at once on each node or a combination of these two scenarios.

**Destination** This is the location within PBS where a job is sent for processing. A destination may uniquely define a single queue at a single Server or it may map into many locations.

**Destination Identifier** This is a string that names the destination. It is composed two parts and has the format queue@server where server is the name of a PBS Server and queue is the string identifying a queue on that Server.

**File Staging** *File staging* is the movement of files between a specified location and the execution host. See “Stage In” and “Stage Out” below.

**Group ID (GID)** This numeric identifier is uniquely assigned to each group (see Group).

**Group** *Group* refers to collection of system users (see Users). A user must be a member of a group and may be a member of more than one. Within UNIX and POSIX systems, membership in a group establishes one level of privilege. Group membership is also often used to control or limit access to system resources.

**Hold** An artificial restriction which prevents a job from being selected for processing. There are three types of holds. One is applied by the job owner, another is applied by a PBS *Operator*, and a third is applied by the system itself or the PBS *Manager*.

**Job or Batch Job** The basic execution object managed by the batch subsystem. A job is a collection of related processes which is managed as a whole. A job can often be thought of as a shell script running in a POSIX session. (A session is a process group the member processes cannot leave.) A non-singleton job consists of multiple tasks of which each is a POSIX session. One *task* will run the job shell script.

**Manager** The *manager* is the person authorized to use all restricted capabilities of PBS. The PBS Manager may act upon the Server, queues, or jobs. The Manager is also called the Administrator.

**Operator** A person authorized to use some but not all of the restricted capabilities of PBS is an *operator*.

**Owner** The owner is the user who submitted the job to PBS.

**POSIX** Refers to the various standards developed by the “Technical Committee on Operating Systems and Application Environments of the IEEE Computer Society” under standard P1003.

**Rerunable** If a PBS job can be terminated and its execution restarted from the beginning without harmful side effects, the job is rerunable.

**Stage In** This process refers to moving a file or files to the execution host prior to the PBS job beginning execution.

**Stage Out** This process refers to moving a file or files off of the execution host after the PBS job completes execution.

**User** Each system *user* is identified by a unique character string (the user name) and by a unique number (the user id).

**Task** *Task* is a POSIX session started by MOM on behalf of a job.

**User ID (UID)** Privilege to access system resources and services is typically established by the *user id*, which is a numeric identifier uniquely assigned to each user (see User).

**Virtual Processor (VP)** A node may be declared to consist of one or more *virtual processors (VPs)*. The term virtual is used because the number of VPs declared does not have to equal the number of real processors (CPUs) on the physical node. The default number of virtual processors on a node is the number of currently functioning physical processors; the PBS Manager can change the number of VPs as required by local policy. (See also *cluster node* and *timeshared node*.)

## Chapter 3

# Pre-Installation Planning

This chapter presents information needed prior to installing PBS. First, a reference to new features in this release of PBS Pro is provided. Next is the information necessary to make certain planning decisions.

## 3.1 New Features in PBS Pro 5.4

The *Release Notes* included with this release of PBS Pro lists all new features in this version of PBS Pro, information on upgrading, and any warning or caveats. Be sure to review the Release Notes, as they may contain information that was not available when this book went to press. The following is a list of major new features.

**Administrator Guide** Support for Server Redundancy and Failover. See section 6.15 “Configuring PBS Redundancy and Failover” on page 84.

**Administrator Guide** Support for automatic requeue and rescheduling of jobs from failed nodes. See the Server parameter “node_fail_requeue” on page 57.

**Administrator Guide** New external action on job termination, enables application-level checkpointing. See section 7.5 “Site-specific Job Termination Action” on page 107.

**Administrator Guide** New feature to allow a site to provide a local program to be used for job checkpoint and restart, enables site-specific system-level checkpointing. See section 7.6 “Site-Specific Job Checkpoint / Restart” on page 109.

**Administrator Guide** New node grouping feature for better allocation for multi-node jobs. See section 6.8 “Node Grouping” on page 77.

**Administrator Guide** New option to control how multi-node jobs are handled on cycle-harvesting systems. See section 7.7.3 “Cycle Harvesting and Multi-node Jobs” on page 114.

**Administrator Guide** New command to print post-installation diagnostic information. See section 11.3 “The pbs_probe Command” on page 198.

**Administrator Guide** Job exit status now passed to epilogue. See section 10.11.1 “Prologue and Epilogue Arguments” on page 181.

**Administrator Guide** New per resource group limit (hard and soft) for jobs. See discussion of `max_group_res` in section 6.3 “Hard versus Soft Limits” on page 52.

**Administrator Guide** Support for SGI “cpusets” on Altix systems. See section 7.10 “Enhanced SGI “cpusets” Support” on page 119.

**Administrator Guide** New appendix added to this manual providing a directory and file listing for this release of PBS Pro. See Appendix C.

**User Guide** New “blocking qsub” option. (See “Requesting qsub Wait for Job Completion” on page 94 in the **PBS Pro User Guide**.)

**User Guide** New option to control output file permissions. (See “Changing Job umask” on page 94 in the **PBS Pro User Guide**.)

**User Guide** Features for using PBS within a DCE environment. (See “Running PBS in a DCE Environment” on page 114 in the **PBS Pro User Guide**.)

## 3.2 Changes to Time-shared and Cluster Nodes

For planning purposes, note that this release of PBS Pro has reduced the differences between time-shared and cluster nodes to:

1. Time-shared nodes are first choice for jobs that do not have a node specification.
2. Time-shared nodes may not be requested for exclusive use with the `#excl` suffix.
3. More processes than CPUs can be run on time-shared nodes but not on cluster nodes.
4. If load balancing by "load average" is activated in the Job Scheduler, it applies only to time-shared nodes.
5. Allocation of cluster nodes remains based on the number of (virtual) processors.
6. Cluster nodes do not support node specifications.

## 3.3 Planning

PBS is able to support a wide range of configurations. It may be installed and used to control jobs on a single system or to load balance jobs on a number of systems. It may be used to allocate nodes of a cluster or parallel system to both parallel and serial jobs. It can also deal with a mix of these situations. While this chapter gives a quick overview of different configurations for planning purposes, you may wish to read Chapter 12 Example Configurations, prior to installing PBS Pro. You should also review the Glossary of terms prior to continuing with the installation and configuration of PBS Pro. (See also “Concepts and Terms” on page 5.)

**Important:** When planning the most appropriate setup for PBS Pro in your environment, keep in mind the differences discussed in section 3.2 above. Also remember that when jobs request entire nodes (i.e. via `qsub -l nodes=xyz` requests), they will not have resource limits (such as `cput` and `ncpus`) enforced upon them.

### 3.3.1 Planning for File Access

In distributed environments it will be necessary to plan for how the users will access their input files, datasets, etc. Various options exist (such as NFS, rcp, scp, etc). These need to be considered prior to installing PBS Pro, as such decisions can change which parameters are selected for tuning PBS. For details, see the MOM configuration parameter “$usecp” on page 103 and section 11.4 “The pbs_rcp vs. pbs_scp Command” on page 198. The impact of file location and delivery are discussed further in Chapter 8 of the **PBS Pro User Guide** under the heading “Delivery of Output Files”.

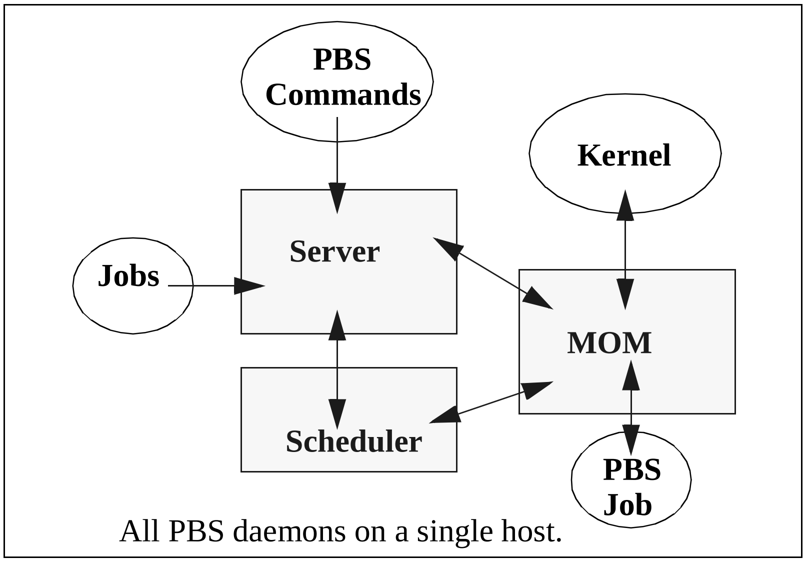
### 3.3.2 SGI “cpusets” Requires ProPack Library

Customers who intend to run PBS Pro on SGI Altix systems using “cpusets” should note that there are strict requirements for SGI ProPack (containing the “cpuset” API). ProPack 2.2 or later is required. The library is required on MOM nodes where “cpuset” functionality is desired. To test if the library is currently installed, execute the following command:

```
# ls /usr/lib/libcpuset.so*
```

## 3.4 Single Execution System

If PBS is to be installed on a time-sharing system, all three daemons would normally be installed on that same system. During installation (as discussed in the next chapter) be sure to select option 1 (all components) from the PBS Installation tool.



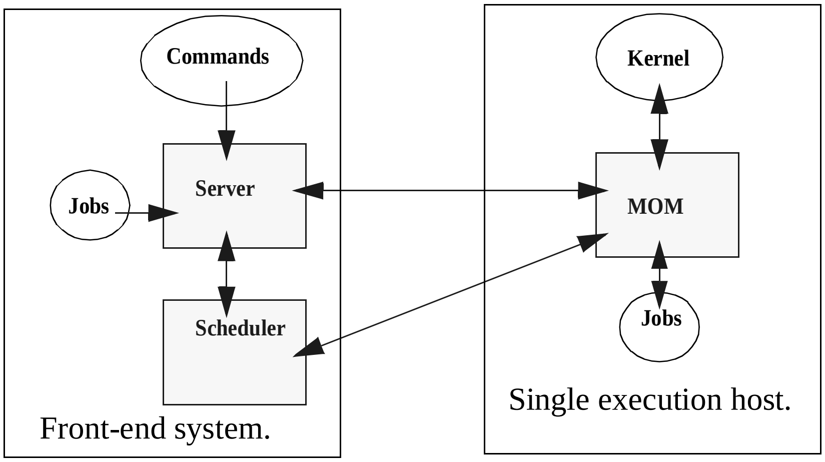
All PBS daemons on a single host.

### 3.4.1 Large SMP systems

Large SMP systems (e.g. SGI Origin or Cray vector systems) are best treated as “time-shared” systems in order to maximize the scheduling efficiency on these systems.

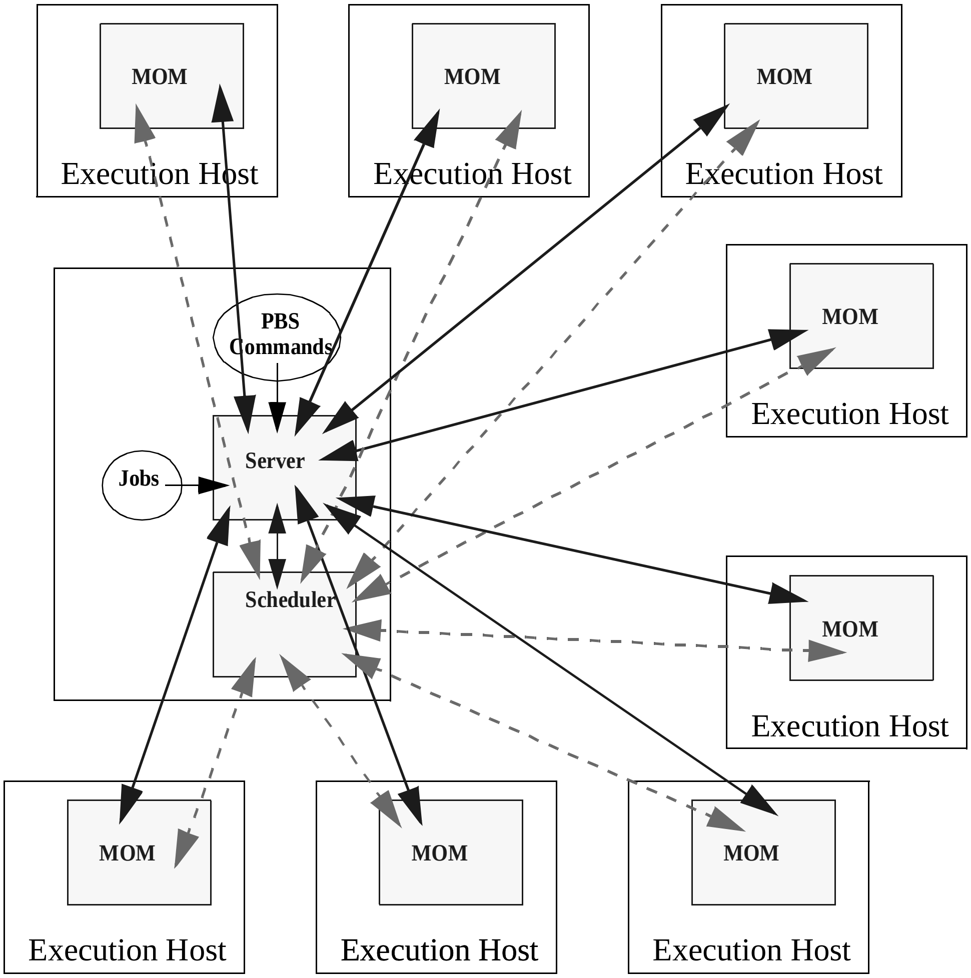
### 3.4.2 Single Execution System with Front-end

If you wish, the PBS Server and Scheduler (`pbs_server` and `pbs_sched)`can run on one system and jobs can execute on another.



## 3.5 Multiple Execution Systems

If PBS is to be installed on a collection (or cluster) of time-sharing systems, normally the Server (`pbs_server`) and the Scheduler (`pbs_sched`) is installed on a “front end” system (option 1 from the PBS Installation tool), and a MOM (`pbs_mom`) is installed (option 2 from the Installation tool) and run on each execution node (i.e. each system where jobs are to be executed). The following diagram illustrates this for an eight node cluster.



## 3.6 PBS Pro as a Front End for Globus

If support for submitting jobs to Globus is enabled, then a separate `pbs_mom_globus` must be run. In order to forward a job to a Globus site, `pbs_mom_globus` will read the user's grid credential on the host where it is running. Thus, `pbs_mom_globus` should run on the host from which users submit their PBS Globus jobs. Globus-specific configuration information is given in the following sections of this manual:

“Server Support for Globus” on page 83
“MOM Globus Configuration” on page 124
“Manually Starting Globus MOM” on page 167
“Globus Support” in the **PBS Pro User Guide**.

# Chapter 4
# Installation

This chapter discusses the installation procedures for PBS Pro. It is recommended that you read Chapter 3: Planning prior to beginning the installation.

**Important:** Be sure to read the Release Notes included in the PBS Pro distribution, as it will contain any information that was unavailable when this book went to press. The Release Notes also contain a detailed description of all new features in a given release.

## 4.1 Overview

The PBS software can be installed from the PBS CD-ROM or downloaded from the Customer Login Area of the PBS website (http://www.pbspro.com). The installation procedure is slightly different depending on the distribution source. However, the basic steps of PBS installation are:

| | |
|---|---|
| Step 1 | Prepare distribution media |
| Step 2 | Extract and install the software |
| Step 3 | Acquire a PBS license |
| Step 4 | Install the license |

## 4.2 Default Install Options

The installation program installs the various PBS components into specific locations on the system. The installation program allows you to override these default locations if you wish. (Note that some operating systems' software installation programs do not permit software relocation, and thus you are not able to override the defaults on those systems.) The locations are written to the `pbs.conf` file created by the installation process. For details see the description of "pbs.conf" on page 159.

## 4.3 Installation on UNIX/Linux Systems

This section describes the installation process for PBS Pro on UNIX or Linux systems.

### 4.3.1 Media Setup

CD-ROM: If installing from the PBS CD-ROM, insert the PBS CD into the system CD-ROM drive, mount the CD-ROM device (if needed), then `cd` to the distribution directory.

```
# mount /cdrom
# cd /cdrom/PBSPro_5.4.0
```

Download: If not installing from CD-ROM, follow these instructions:

Step 1 Download the distribution file from the PBS website. (Follow the instructions you received with your order confirmation.)
Step 2 Move the distribution file to `/tmp` on the system on which you intend to install PBS,
Step 3 Uncompress and extract the distribution file
Step 4 Then `cd` to the distribution directory

```
# cd /tmp
# gunzip /tmp/pbspro_5.4.0-arch.tar.gz
# tar -xvf /tmp/pbspro_5.4.0-arch.tar
# cd PBSPro_5.4.0
```

### 4.3.2 Installation Overview

For a given system, the PBS install script uses the native package installer provided with that system. This means that the PBS package should install into what is considered the “normal” location for third-party software. The following examples shows a typical installation under the Sun Solaris operating system. The process is very similar for other operating systems, but may vary depending on the native package installer on each system. Launch the installation process by executing the `INSTALL` command, as shown below.

```
# ./INSTALL
Installation of PBS

The following directory will be the root of the
installation. Several subdirectories will be created if
they don't already exist: bin, sbin, lib, man and include.
Execution directory? [/opt/pbs]

PBS needs to have a private directory (referred to as
"PBS_HOME" in the documentation) where it can permanently
store information.
Home directory? [/usr/spool/PBS]
/usr/spool/PBS does not exist, I'll make it...done

[ Description of the different configuration options ]

PBS Installation:
        1. Server, execution and commands
        2. Execution only
        3. Commands only
(1|2|3)?
```

Next, you need to decide what kind of PBS installation you want for each machine in your cluster. There are three possibilities: a Server node, an execution node, or a client host. If you are going to run PBS on a single timesharing host, install the full Server package (option 1). If you are going to have a cluster of machines, you need to pick one to be the front-end and install the Server package (option 1) there. Then, install the execution package (option 2) on all the compute nodes in the cluster. The client package (option 3) is for hosts which will not be used for execution but need to have access to PBS. It contains the commands, the GUIs and man pages. This gives the ability to submit jobs and check status of jobs as well as queues and multiple PBS Servers.

The following sections illustrate the differences between installation on a single server system versus a cluster of workstations.

### 4.3.3 Installation on a (stand-alone) System

For the following examples, we will assume that you are installing PBS on a single large server or execution host, on which all the PBS daemons will run, and from which users will submit jobs. Example of such a system might be an SMP system such as an SGI Origin3000 or a Cray T90.

To achieve this, we select option **1** to the question shown in the example above, followed by "all" when asked which packages to add, as shown:

```
(1|2|3)? 1

Installing PBS for a Server Host.
The following packages are available:
  1  pbs64     pbs64    (sparc) 5.0

Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]: all


## Processing package information.
## Processing system information.
## Verifying disk space requirements.
## Checking for conflicts with packages already installed.
## Checking for setuid/setgid programs.
```

**Important:** Some systems' installation program (e.g. Solaris `pkgadd`) will ask you to confirm that it is acceptable to install setuid/setgid programs as well as to run installation sub-programs as root. You should answer yes (or "**y**") to either of these questions, if asked.

Next, the installation program will proceed to extract and install the PBS package(s) that you selected above. The process should look similar to the example below.

```
Installing pbs64 as <pbs64>

## Installing part 1 of 1.
/etc/init.d/pbs
[ listing of files not shown for brevity ]

## Executing postinstall script.
*** PBS Installation Summary
***
*** The PBS Server has been installed in /opt/pbs/sbin.
***
*** The PBS commands have been installed in /opt/pbs/bin.
***
*** This host has the PBS Server installed, so
*** the PBS commands will use the local server.
*** The PBS command server host is mars
***
*** PBS MOM has been installed in /opt/pbs/sbin.
***
*** The PBS Scheduler has been installed in /opt/pbs/sbin.
***
Installation of <pbs64> was successful.
```

Finally, the installation program will request the license key for your system. Follow the instructions in section 4.4 “Installing the PBS License Key” on page 24 below.

### 4.3.4 Installing on a UNIX/Linux Cluster

A typical cluster of computers has a front-end system which (usually) manages the whole cluster. Most sites install the PBS Server and Scheduler on this front-end system, but not the MOM daemon (as most sites tend *not* to want to run batch-jobs on the front-end node). The MOM daemon is then installed on each computation node (execution host) within the cluster.

In either case, you will need to run the `INSTALL` program multiple times in order to install PBS Pro on your cluster system. (Alternatively, you could install one of the execution nodes, and then distribute the installation to other nodes via a program such as `rdist`.)

First, install PBS on the front-end node, following the instructions given in section 4.3.3 “Installation on a (stand-alone) System” on page 22. Enter “**`no`**” when asked if you want to start PBS. Then, if you do *not* want to run batch jobs on the front-end node, edit the newly installed `pbs.conf` file, setting `PBS_START_MOM=0`, indicating that you do not want a PBS MOM daemon started on this system.

Next, create the list of machines PBS will manage. Edit the file `PBS_HOME/server_priv/nodes`, adding one machine name per line for each execution machine in your cluster. Append "`:ts`" to all node names that you wish to be time-shared nodes. (Discussion of the different node types is given in section 6.6 "Nodes" on page 68.)

Lastly, start the PBS software by running the PBS startup script, the location for which varies on different systems. See "Starting and Stopping PBS: UNIX and Linux" on page 161.

Now that the PBS Server has been installed and started, you should install each compute node. Do this by running the `INSTALL` program on each compute node, selecting the execution package only (option **`2`**). When prompted if you wish to start PBS on that node, enter "**`yes`**".

### 4.3.5 Installing MOM with SGI "cpuset" Support

PBS Pro for SGI systems provides (site-selectable) support for IRIX and Altix "cpusets". A cpuset is a named region of the SGI system which contains a specific set of CPUs and associated memory. PBS has the ability to use the cpuset feature to "fence" PBS jobs into their own cpuset. This helps to prevent jobs from interfering with each other. To enable use of this feature, a different PBS MOM binary needs to be installed, as follows. Stop the MOM process, install the new MOM as shown below, then restart MOM. (See also section 10.2 "Starting and Stopping PBS: UNIX and Linux" on page 161.)

```
# cd /usr/pbs/sbin
# rm pbs_mom
# ln -s pbs_mom.cpuset pbs_mom
```

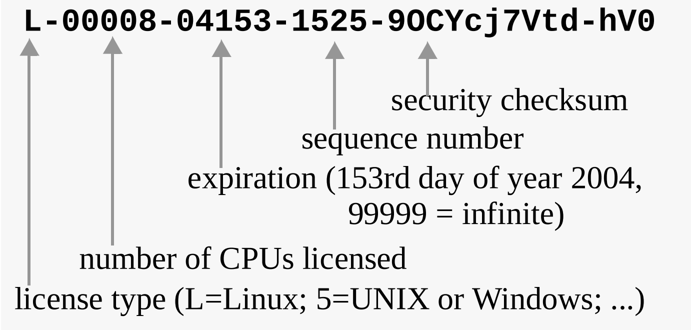
Additional information on configuring and using SGI cpusets is discussed later in this manual.

## 4.4 Installing the PBS License Key

The PBS license manager can handle multiple license keys in the PBS license file. This is useful when you expand your PBS configuration: you can simply add the additional licenses. This section describes adding a single license (such as following an initial installation of PBS Pro). The next section discusses adding multiple licenses. Optional “floating licenses” are discussed in section 4.4.3 “Using Floating Licenses” on page 27.

Note that when requesting or generating your license key(s), the number of CPUs specified should correspond with the total number of actual CPUs on all the systems upon which you wish to run PBS jobs.

**Important:** Each PBS Pro license key has specific data encoded within it. The following diagram explains what the different fields of the key represent.


L-00008-04153-1525-9OCYcj7Vtd-hV0
security checksum
sequence number
expiration (153rd day of year 2004, 99999 = infinite)
number of CPUs licensed
license type (L=Linux; 5=UNIX or Windows; ...)

**Important:** The PBS license key is keyed to the hostid of the PBS Server host. If the hostid changes (such as by upgrading the Server hardware) then the license key will become invalid and stop working. A new license key will then be needed.

When the installation of PBS is complete, you will need to install your PBS license key. If you already have your PBS license key, type it in when prompted by the license installation program, as shown below.

However, if you have not yet received your PBS license key, follow the instructions printed by the installation program (see example below) to receive your key. (The **PBS Pro Quick Start Guide** also provides step-by-step instructions on generating your license key.) Then rerun the PBS license key installation program as root:

```
# /usr/pbs/etc/pbs_setlicense
```

.

```
PBS license installation

Using /usr/spool/PBS as PBS_HOME
To get a license, please visit
        www.pbspro.com/license.html
 or call PBSpro toll free at 877-905-4PBS
 and have the following information handy:

***     host name:      mars.pbspro.com
***     host id:        12927f28
***     site id from the PBSPro package
***     number of cpus you purchased

Please enter the license string(s) (^d to end).
? 5-00020-99999-0044-PfV/fjuivg-5Jz

Installing: 5-00020-99999-0044-PfV/fjuivg-5Jz

Please enter the next license string(s) (^d to end).
?
Would you like to start PBS now (y|[n])? n
```

### 4.4.1 License Expiration Notice

If the PBS Pro software license that you install has an expiration date, you will be notified before the license expires. Email will be sent to the account (admin by default) that is defined by the Server parameter `mail_from`, which will be discussed in “Server Configuration Parameters” on page 52. Messages are sent 30 days in advance, 15 days in advance, and daily starting 7 days in advance. Upon expiration, the PBS Server will cease to run jobs.

### 4.4.2 Installing Multiple PBS Licenses

It is possible to add multiple licenses to the PBS License file. This can be done during installation of PBS Pro, or at some future time. If the installation program detects that you already have a PBS license file, it will prompt you as to what you want done: keep the file, replace the file, or append to it. Specify the option that corresponds to the action you wish to take.

Then, if you have multiple license keys, simply type them in when prompted by the license installation program, as shown in the example below.

Note that you can invoke the license key installation program directly (as may be needed following an increase in the size of the system or cluster managed by PBS), using the “-a” (append) option:

# **/usr/pbs/etc/pbs_setlicense -a**

Alternatively, you can manually edit the license file, adding the new license as a separate line in the file. Then you will need to shutdown the Server with “`qterm -t quick`” and restart it for the new license to be recognized.

```
PBS license installation
...
Please enter the license string(s) (^d to end).
? 5-00020-99999-0044-PfV/fjuivg-5Jz

Installing: 5-00020-99999-0044-PfV/fjuivg-5Jz

Please enter the next license string(s) (^d to end).
? 5-00020-99999-0010-XdsXdfssf-5Xj

Installing: 5-00020-99999-0010-XdsXdfssf-5Xj

Please enter the next license string(s) (^d to end).
?

Would you like to start PBS now (y|[n])? n
#
```

### 4.4.3 Using Floating Licenses

PBS can be purchased with *floating licenses*. When using floating licenses, you may have more CPUs configured online than the number of CPUs licensed. Nodes become licensed as needed to run jobs up to the number of floating licenses. The licenses are released from a node when no jobs remain running on the node.

The Server parameter "FLicense" shows the number of floating licenses currently available. A node attribute "pcpus" show the number of physical CPUs on the node, which determines the number of licenses required for that node. Another node attribute "license" shows the node "license state":

| | |
|---|---|
| `u` | unlicensed |
| `l` | licensed with a node-locked license |
| `f` | licensed with a floating license |

## 4.5 Installation on Windows 2000 and XP Systems

When PBS is installed on a cluster, the MOM service must be on each execution host, and the Server and Scheduler should be installed on one of the systems or on a front-end system. For Windows 2000 and XP clusters, PBS is provided in a single package containing:

the **PBS Pro Quick Start Guide** in PDF format,
the **PBS Pro Administrator Guide** in PDF format,
the **PBS Pro User Guide** in PDF format,
the PBS Pro software, and
supporting text files (software license, README, release notes, etc.)

Before installing PBS Pro on a Windows 2000 or XP cluster, perform the following system configuration steps first.

**Important:** Be sure to read the Readme file included in the Windows distribution of PBS Pro, as it contains important installation information that may not have been available when this book went to press.

The following discussion assumes that the `pbs_server` and `pbs_sched` services will be installed on a front-end host called “hostA”, and the `pbs_mom` service will installed on all the nodes in the cluster that will be running jobs, “hostB ... hostZ”.

1. Be sure that hostA, hostB, ..., hostZ consistently resolve to the correct IP addresses. A wrong IP address to hostname translation can cause PBS to misbehave. To set up the translations, edit `%WINDIR\system32\drivers\etc\hosts` on all the hosts within the cluster, as follows:

```
IP.address hostA
IP.address hostB
...
IP.address hostZ
```

The hosts file can usually be found under either of the following locations:

```
C:\winnt\system32\drivers\etc\hosts
C:\windows\system32\drivers\etc\hosts
```

2. Set up the account that will be used to run PBS jobs. We recommend the accounts be a non-Administrator type of account, that is, not a member of the “Administrators” group so that basic authentication using `hosts.equiv` can be used.

   The accounts can be set up using:

   Start->Control Panel->Administrative Tools->Computer Management->Local Users & Groups

   - or -

   Start->Control Panel->User Manager

   Once the accounts have been set up, edit the `hosts.equiv` file on all the hosts to include hostA, hostB, ..., hostZ to allow accounts on these hosts to access PBS services, such as job submission and remote file copying.

   The `hosts.equiv` file can usually be found in either of the following locations:

   ```
   C:\winnt\system32\drivers\etc\hosts.equiv
   C:\windows\system32\drivers\etc\hosts.equiv
   ```

3. Next you will need to install the PBS software. The PBS Pro installation program will walk you through the installation process. If you are installing from the PBS Pro CD-ROM, insert the CD-ROM into your computer’s CD-ROM drive, browse to your CD-ROM drive, and click on the PBS Pro program icon.

   Alternatively, you can download the latest PBS Pro package from the PBS Pro Web site, and save it to your hard drive. From there you can manually run the self-extracting `pbspro.exe` package, and then the installation program, as shown below.

   ```
   Admin> PBSpro_5.4.0-windows.exe
   ```

**Important:** Note that you must be logged in as Administrator to run the PBS installation program.

Install the execution nodes of your cluster, selecting the “Execution” node option from the install tool. When prompted for the PBS Server hostname, enter the correct hostname. In this example, it would be hostA.

4. Install PBS Pro on hostA, selecting the “All” option. Next, you will be prompted for your software license key(s). Following this, the install program will prompt for information needed in setting up the `nodes` file, the `hosts.equiv` file, etc. Enter the information requested for hosts hostB, hostC, ..., hostZ.

5. Finally, run “pbsnodes -a” on hostA to see if it can communicate with the execution nodes in your cluster. If some of the nodes are still down, then you may go to the problem node and restart the MOM, using the commands:

```
Admin> net stop pbs_mom
Admin> net start pbs_mom
```

### 4.5.1 Windows' pbs_rshd

The windows version of PBS contains a fourth service called `pbs_rshd` for supporting remote file copy requests issued by `pbs_rcp`. `pbs_rcp` is what PBS uses for delivering job output and error files to destination hosts. `pbs_rshd` is started automatically during installation but can also be started manually by doing:

```
net start pbs_rshd
```

-or-

```
pbs_rshd -d
```

in debug mode where logging output will be displayed on the command line.

Given a remote copy request issued by userA on hostA to hostB running `pbs_rshd`:

```
userA@hostA> pbs_rcp <file1> <hostB>:<file2>
```

If userA is a non-administrator account (e.g. not belonging to the Administrators group), then the copy will succeed in one of 2 ways: (1) userA@hostA is authenticated via `hosts.equiv` file; or (2) userA@hostA is authenticated via user's `[HOMEDIR]\.rhosts` file

For the above `pbs_rcp` request, you will need a `hosts.equiv` on hostB an entry of:

```
<hostA>
```

or, an entry in `[HOMEDIR]\.rhosts` on userA's account on hostB as follows:

```
<hostA> <userA>
```

if userA is an administrator account, or if a remote copy request looks like:

```
userA@hostA] pbs_rcp <file1> <userB>@<hostB>:<file2>
```

then `[HOMEDIR]\.rhosts` is the only way to authenticate. So in userB's account on hostB will need to have a `.rhosts` file containing the entry:

```
<hostA> <userA>
```

These two methods of authentication are further discussed in the **PBS Pro User Guide**.

## 4.6 Post Installation Validation

If you wish to validate the installation of PBS Pro, at any time, run the `pbs_probe` command. It will review the installation (installed files, directory and file permissions, etc) and report any problems found. For details, see section 11.3 “The pbs_probe Command” on page 198. (The `pbs_probe` command is not available under Windows.)

## 4.7 Network Addresses and Ports

PBS makes use of fully qualified host names for identifying the jobs and their location. A PBS installation is known by the host name on which the Server is running. The name used by the daemons, and used to authenticate messages is the canonical host name. This name is taken from the primary name field, `h_name`, in the structure returned by the library call `gethostbyaddr`(). According to the IETF RFCs, this name must be fully qualified and consistent for any IP address assigned to that host.

The daemons/services and the commands will attempt to use the system `services` file to identify the standard port numbers to use for communication. If the services cannot be found in the system file, the PBS components will use the default port numbers listed below. The service names that should be added to `services` are:

```
pbs               15001/tcp # Client/Scheduler to Server
pbs_server        15001/udp # Server to MOM via RPP
pbs_mom           15002/tcp # MOM to/from Server
pbs_resmon        15003/tcp # MOM RM requests
pbs_resmon        15003/udp # MOM RM requests
pbs_sched         15004/tcp # PBS Scheduler
pbs_mom_globus    15005/tcp # MOM Globus
pbs_resmon_globus 15006/tcp # MOM Globus RM requests
pbs_resmon_globus 15006/udp # MOM Globus RM requests
```

Under UNIX, the `services` file is named `/etc/services`.
Under Windows, it is named `%WINDIR%\system32\drivers\etc\services`.

The port numbers listed are the default numbers used by this version of PBS. If you change them, be careful to use the same numbers on all systems. (Note, the name *pbs_resmon* is a carry-over from early versions of PBS when there existed separate daemons for job execution (`pbs_mom`) and resource monitoring (`pbs_resmon`). The two functions were combined into `pbs_mom` though the term "resmom" might be found referring to the combined functions.)

# Chapter 5
# Upgrading PBS Pro

This chapter provides important information on upgrading from a previous version of PBS Pro. If PBS Pro is not currently installed on your system, you can safely skip this chapter at this time. However, be sure to refer to it before performing a future upgrade of PBS.

There are two types of upgrades available for PBS Pro:

*overlay upgrade* (which installs the new binaries on top of the old ones)

*migration upgrade* (which involves moving jobs from the old server to the new server).

One or the other of these needs to be used whenever there is a need to preserve existing jobs (migration won't preserve running jobs).

Usually the upgrade that is required is a simple overlay upgrade. The migration upgrade is only necessary when the new release has a change to the Server's internal "JOB" data structure, or if you are upgrading PBS Pro on Microsoft Windows.

**Important:** The Release Notes will indicate if a migration upgrade is necessary.

**Important:** Under Microsoft Windows a migration upgrade is always required.

## 5.1 Overlay Upgrade

Most new releases of PBS (especially minor versions and patch releases) will allow you to upgrade quickly. Except for the operating environments which are specifically singled out below, use the following steps to perform the overlay upgrade:

1 Shutdown PBS Pro by typing the command:

```
qterm -m -s          (PBS versions prior to PBSPro_5.4.0)
qterm -m -s -f       (PBS versions PBSPro_5.4.0 and later)
```

Also see “Starting and Stopping PBS: UNIX and Linux” on page 161, and section 10.2.6 “Impact of Shutdown / Restart on Running Jobs” on page 170.

2 Back up the Server's job directory (see details below).

3 Install the new version of PBS Pro on all nodes without uninstalling the previous installation.

4 Restart PBS Pro on all nodes, starting the execution hosts first.

The special case of overlay upgrade for Solaris environments:

1 Shutdown PBS Pro by typing the command:

```
qterm -m -s          (PBS versions prior to PBSPro_5.4.0)
qterm -m -s -f       (PBS versions PBSPro_5.4.0 and later)
```

2 Move the entire PBS home hierarchy to a temporary location:

```
mv /usr/spool/PBS /usr/spool/PBS.old
```

(this assumes the default location was used)

3 Remove the old PBS package:

("`pkginfo | grep -i pbs`" gets package name)

```
pkgrm  pbs64
   -or-
pkgrm  pbs32
```

4. Move PBS Pro home hierarchy back to its original location:

   **`mv /usr/spool/PBS.old /usr/spool/pbs`**
   (this assumes the default location was used)

5. Install the new PBS Pro version:

   **`./INSTALL`**

6. Restart PBS Pro on all nodes, starting the execution hosts first.

**Important:** The above procedures, platform non-specific and specific, apply only to PBS in UNIX environments.

**Important:** At the current time, overlay upgrades are not supported for PBS in Microsoft Windows environments. For those environments you must use the migration upgrade, which is discussed in section 5.2 below.

You may use `tar` to perform the backup of the jobs directory. This backup step is precautionary only; allowing jobs to be restored should an error occur.

```
# cd /usr/spool/PBS/server_priv
# tar -cf /tmp/pbs_jobs_save.tar jobs
```

Follow the normal installation procedures for installing the new version. The installation program will pick up your existing installation parameters from the `pbs.conf` file, and prompt you to confirm that you wish to use them.

## 5.2 Migration Upgrade

A migration upgrade is more complicated than an overlay upgrade. It is needed whenever the internal PBS job structure is different between the version of PBS that is installed and that to which you are upgrading. The release notes will indicate if the job migration procedure is necessary to retain existing jobs.

When the new version PBS is ready to be placed into service, you will probably want to move jobs from the old system to the new. The steps which follow form a procedure for carrying out this migration of jobs. It uses PBS's capability of having more than one copy of PBS running on the same system, by specifying alternative daemon directories and port numbers.

All Servers in the UNIX environment must be run as root, while those in the Microsoft Windows environment must be run as Administrator. The `qmgr` and `qmove` commands which follow need to be run by a user who is a PBS Administrator—typically this is going to be "root" if you are in a UNIX environment or "Administrator" if you are in a Windows environment.

**Step 1** Save a copy of the Server and nodes configuration to files, as this information will be necessary later on in the migration process. You should, as a precaution, check that these saved files really exist and appear to be correct in so far as size is concerned.

UNIX:

```
# cd /tmp
# qmgr -c “print server” > server.out
# qmgr -c “print node @default” > nodes.out
```

WINDOWS:

```
Admin> cd C:\WINDOWS\TEMP      (Windows XP environments)
Admin> cd C:\WINNT\TEMP        (Windows 2000 environments)

Admin> qmgr -c “print server” > server.out
Admin> qmgr -c “print node @default” > nodes.out
```

**Step 2** Next, disable the queues by setting each queue's "enabled" attribute to false, and stop any further scheduling requests to the Scheduler by setting the Server's "scheduling" attribute to false. This will prevent any new jobs from being accepted or started by PBS.

UNIX:

```
# qmgr -c “set server scheduling = false”

# qmgr -c “set queue queuename enabled = false”
(repeat above command for each queue)
```

WINDOWS:

```
Admin> qmgr -c “set server scheduling = false”
Admin> qstat                           (check for any running jobs)
Admin> qrerun -W force jobIDs   (requeue running jobs if possible)
Admin> qstat                           (check that no jobs are running)

(If any jobs remain in state RUNNING, you either need to wait for their termination or use the qdel command to delete them.)

# qmgr -c “set queue queuename enabled = false”
(repeat above command for each queue)
```

**Step 3** Shutdown PBS.

If your version of the Server doesn't have the "fail over" feature, omit the "-f" option that appears in the `qterm` command below.

UNIX:

```
# qterm -m -s -f
```

If the version of `qterm` does not have the other options, do a shutdown with the init script.

```
# /etc/init.d/pbs stop
    -or-
# /etc/rc.d/init.d/pbs stop
```

WINDOWS:

```
Admin> qterm -m -s -f
Admin> net stop pbs_rshd
```

**Step 4** Backup the Server’s jobs directory, *`PBS_HOME/`*`server_priv/jobs`.

In the UNIX environment the value for `PBS_HOME` can be found in file `/etc/pbs.conf`, and in the Windows environment it is the value, `\Program Files\PBS Pro\home`.

Backup of the Server's jobs directory is only precautionary, but an extremely prudent step in the event of operator or other error.

UNIX:

```
# cd PBS_HOME/server_priv
# tar -cf /tmp/pbs_jobs_save.tar jobs
```

WINDOWS: In either case below, specify "D" for directory when prompted and subsequently verify that this backup of your PBS jobs seems successful.

For Windows-XP environments, type the following command on a single line:

```
xcopy /o /e "\Program Files\PBS Pro\home\server_priv\jobs"
 "\WINDOWS\TEMP\jobs"
```

For Windows-2K environments, type the following command on a single line:

```
xcopy /o /e "\Program Files\PBS Pro\server_priv\jobs"
 "\WINNT\TEMP\jobs"
```

**Step 5** Install the new version of `pbs_mom` on each execution-only nodes. At the end of each installation, do not opt for restarting the MOM at this time, wait until all jobs have been successfully moved over to the new Server.

UNIX: Select option #2 of the INSTALL program.

WINDOWS: Choose, on the relevant installation menu, the choice "execution". As this step will install and launch `pbs_mom`, follow this by the command:

```
net stop pbs_mom
```

**Step 6** On the Server node, move the existing `PBS_HOME` and execution hierarchies to a temporary location.

LINUX:

```
# mv /var/spool/PBS /var/spool/PBS.old
# mv /usr/pbs /usr/pbs.old
```

UNIX:

```
# mv /usr/spool/PBS /usr/spool/PBS.old

# mv /usr/pbs /usr/pbs.old
-or-
# mv /usr/local/pbs /usr/local/pbs.old
-or-
# mv /opt/pbs /opt/pbs.old
```

WINDOWS: For the Windows environments do a `copy` operation rather than a `move`, since the installation program doesn't accept a missing `PBS_HOME` directory:

```
xcopy /o /e "\Program Files\PBS Pro" "\Program Files\PBS Pro.old"
```

In the above, specify "D" for directory when prompted. If you get an "access denied" error message while deleting a file in \Program Files\PBS Pro\home\spool, then bring up Start menu->Programs->Accessories->Windows Explorer, and right mouse click to select this file and bring up a pop-up menu, choose "Properties", select "Security" tab, then "Advanced", then Owners tab, from which you can reset the ownership of the file to "Administrators". Be sure also that "Administrators" has permission to read the file. Then rerun xcopy.

**Step 7** Install the new version of PBS on the Server host.

UNIX: Perform the installation on the Server host using the INSTALL program, and then start PBS on that host.

WINDOWS: For windows environments you need to run the PBS package executable twice. The first run attempts to remove the currently installed package. The second run does the actual install.

A word of caution is appropriate at this point. Namely, make sure there isn't any access occurring on any file in the hierarchy

```
C:\Program Files\PBS Pro
```

when you do the uninstall of PBS Pro (e.g. don't be cd-ing to a subdirectory.) If there is access, you will be presented with a pop-up window stating that the above hierarchy could not be removed. This will force you to have to do the removal and the install manually, in which case you need to consult the Release Note that covers this scenario before you do the re-install.

The manual removal is done from a cmd shell window by typing:

```
rmdir /S /Q "C:\Program Files\PBS Pro"
```

Do not use the "del" command in place of rmdir as there may be circumstances where not everything is removed, which will later on cause problems in the migration procedure.

As the install also launches the PBS services, issue the following command to stop pbs_mom on the server node:

```
net stop pbs_mom
```

**Step 8** Restart the server in a way that it receives an empty configuration. Do this as follows:

UNIX:

```
# qterm

Do the appropriate one of the following:
# rm /usr/spool/PBS/server_priv/nodes
# rm /var/spool/PBS/server_priv/nodes

Do the appropriate one of the following:
# /usr/pbs/bin/pbs_server -t create
# /usr/local/pbs/sbin/pbs_server -t create
# /opt/pbs/sbin/pbs_server -t create
```

WINDOWS:

```
Admin> qterm
Admin> del "\Program Files\PBS Pro\home\server_priv\nodes"
Admin> "\Program Files\PBS Pro\exec\sbin\pbs_server" -C
Admin> net start pbs_server
```

**Step 9** Now give to this new Server instance a duplicate of the queues and Server parameters that got saved in Step 1 (this assumes you want this new Server instance to have the same configuration as the old).

UNIX:

```
# qmgr < /tmp/server.out
# qmgr < /tmp/nodes.out
```

WINDOWS:

For windows-XP environments:

```
Admin> qmgr < "\WINDOWS\TEMP\server.out"
Admin> qmgr < "\WINDOWS\TEMP\nodes.out"
```

For windows-2K environments:

```
Admin> qmgr < "\WINNT\TEMP\server.out"
Admin> qmgr < "\WINNT\TEMP\nodes.out"
```

**Step 10** Now start the original Server on different ports and direct it to take for the location of its database the location that you specified in Step 6.

UNIX:

```
# /usr/pbs.old/sbin/pbs_server -p 13001 -M 13002 \
  -R 13003 -S 13004 -g 13005 -d /usr/spool/pbs.old

# /usr/pbs.old/sbin/pbs_server -p 13001 -M 13002 \
  -R 13003 -S 13004 -g 13005 -d /var/spool/pbs.old

# /usr/local/pbs.old/sbin/pbs_server -p 13001 -M 13002 \
  -R 13003 -S 13004 -g 13005 -d /usr/spool/pbs.old

# /opt/pbs.old/sbin/pbs_server -p 13001 -M 13002 \
  -R 13003 -S 13004 -g 13005 -d /usr/spool/pbs.old
```

WINDOWS: Type the following command(s), without splitting across lines. Be sure that the command(s) reference the “old” Server and the “old” hierarchy.

You may omit the first command (`del`) if you are not upgrading from `PBSPro_5.3.3-wp1`.

```
Admin> del "\Program Files\PBS Pro.old\home\server_priv\
server.lock"
Admin> "\Program Files\PBS Pro.old\exec\sbin\pbs_server" -N
-p 13001 -M 13002 -R 13003 -S 13004 -g 13005 -d "\Program
Files\PBS Pro.old\home”
```

Verify that the old pbs_server is running on the alternate ports by going to another Cmd prompt window and running:

```
qstat @hostname:13001
```

**Step 11** Now list the jobs at the old Server and move a few jobs one at a time from the old to the new Server. Note that the standard port number of the *new* Server (15001) is needs to be part of the destination queue specification.

UNIX: The commands need to be run out of the old version of PBS which will be in `/usr/pbs.old/bin`, `/usr/local/pbs.old/bin` or `/opt/pbs.old/bin` depending on architecture (see step 6). Call this location `/oldpath/`.

This first step in the sequence is here to catch the special case where one is migrating jobs from a pre-`PBSPro_5.4.0` version of the Server and that Server's host happens to be also running a `pbs_mom`. The command will present a message about "no such node" if the host is not in the Server's nodes set. This is not a problem.

```
# /oldpath/qmgr -c "d n serverhost" serverhost:13001
```

If the message "Cannot delete busy object" occurs, rerun the command again, after you use `qrerun` or `qdel` to remove job(s) currently using the node.

```
# /oldpath/qstat @host:13001
# /oldpath/qmove queue@host:15001 job_id1@host:13001
# /oldpath/qmove queue@host:15001 job_id2@host:13001
...
# /oldpath/qmove queue@host:15001 job_idN@host:13001
# qstat
```

WINDOWS: On another command prompt window, do:

```
Admin> qstat @host:13001
Admin> qmove queue@host:15001 job_id1@host:13001
Admin> qmove queue@host:15001 job_id2@host:13001
...
Admin> qmove queue@host:15001 job_idN@host:13001
Admin> qstat
```

**Step 12** At this point, all of the jobs should be under control of the new Server and located in the new Server’s home. The old Server can now be shut down and the MOMs (`pbs_mom`) started.

## 5.3 Alternate Test Systems

Running an alternate or test version of PBS requires a couple extra steps. In particular, the alternate version will need a separate PBS directory structure from which to run, this must be created before attempting to start the alternate version.

As a precautionary step, you can copy your existing PBS directory tree to the new location. (`tar` may be used for this.) However, you will want to ensure that you delete any job or reservation data that might have been copied to the alternate location.

```
# cd existing_PBS_HOME
# tar -cvf /tmp/pbs_tree.tar .
# mkdir /path/to/alternate_PBS_HOME
# cd /path/to/alternate_PBS_HOME
# tar -xvf /tmp/pbs_tree.tar
# /bin/rm server_priv/jobs/* mom_priv/jobs/*
# /bin/rm server_priv/resvs/*
```

Alternate or test copies of the various daemons may be run through the use of the command line options which set their home directory and service port. For example, the following commands would start the three daemons with a home directory of `/tmp/altpbs` and port assignments as follows: the Server on 13001, MOM on 13002 and 13003, optional MOM Globus on 13004, 13005, and the Scheduler on 13004.

```
# pbs_server -t create -d /tmp/altpbs -p 13001 -M 13002 \
   -R 13003 -S 13004 -g 13005 -G 13006
# pbs_mom -d /tmp/altpbs -S 13001 -M 13002 -R 13003
# pbs_sched -d /tmp/altpbs -S 13004 -R 13003
# pbs_mom_globus -d /tmp/altpbs -S 13001 -M 13005 -R 13006
```

Note that when the Server is started with a non-standard port number (i.e. with the `-p` option as shown above) the Server “name” becomes *host_name.domain:port*, where port is the numeric port number being used.

Jobs may now be directed to the test system by using the `-q` option to qsub with the `server:port` syntax. Status is also obtained using the `:port` syntax. For example, to submit a job to the default queue on the above test Server, request the status of the test Server, and request the status of jobs at the test Server:

```
      # qsub  -q @host:13001 jobscript
      # qstat -Bf host:13001
      # qstat @host:13001
```

**Important:** If using job dependencies on or between test systems, there are minor problems of which you (and the users) need to be aware. The syntax of both the dependency string and the job `host:port` syntax use colons in an indistinguishable manner.

## 5.4 Dependent Jobs and Test Systems

If you have users running on a test batch system using an alternative port number, `-p` option to `pbs_server`, problems may occur with job dependency if the following requirements are not observed:
For a test system, the job identifier in a dependency specification must include at least the first part of the host name.

The colon in the port number specification must be escaped by a back-slash. This is true for both the Server and current Server portion of the jobid. For example:

```
23.test_host\:17000
23.old_host@test_host\:17000
23.test_host\:17000@diff_test_host\:18000
```

On a shell line, the back slash itself must be escaped from the shell, so the above become:

```
23.test_host\\:17000
23.old_host@test_host\\:17000
23.test_host\\:17000@diff_test_host\\:18000
```

These rules are not documented on the qsub/qalter man pages since the likelihood of the general user community finding themselves setting up dependencies with jobs on a test system is small and the inclusion would be generally confusing.

Chapter 6

# Configuring the Server

Now that PBS Pro has been installed, the Server and MOMs can be configured and the scheduling policy selected. The next three chapters will walk you through this process. Further configuration may not be required as the default configuration may completely meet your needs. However, you are advised to read this chapter to determine if the default configuration is indeed complete for you, or if any of the optional settings may apply.

## 6.1 The qmgr Command

The PBS manager command, `qmgr`, provides a command-line administrator interface to the PBS Server. The command usage is:

```
qmgr [-a] [-c command] [-e] [-n] [-z] [server...]
```

The available options, and description of each, follows.

| Option | Action |
|---|---|
| `-a` | Abort `qmgr` on any syntax errors or any requests rejected by a Server. |

| Option | Action |
|---|---|
| `-c command` | Execute a single command and exit `qmgr`. The command must be enclosed in quote marks, e.g. `qmgr -c "print server"` |
| `-e` | Echo all commands to standard output. |
| `-n` | No commands are executed, syntax checking only is performed. |
| `-z` | No errors are written to standard error. |

If `qmgr` is invoked without the `-c` option and standard output is connected to a terminal, `qmgr` will write a prompt to standard output and read a directive from standard input.

A command is terminated by a new line character or a semicolon (";”) character. Multiple commands may be entered on a single line. A command may extend across lines by escaping the new line character with a back-slash (“\”). Comments begin with the “#” character and continue to end of the line. Comments and blank lines are ignored by `qmgr`. The syntax of each directive is checked and the appropriate request is sent to the Server(s). A `qmgr` directive takes one of the following forms:

```
command server [names] [attr OP value[,...]]
command queue  [names] [attr OP value[,...]]
command node   [names] [attr OP value[,...]]
```

Where command is the sub-command to perform on an object. Commands are:

| Command | Explanation |
|---|---|
| `active` | Sets the active objects. If the active objects are specified, and the name is not given in a `qmgr` command the active object names will be used. |
| `create` | Create a new object, applies to queues and nodes. |
| `delete` | Destroy an existing object, applies to queues and nodes. |
| `help` | Prints command specific help and usage information |
| `list` | List the current attributes and associated values of the object. |
| `print` | Print settable queue and Server attributes in a format that will be usable as input to the `qmgr` command. |

| Command | Explanation |
|---|---|
| `set` | Define or alter attribute values of the object. |
| `unset` | Clear the value of the attributes of the object. Note, this form does no accept an OP and value, only the attribute name. |

Other `qmgr` syntax definitions follow:

| Variable | `qmgr` Variable/Syntax Description |
|---|---|
| `names` | List of one or more names of specific objects. The name list is in the form:<br>`[name][@server][,name[@server]...]`<br>with no intervening white space. The name of an object is declared when the object is first created. If the name is `@server`, then all the objects of specified type at the Server will be effected. |
| `attr` | Specifies the name of an attribute of the object which is to be set or modified. The attributes of objects are described in section 2 of the **PBS ERS**, and on the relevant attribute man page (e.g. `pbs_node_attributes(3B)`). If the attribute is one which consists of a set of resources, then the attribute is specified in the form:<br>`attribute_name.resource_name` |
| `OP` | An operation to be performed with the attribute and its value: |
| `=` | Set the value of the attribute. If the attribute has an existing value, the current value is replaced with the new value. |
| `+=` | Increase the value of the attribute by the amount specified. |
| `-=` | Decrease the value of the attribute by the amount specified. |
| `value` | The value to assign to an attribute. If value includes white space, commas, or other special characters, such as “#”, the value string must be inclosed in quote marks (“ ”). |

A few examples of the `qmgr` command follow. Full explanation of these and other `qmgr` commands are given below in explanation of the specific tasks they accomplish.

```
% qmgr
Qmgr: create node mars ntype=cluster
Qmgr: set node mars resources_available.ncpus=2
Qmgr: create node venus properties=”inner,moonless”
Qmgr: set node mars properties = inner
Qmgr: set node mars properties += haslife
Qmgr: delete node mars
Qmgr: d n venus
```

**Important:** Commands can be abbreviated to their minimum unambiguous form (as shown in the last line in the example above).

### 6.1.1 `qmgr` Help System

The `qmgr` built-in help function, invoked using the “help” sub-command, is illustrated by the next example.

```
% qmgr
Qmgr: help
To get help on any topic, type help <topic>
Help is available on all commands and topics.
Available commands:
active, create, delete, set, unset, list, print, quit
Other topics: attributes, operators, names, and values
```

For example, requesting usage information on `qmgr`’s set command would produce the following output.

```
% qmgr
Qmgr: help set
Syntax:
    set object [name][,name...] attribute[.resource] OP value
Objects can be "server" or "queue", "node"
The "set" command sets the value for an attribute on the spec-
ified object. If the object is "server" and name is not spec-
ified, the attribute will be set on all the servers specified
on the command line. For multiple names, use a comma separated
list with no intervening whitespace.
Examples:
set server s1 max_running = 5
set server managers = root
set server managers += susan
set node n1,n2 state=down
set queue q1@s3 resources_max.mem += 5mb
set queue @s3 default_queue = batch
```

## 6.2 Default Configuration

Server management consists of configuring the Server parameters, defining nodes, and establishing queues and their attributes. The default configuration from the binary installation sets the minimum Server settings, and some recommended settings for a typical PBS cluster. (The default Server configuration is shown below.) The subsequent sections in this chapter list, explain, and provide the default settings for all the Server's parameters for the default binary installation.

```
% qmgr
Qmgr: print server
# Create queues and set their attributes.
#
# Create and define queue workq
#
create queue workq
set queue workq queue_type = Execution
set queue workq enabled = True
set queue workq started = True
#
# Set server attributes.
#
set server scheduling = True
set server default_queue = workq
set server log_events = 511
set server mail_from = adm
set server query_other_jobs = True
set server scheduler_iteration = 600
```

### 6.2.1 PBS Levels of Privilege

The `qmgr` command is subject to the three levels of privilege in PBS: Manager, Operator, and user. In general, a "Manager" can do everything offered by `qmgr` (such as creating/deleting new objects like queues and nodes, modifying existing objects, and changing parameters that affect policy). The "Operator" level is more restrictive. Operators cannot create new objects nor modify any parameter that changes scheduling policy. A "user" can view, but cannot change, Server configuration information. For example, the `help`, `list` and `print` sub-commands of `qmgr` can be executed by the general user. Creating or deleting a queue requires PBS Manager privilege. Setting or unsetting `Server` or `queue` attributes (discussed below) requires PBS Operator or Manager privilege. Specifically, Manager privilege is required to create and delete queues or nodes, and set/alter/unset the following parameters:

| | | |
|---|---|---|
| all node properties | server acl_host_enable | server acl_host_list |
| server acl_user_enable | server acl_users | server acl_roots |
| server managers | server operators | server default_node |
| server mail_from | server query_other_jobs | |

For details on setting these levels of privilege, see the `managers` and `operators` Server parameters, discussed in “Server Configuration Parameters” on page 52; for security-related aspects of PBS privilege, see section 10.7 “External Security” on page 178.)

## 6.3 Hard versus Soft Limits

Most limits under PBS are *hard* limits (i.e. they cannot be exceeded by the job). However, a *soft* limit is a limit which *can* be exceeded, if additional resources are available when the job is running. Note that any job which exceeds its soft limits is eligible to be preempted by jobs of other users who are not exceeding the soft limits. In the sections on Server and Queue parameters that follow, all parameters that pertain to a limit are in fact hard limits, with the exception of the following parameters: `max_user_res_soft`, `max_user_run_soft`, `max_group_res_soft`, and, `max_group_run_soft`. See also the discussion of scheduling parameters using soft limits in Chapter 8 .

**Important:** There are no soft limits attributes for nodes.

## 6.4 Server Configuration Parameters

This section explains all the available Server configuration parameters and gives the default values for each. Note that the possible values for the “boolean” format are any of: “TRUE”, “True”, “true”, “Y”, “y”, “1”; “FALSE”, “False”, “false”, “N”, “n”, “0”. The privilege required to set or change some Server parameters has changed since the previous release.

acl_group_enable When true directs the Server to use the specified group access control list `acl_groups`.
Format: boolean
Default value: false = disabled

```
Qmgr: set server acl_group_enable=true
```

acl_groups

List which allows or denies enqueuing of jobs owned by members of the listed groups. The groups in the list are groups on the Server host, not submitting host. Note that the job's execution GID is evaluated (which is either the user's default group, or the group specified by the user via the `-Wgroup_list` option to `qsub`) against the user's primary groupID. See also `acl_group_enable`.
Format: "[+|-]group_name[,...]"
Default value: unset, thus all groups allowed

```
Qmgr: set server acl_groups="math,physics"
```

acl_host_enable

When `true` directs the Server to use the `acl_hosts` access control lists. Requires Manager privilege to set or alter.
Format: boolean
Default value: false = disabled

```
Qmgr: set server acl_host_enable=true
```

acl_hosts

List of hosts which may request services from this Server. This list contains the fully qualified network name of the hosts. Local requests, i.e. from the Server's host itself, are always accepted even if the host is not included in the list. Wildcards ("*") may be used in conjunction with host names and host.subdomain names. See also `acl_host_enable`.
Format: "[+|-]hostname.domain[,...]"
Default value: all hosts

```
Qmgr: set server acl_hosts=*.pbspro.com
```

acl_resv_host_enable

When true directs the server to use the `acl_resv_hosts` access control list. Requires Manager privilege to set or alter.
Format: boolean
Default value: false = disabled

```
Qmgr: set server acl_resv_host_enable=true
```

acl_resv_hosts

List of hosts which may request reservation services from this server. This list contains the network name of the hosts. Local requests, i.e. from the Server's host itself, are always accepted even if the host is not included in the list. Wildcards ("*") may be used in conjunction with host names and host.subdomain names. Requires Manager privilege to set or alter. See also `acl_resv_enable`.
Format: "[+|-]hostname.domain[,...]"
Default value: all hosts

```
Qmgr: set server acl_resv_hosts=*.pbspro.com
```

acl_resv_group_enable

When true directs the Server to use the reservation group access

control list `acl_resv_groups`. Requires Manager privilege to set or alter.
Format: boolean
Default value: false = disabled

```
Qmgr: set server acl_resv_group_enable=true
```

acl_resv_groups

List which allows or denies accepting reservations owned by members of the listed groups. The groups in the list are groups on the Server host, not submitting hosts. See also `acl_resv_group_enable`.
Format: "[+|-]group_name[,...]"
Default value: all groups allowed

```
Qmgr: set server acl_resv_groups=”blue,green”
```

acl_resv_user_enable

When true directs the Server to use the Server level `acl_resv_users` access list. Requires Manager privilege to set or alter.
Format: boolean
Default value: disabled

```
Qmgr: set server acl_resv_user_enable=true
```

acl_resv_users

List of users allowed or denied the ability to make any reservation requests of this Server. Requires Manager privilege to set or alter. See also `acl_resv_user_enable`.
Format: "[+|-]user[@host][,...]"
Default value: all users allowed

```
Qmgr: set server acl_resv_users=”bob,sue@sol”
```

acl_user_enable

When true directs the Server to use the Server level `acl_users` access list. Requires Manager privilege to set or alter.
Format: boolean
Default value: disabled

```
Qmgr: set server acl_user_enable=true
```

acl_users

List of users allowed or denied the ability to make any requests of this Server. Requires Manager privilege to set or alter. See also `acl_users_enable`.
Format: “[+|-]user[@host][,...]”
Default value: all users allowed

```
Qmgr: set server acl_users=”bob,tom@sol”
```

acl_roots

List of superusers who may submit to and execute jobs at this Server. If the job execution id would be zero (0), then the job owner, root@host, must be listed in this access control list or the job is rejected.
Format: “[+|-]user[@host][,...]”
Default value: no root jobs allowed

```
Qmgr: set server acl_roots=host
```

comment

A text string which may be set by the Scheduler or other privileged client to provide information to PBS users.
Format: any string
Default value: none

```
Qmgr: set server comment="Planets Cluster"
```

default_node

A node specification to use if there is no other supplied specification. The default value allows jobs to share a single node.
Format: a node specification string
Default value: 1#shared

```
Qmgr: set server default_node="1#shared"
```

default_queue

The queue which is the target queue when a request does not specify a queue name.
Format: a queue name.
Default value: none, must be set to an existing queue

```
Qmgr: set server default_queue=workq
```

flatuid

Attribute which directs the Server to automatically grant authorization for a job to be run under the user name of the user who submitted the job even if the job was submitted from a different host. If not set `true`, then the Server will check the authorization of the job owner to run under that name if not submitted from the Server's host. See section 10.6.5 “User Authorization” on page 176 for usage and important caveats.
Format: boolean
Default value: false = disabled

```
Qmgr: set server flatuid=True
```

log_events

A bit string which specifies the type of events which are logged, See also section 10.12 “Use and Maintenance of Logfiles” on page 183.
Format: integer
Default value: 511 (all events)

```
Qmgr: set server log_events=255
```

mail_from

The email address used as the "from" address for Server generated mail sent to users.
Format: string
Default value: adm

```
Qmgr: set server mail_from=boss@pbspro.com
```

Note: This email address will also be used by the Server as the "to" address to send important events and warnings. Currently, these are limited to warnings regarding expiration of PBS licenses. Therefore, this should be set to a valid address or alias to which mail, when sent, is actually read.

managers

List of users granted PBS Manager privileges. The host, subdomain, or domain name may be wild carded by the use of an * character. Requires Manager privilege to set or alter.
Format: “user@host.sub.domain[,user@host.sub.domain...]”
Default value: root on the local host

```
Qmgr: set server managers+=boss@sol.pbspro.com
```

max_running

The maximum number of jobs allowed to be selected for execution at any given time.
Format: integer
Default value: none

```
Qmgr: set server max_running=24
```

max_group_res
max_group_res_soft

The maximum amount of the specified resource that all members of the same group may consume simultaneously. The named resource can be any valid PBS resource, such as “ncpus”, “mem”, “pmem”, etc. This limit can be specified as either a *hard* or *soft* limit. (See also section 6.3 “Hard versus Soft Limits” on page 52.)
Format: “max_group_res.resource_name=value[,...]”
Format: “max_group_res_soft.resource_name=value[,...]”
Default value: none

```
Qmgr: set server max_group_res.ncpus=10
Qmgr: set server max_group_res_soft.mem=1GB
```

The first line in the example above sets a normal (e.g. *hard*) limit of 10 CPUs as the aggregate maximum that any group may consume. The second line in the example illustrates setting a group *soft* limit of 1GB of memory.

max_group_run
max_group_run_soft

The maximum number of jobs owned by any user in a single group that are allowed to be running from this queue at one time.This limit can be specified as either a *hard* or *soft* limit. (See also section 6.3 “Hard versus Soft Limits” on page 52.)
Format: integer
Default value: none

```
Qmgr: set server max_group_run=10
Qmgr: set server max_group_run_soft=7
```

max_user_run
max_user_run_soft

The maximum number of jobs owned by a single user that are allowed to be running at one time. This limit can be specified as either a *hard* or *soft* limit. (See also section 6.3 “Hard versus Soft Limits” on page 52.)

Format: integer
Default value: none

```
Qmgr: set server max_user_run=6
Qmgr: set server max_user_run_soft=3
```

max_user_res
max_user_res_soft

The maximum amount of the specified resource that any single user may consume. The named resource can be any valid PBS resource, such as “ncpus”, “mem”, “pmem”, etc. This limit can be specified as either a *hard* or *soft* limit. (See also section 6.3 “Hard versus Soft Limits” on page 52.)
Format: “max_user_res.resource_name=value[,...]”
Format: “max_user_res_soft.resource_name=value[,...]”
Default value: none

```
Qmgr: set server max_user_res.ncpus=6
Qmgr: set server max_user_res_soft.ncpus=3
```

The first line in the example above sets a normal (e.g. *hard*) limit of 3 CPUs as a maximum that any single user may consume. The second line in the example illustrates setting a *soft* limit of 6 CPUs on the same resource.

node_fail_requeue

Controls if running multi-node jobs are automatically requeued (or deleted) if the primary execution node fails (e.g. due to system crash or power failure). If this parameter is unset or set to a value less than or equal to zero, PBS will leave the job in a Running state when the first node allocated to the job (Mother Superior node) is reported down. (This default behavior is the same as in previous versions of PBS Pro.) However, if this parameter is set to any non-zero positive integer (“N”), it defines the number of seconds that the PBS Server will wait for the node to come back online before the job is requeued or deleted. If after *N* seconds the node is still down, any job which has that node as its first node will be (a) requeued if the job's rerun attribute is set to 'y' (yes); or (b) deleted if it is set to 'n' (no). If a job is deleted, mail will be sent to the owner of the job. Requires either Manager or Operator privilege to set. The value selected for *N* should be long enough to exceed any transient non-node failures, but short enough to requeue the job in a timely fashion.
Format: integer
Default value: unset

```
Qmgr: set server node_fail_requeue=300
```

node_group_enable

When true directs the Server to enable node grouping. Requires Manager privilege to set or alter. See also `node_group_key`, and

section 6.8 “Node Grouping” on page 77.
Format: boolean
Default value: disabled

```
Qmgr: set server node_group_enable=true
```

node_group_key

Specifies the resource to use for node grouping. Requires Manager privilege to set or alter. See also `node_group_enable`, and section 6.8 “Node Grouping” on page 77.
Format: string
Default value: disabled

```
Qmgr: set server node_group_key=resource
```

node_pack

Deprecated.

operators

List of users granted PBS Operator privileges.
Format of the list is identical with `managers` above. Requires Manager privilege to set or alter.
Default value: root on the local host.

```
Qmgr: set server operators=”sue,bob,joe,tom”
```

query_other_jobs

The setting of this parameter controls whether or not general users, other than the job owner, are allowed to query the status of or select the job. Requires Manager privilege to set or alter.
Format: boolean
Default value: false (users may not query or select jobs owned by other users)

```
Qmgr: set server query_other_jobs=true
```

require_cred_enable

When true directs the Server to use the credential authentication method specified by `require_cred`. Requires Manager privilege to set or alter. Depends on optional KRB and DCE support.
Format: boolean
Default value: false = disabled

```
Qmgr: set server require_cred_enable=true
```

require_cred

Specifies the credential type required. All jobs submitted without the specified credential will be rejected. Requires Manager privilege to set or alter. See also `require_cred_enable`. Depends on optional KRB and DCE support.
Format: string (krb5 or dce)
Default value: unset

```
Qmgr: set server require_cred=krb5
```

The above example would cause the Server to reject all jobs that do not have a credential type of "krb5".

resources_available

The list of resources and amounts available to jobs run by this

Server. The sum of the resources of each type used by all jobs running by this Server cannot exceed the total amount listed here.
Format: “resources_available.resource_name=value[,...]”
Default value: unset

```
Qmgr: set server resources_available.ncpus=16
Qmgr: set server resources_available.mem=400mb
```

resources_default The list of default resource values that are set as limits for a job executing on this Server when the job does not specify a limit, and there is no queue default. See also section 6.10 “Resource Default/Min/Max Attributes” on page 79.
Format: “resources_default.resource_name=value[,...]
Default value: no limit

```
Qmgr: set server resources_default.mem=8mb
Qmgr: set server resources_default.ncpus=1
```

resources_max The maximum amount of each resource which can be requested by a single job executing on this Server if there is not a `resources_max` valued defined for the queue in which the job resides. See also section 6.10 “Resource Default/Min/Max Attributes” on page 79.
Format: “resources_max.resource_name=value[,...]
Default value: infinite usage

```
Qmgr: set server resources_max.mem=1gb
Qmgr: set server resources_max.ncpus=32
```

resources_min The minimum amount of each resource which can be requested by a single job executing on this Server if there is not a `resources_min` valued defined for the queue in which the job resides. See also section 6.10 “Resource Default/Min/Max Attributes” on page 79.
Format: “resources_min.*resource_name*=*value*”
Default value: unset

```
Qmgr: set server resources_min.mem=1kb
Qmgr: set server resources_min.ncpus=1
```

resv_enable This parameter can be used as a master switch to turn on/off advance reservation capability on the Server. If set False, advance reservations are not accepted by the Server, however any already existing reservations will not be automatically removed. If this parameter is set True the Server will accept, for the Scheduler’s subsequent consideration, any reservation submission not otherwise rejected do to the functioning of some Administrator established ACL list controlling reservation submission. Requires Manager privilege to set or alter.
Format: boolean

Default value: True = enabled

```
Qmgr: set server resv_enable=true
```

scheduler_iteration

The time, in seconds, between iterations of attempts by the Server to schedule jobs. On each iteration, the Scheduler examines the available resources and runnable jobs to see if a job can be initiated. This examination also occurs whenever a running job terminates or a new job is placed in the queued state in an execution queue.
Format: integer seconds
Default value: 600

```
Qmgr: set server scheduler_iteration=300
```

scheduling

Controls if the Server will request job scheduling by the PBS Scheduler. If true, the Scheduler will be called as required; if false, the Scheduler will not be called and no job will be placed into execution unless the Server is directed to do so by a PBS Operator or Manager. Setting or resetting this parameter to `true` results in an immediate call to the Scheduler.
Format: boolean
Default value: value of `-a` option when Server is invoked; if `-a` is not specified, the value is recovered from the prior Server run. If it has never been set, the value is "false".

```
Qmgr: set server scheduling=true
```

The following parameters are read-only: they are maintained by the Server and cannot be changed by a client.

FLicenses

Shows the number of floating licenses currently available.

PBS_version

The release version number of the Server.

resources_assigned

The total amount of certain resources allocated to running jobs.

server_host

The name of the host on which the current (Primary or Secondary) Server is running, in failover mode.

server_name

The name of the Server, which is the same as the host name. If the Server is listening to a non-standard port, the port number is appended, with a colon, to the host name. For example: `host.domain:9999`.

state_count

Tracks the number of jobs in each state currently managed by the Server

server_state

The current state of the Server. Possible values are:

| | |
|---|---|
| Active | The Server is running and will invoke the Scheduler as required to schedule jobs for execution. |
| Hot_Start | The Server may remain in this state for up to five minutes after being restarted with the “hot” option on the command line. Jobs that are already running will remain in that state and jobs that got requeued on shutdown will be rerun. |
| Idle | The Server is running but will not invoke the Scheduler. |
| Scheduling | The Server is running and there is an outstanding request to the job scheduler. |
| Terminating | The Server is terminating. No additional jobs will be scheduled. |
| Terminating, Delayed | The Server is terminating in delayed mode. The Server will not run any new jobs and will shutdown when the last currently running job completes. |

total_jobs The total number of jobs currently managed by the Server.

## 6.5 Queues within PBS Pro

Once you have the Server parameters set the way you want them, you will next want to review the queue settings. The default (binary) installation creates one queue with the attributes shown in the example below. You may wish to change these settings or add other attributes or add additional queues. The following discussion will be useful in modifying the PBS queue configuration to best meet your specific needs.

```
% qmgr
Qmgr: print queue workq
#
# Create and define queue workq
#
create queue workq
set queue workq queue_type = Execution
set queue workq enabled = True
set queue workq started = True
```

### 6.5.1 Execution Queues vs. Route Queues

There are two types of queues defined by PBS: routing and execution. A **routing queue** is a queue used to move jobs to other queues including those which exist on different PBS Servers. (Routing queues are similar to the old NQS pipe queues.) A job must reside in an **execution queue** to be eligible to run. The job remains in the execution queue during the time it is running. In spite of the name, jobs in a queue need not be processed in queue-order (first-come first-served or *FIFO*). A Server may have multiple queues of either or both types, but there must be at least one queue defined. Typically it will be an execution queue; jobs cannot be executed while residing in a routing queue.

See the following sections for further discussion of execution and route queues:

section 6.5.3 “Attributes for execution queues only” on page 66
section 6.5.4 “Attributes for route queues only” on page 67
section 6.11 “Selective Routing of Jobs into Queues” on page 80
section 6.15.6 “Failover and Route Queues” on page 94
section 12.5 “Complex Multi-level Route Queues” on page 226.

### 6.5.2 Queue Configuration Parameters

Queue configuration parameters fall into three groups: those which are applicable to both types of queues, those applicable only to execution queues, and those applicable only to routing queues. If an “execution queue only” attribute is set for a routing queue, or vice versa, it is simply ignored by the system. However, as this situation might indicate the Administrator made a mistake, the Server will issue a warning message (on stderr) about the conflict. The same message will be issued if the queue type is changed and there are attributes that do not apply to the new type.

Queue public attributes are alterable on request by a client. The client must be acting for a user with Manager or Operator privilege. Certain parameters require the user to have full Administrator privilege before they can be modified. The following parameters apply to both queue types:

**Important:** Note, an *unset* resource limit (i.e. a limit for which there is no default, minimum, nor maximum) is treated as an infinite limit.

acl_group_enable
When true directs the Server to use the queue’s group access control list `acl_groups`.
Format: boolean
Default value: false = disabled
`Qmgr:` **`set queue`** ***`QNAME`*** **`acl_group_enable=true`**

acl_groups

List which allows or denies enqueuing of jobs owned by members of the listed groups. The groups in the list are groups on the Server host, not submitting host. Note that the job’s execution GID is evaluated (which is either the user’s default group, or the group specified by the user via the `-Wgroup_list` option to `qsub`.) See also `acl_group_enable`.
Format: “[+|-]group_name[,...]”
Default value: all groups allowed

```
Qmgr: set queue QNAME acl_groups=”math,physics”
```

acl_host_enable

When true directs the Server to use the `acl_hosts` access list for the named queue.
Format: boolean
Default value: disabled

```
Qmgr: set queue QNAME acl_host_enable=true
```

acl_hosts

List of hosts which may enqueue jobs in the queue. See also `acl_host_enable`.
Format: “[+|-]hostname[,...]”
Default value: all hosts allowed

```
Qmgr: set queue QNAME acl_hosts=”sol,star”
```

acl_user_enable

When true directs the Server to use the `acl_users` access list for this queue.
Format: boolean (see acl_group_enable)
Default value: disabled

```
Qmgr: set queue QNAME acl_user_enable=true
```

acl_users

List of users allowed or denied the ability to enqueue jobs in this queue. See also `acl_user_enable`.
Format: “[+|-]user[@host][,...]”
Default value: all users allowed

```
Qmgr: set queue QNAME acl_users=”sue,bob@star”
```

enabled

When true, the queue will accept new jobs. When false, the queue is *disabled* and will not accept jobs.
Format: boolean
Default value: disabled

```
Qmgr: set queue QNAME enabled=true
```

from_route_only

When true, this queue will accept jobs only when being routed by the Server from a local routing queue. This is used to force users to submit jobs into a routing queue used to distribute jobs to other queues based on job resource limits.
Format: boolean
Default value: disabled

```
Qmgr: set queue QNAME from_route_only=true
```

max_group_res
max_group_res_soft

The maximum amount of the specified resource that all members of the same group may consume simultaneously, in the specified queue. The named resource can be any valid PBS resource, such as “ncpus”, “mem”, “pmem”, etc. This limit can be specified as either a *hard* or *soft* limit. (See also section 6.3 “Hard versus Soft Limits” on page 52.)
Format: “max_group_res.resource_name=value[,...]”
Format: “max_group_res_soft.resource_name=value[,...]”
Default value: none

```
Qmgr: set queue QNAME max_group_res.mem=1GB
Qmgr: set queue QNAME max_group_res_soft.ncpus=10
```

The first line in the example above sets a normal (e.g. *hard*) limit of 1GB on memory as the aggregate maximum that any group in this queue may consume. The second line in the example illustrates setting a group *soft* limit of 10 CPUs.

max_queuable

The maximum number of jobs allowed to reside in the queue at any given time. Once full, no new jobs will be accepted into the queue.
Format: integer
Default value: infinite

```
Qmgr: set queue QNAME max_queuable=200
```

max_user_res
max_user_res_soft

The maximum amount of the specified resource that any single user may consume. The named resource can be any valid PBS resource, such as “ncpus”, “mem”, “pmem”, etc. This limit can be specified as either a *hard* or *soft* limit. (See also section 6.3 “Hard versus Soft Limits” on page 52.)
Format: “max_user_res.resource_name=value[,...]”
Format: “max_user_res_soft.resource_name=value[,...]”
Default value: none

```
Qmgr: set queue QNAME max_user_res.ncpus=6
Qmgr: set queue QNAME max_user_res_soft.ncpus=3
```

priority

The priority of this queue against other queues of the same type on this Server. (A larger value is higher priority than a smaller value.) May affect job selection for execution/routing.
Format: integer in range of -1024 thru +1024, inclusive
Default value: 0

```
Qmgr: set queue QNAME priority=123
```

queue_type

The type of the queue: execution or route. This parameter must be explicitly set.
Format: “execution”, “e”, “route”, “r”
Default value: none, must be specified

```
Qmgr: set queue QNAME queue_type=route
Qmgr: set queue QNAME queue_type=execution
```

require_cred_enable

Attribute which when true directs the Server to use the credential authentication method specified by `require_cred` for this queue. Requires full Manager privilege to set or alter.
Format: boolean
Default value: false = disabled

```
Qmgr: set queue QNAME require_cred_enable=true
```

require_cred

Specifies the credential type required. All jobs submitted to the named queue without the specified credential will be rejected. Requires full Manager privilege to set or alter.
Format: string: “krb5” or “dce”
Default value: unset

```
Qmgr: set queue QNAME require_cred=krb5
```

resources_default

The list of default resource values which are set as limits for a job residing in this queue and for which the job did not specify a limit. If not set, the default limit for a job is determined by the first of the following parameters which is set: Server’s `resources_default`, queue’s `resources_max`, Server’s `resources_max`. An unset resource is viewed as unlimited. See also section 6.10 “Resource Default/Min/Max Attributes” on page 79.
Format: “resources_default.*resource_name*=*value*”
Default value: none

```
Qmgr: set queue QNAME resources_default.mem=1kb
Qmgr: set queue QNAME resources_default.ncpus=1
```

resources_max

The maximum amount of each resource which can be requested by a single job in this queue. The queue value supersedes any Server

wide maximum limit. See also section 6.10 “Resource Default/Min/Max Attributes” on page 79.
Format: “resources_default.*resource_name*=*value*”
Default value: unset

```
Qmgr: set queue QNAME resources_max.mem=2gb
Qmgr: set queue QNAME resources_max.ncpus=32
```

resources_min

The minimum amount of each resource which can be requested by a single job in this queue. See also section 6.10 “Resource Default/Min/Max Attributes” on page 79.
Format: “resources_min.*resource_name*=*value*”
Default value: unset

```
Qmgr: set queue QNAME resources_min.mem=1kb
Qmgr: set queue QNAME resources_min.ncpus=1
```

started

Jobs may be scheduled for execution from this queue. When false, the queue is considered *stopped.*
Format: boolean
Default value: unset

```
Qmgr: set queue QNAME started=true
```

### 6.5.3 Attributes for execution queues only

checkpoint_min

Specifies the minimum interval of CPU time, in minutes, which is allowed between checkpoints of a job. If a user specifies a time less than this value, this value is used instead.
Format: integer
Default value: unset

```
Qmgr: set queue QNAME checkpoint_min=5
```

kill_delay

The amount of the time delay between the sending of SIGTERM and SIGKILL when a `qdel` command is issued against a running job.
Format: integer seconds
Default value: 2 seconds

```
Qmgr: set queue QNAME kill_delay=5
```

max_running

The maximum number of jobs allowed to be selected from this queue for routing or execution at any given time. For a routing queue, this is enforced by the Server, if set.
Format: integer
Default value: infinite

```
Qmgr: set queue QNAME max_running=16
```

max_user_run

The maximum number of jobs owned by a single user that are allowed to be running from this queue at one time.
Format: integer
Default value: unset

```
Qmgr: set queue QNAME max_user_run=5
```

max_group_run

The maximum number of jobs owned by users in a single group that are allowed to be running from this queue at one time.
Format: integer
Default value: unset

```
Qmgr: set queue QNAME max_group_run=20
```

resources_available

The list of resource and amounts available to jobs running in this queue. The sum of the resource of each type used by all jobs running from this queue cannot exceed the total amount listed here.
Format: "resources_available.*resource_name*=*value*"
Default value: unset

```
Qmgr: set queue QNAME resources_available.mem=1gb
```

### 6.5.4 Attributes for route queues only

route_destinations

The list of destinations to which jobs may be routed, listed in the order that they should be tried. See also section 6.11 "Selective Routing of Jobs into Queues" on page 80.
Format: queue_name[,...]
Default value: none, should be set to at least one destination.

```
Qmgr: set queue QNAME route_destinations=QueueTwo
```

route_held_jobs

If true, jobs with a hold type set may be routed from this queue. If false, held jobs are not to be routed.
Format: boolean
Default value: false = disabled

```
Qmgr: set queue QNAME route_held_jobs=true
```

route_lifetime

The maximum time a job is allowed to exist in a routing queue. If the job cannot be routed in this amount of time, the job is aborted. If unset, the lifetime is infinite.
Format: integer seconds
Default value: infinite

```
Qmgr: set queue QNAME route_lifetime=600
```

route_retry_time — Time delay between route retries. Typically used when the network between servers is down.
Format: integer seconds
Default value: 30

```
Qmgr: set queue QNAME route_retry_time=120
```

route_waiting_jobs — If true, jobs with a future `execution_time` attribute may be routed from this queue. If false, they are not to be routed.
Format: boolean
Default value: false = disabled

```
Qmgr: set queue QNAME route_waiting_jobs=true
```

### 6.5.5 Read-only parameters of queues

These parameters are visible to client commands, but cannot be changed by them.

hasnodes — If true, indicates that the queue has nodes associated with it.

total_jobs — The number of jobs currently residing in the queue.

state_count — Lists the number of jobs in each state within the queue.

resources_assigned — Total amount of resources allocated to jobs running from this queue.

## 6.6 Nodes

Where jobs will be run is determined by an interaction between the Scheduler and the Server. This interaction is affected by the contents of the PBS `nodes` file (*`PBS_HOME/server_priv/nodes`*), and the system configuration onto which you are deploying PBS. Without this list of nodes, the Server will not establish a communication stream with the MOM(s) and MOM will be unable to report information about running jobs or notify the Server when jobs complete. If the PBS configuration consists of a single host on which the Server and MOM are both running, all the jobs will run there.

If your complex has more than one execution host, then distributing jobs across the various hosts is a matter of the Scheduler determining on which host to place a selected job.

**Important:** Regardless of node type, each node must be defined in the

Server's nodes file, *PBS_HOME*/server_priv/nodes. Time-shared nodes have :ts appended to their node name. Cluster nodes have no name suffix.

### 6.6.1 PBS Nodes File

A basic nodes file is created for you by the install procedure. This file contains only the name of the host from which the install was run and set as a time-shared host. If you have more than one host in your PBS cluster or you are not planning on running jobs on the Server's host, you need to edit the list of nodes to reflect your site.

You may edit the nodes list in one of two ways. If the Server is not running, you may directly edit the nodes file with a text editor. If the Server is running, you should use qmgr to edit the list of nodes (for details see section 6.6.2 "Creating or Modifying Nodes" on page 70. The Server's *node list* is in the file *PBS_HOME*/server_priv/nodes. This is a simple text file, where each line of the file has the form:

```
    node_name[:ts] [attributes]
```

The *node name* is the network name of the node (host name), it does not have to be fully qualified (in fact it is best if it is as short as possible). The optional :ts appended to the name indicates that the node is a time-shared node.

Nodes can have attributes associated with them. Attributes come in three types: properties, name=value pairs, and name.resource=value pairs. A *property* is nothing more than a string of alphanumeric characters (first character must be alphabetic) established by the Administrator, having no special meaning to PBS itself. A user's job can specify that the node(s) used for the job have a certain set of properties. Properties are useful to create classes of nodes, which jobs can request.

Within the nodes file the expression np=*NUMBER* can be used in place of the cumbersome expression resources_available.ncpus=NUMBER, where *NUMBER* is a specific integer. This declares the number of virtual processors (VPs) associated with the node. This expression will allow the node to be allocated up to *NUMBER* times to one job or more than one job. If np=*NUMBER* is not specified for a cluster node, the number of VPs for the node defaults to "1". If the following parameters are not explicitly set, they will take the value provided by MOM. But if they are explicitly set, that setting will be carried forth across Server restarts.

They are:

```
resources_available.ncpus
resources_available.arch
resources_available.mem
```

The following is an example of a nodes file for a cluster called “planets”:

```
# The first set of nodes are cluster nodes.
# Note that the properties are provided to
# logically group certain nodes together.
# The last node is a time-shared node.
#
mercury    inner moonless
venus      inner moonless np=1
earth      inner np=1
mars       inner np=2
jupiter    outer np=18
saturn     outer np=16
uranus     outer np=14
neptune    outer np=12
pluto:ts
```

### 6.6.2 Creating or Modifying Nodes

After `pbs_server` is started, the node list may be entered or modified via the `qmgr` command. For example, to add a new node, use “create” `qmgr` command:

```
create node node_name [attribute=value]
```

where the attributes and their associated possible values are shown in the table below.

The `busy` state is set by the execution daemon, `pbs_mom`, when a load-average threshold is reached on the node, or the Administrator has configured the node for cycle-harvesting and its owner is using the node. See *max_load* in MOM’s config file (“Static Resources” on page 106). `The job-exclusive` and `job-sharing` states are set when jobs are running on the node.

**Important:** All comma separated strings must be enclosed in quotes.

Below are several examples of creating nodes via `qmgr`.

```
% qmgr
Qmgr: create node mars resources_available=2,ntype=cluster
Qmgr: create node venus properties=”inner,moonless”
```

**Modify nodes:** Once a node has been created, its attributes and/or properties can be modified using the following `qmgr` syntax:

`set node` *`node_name`* `[attribute[+|-]=value]`

where attributes are the same as for `create`. For example:

```
% qmgr
Qmgr: set node mars properties=inner
Qmgr: set node mars properties+=haslife
```

**Delete nodes:** Nodes can be deleted via `qmgr` as well, using the `delete node` syntax, as the following example shows:

```
% qmgr
Qmgr: delete node mars
Qmgr: delete node pluto
```

## 6.7 Node Configuration Parameters

A node has the following configuration parameters:

comment General comment, e.g. the reason the node is marked down or off-line; can be set by a PBS Manager or Operator. If this parameter is not explicitly set, the PBS Server will use it to pass information about the node status, specifically why the node is down. If explicitly set by the Administrator, it will not be modified by the Server.
Format: string
`Qmgr: set node` ***`MyNode`*** **`comment=”Down until 5pm”`**

max_running The maximum number of jobs allowed to be run on this node at any given time.
Format: integer
`Qmgr: set node` ***`MyNode`*** **`max_running=22`**

max_user_run The maximum number of jobs owned by a single user that are allowed to be run on this node at one time.
Format: integer

```
Qmgr: set node MyNode max_user_run=4
```

max_group_run The maximum number of jobs owned by any users in a single group that are allowed to be run on this node at one time.
Format: integer

```
Qmgr: set node MyNode max_group_run=8
```

no_multinode_jobs If this parameter is set true, jobs requesting more than one node will not be run on this node. This parameter can be used in conjunction with Cycle Harvesting on workstations to prevent a select set of workstations from being used when a busy workstation might interfere with the execution of jobs that require more than one node.
Format: boolean

```
Qmgr: set node MyNode no_multinode_jobs=true
```

ntype Defines the type of the node. Currently there are three type of nodes:

| **ntype** | **Description** |
|---|---|
| time-shared | Time-shared nodes are assumed to run multiple jobs; the placement of the jobs among the time-shared nodes is by the site policy defined in the Job Scheduler. |
| cluster | Cluster nodes are typically used for running parallel jobs. A cluster node is allocated to one or more jobs by the Server. Allocation is at the CPU level. A cluster node with one CPU declared can only be allocated to one job. A cluster node with N CPUs declared can be allocated one CPU to N jobs or N CPUs to one job are any combination there of. |
| globus | A special `pbs_mom` is running to transfer jobs to the Globus distributed system. |

Format: string

```
Qmgr: set node MyNode ntype=cluster
```

properties Any alphanumeric string or comma separated set of strings,

starting with an alphabetic character.
Format: string

```
Qmgr: set node MyNode properties=”red,blue”
```

queue

Name of an execution queue (if any) associated with the node. Only jobs from the named queue will be run on associated node. Note: a node can be associated with at most one queue.
Format: queue specification

```
Qmgr: set node MyNode queue=MyQueue
```

resources_available

List of resources available on node.
Format: resource list

state

Shows or sets the state of the node. Certain state values, marked with an * in the following list, may be set by the Manager or Operator, the other states are set internally.

| | |
|---|---|
| free * | Node is up and capable of accepting additional job(s). |
| offline * | Node has been marked by operator or administrator as unusable. |
| down | Node is not responding to queries from the Server. |
| job-busy | All CPUs on the node are allocated to jobs. |
| job-sharing | All CPUs on the node are allocated to jobs, at least one job is willing to share its allocated CPUs with other jobs that are also willing to share. |
| job-exclusive | The entire node has been exclusively allocated to one job at the job’s request. |
| busy | The node is reporting a load average greater than the configured high water value. |
| state-unknown | The Server has never been able to contact the node. Either `pbs_mom` is not running on the node, the node hardware is down, or there is a network problem. |

Format: string

```
Qmgr: set node MyNode state=offline
```

A node has the following read-only parameters:

pcpus Shows the number of physical CPUs on the node, which determine the number of licenses required for that node.

license Indicates the node "license state" as a single character, according to the following table:

| | |
|---|---|
| u | unlicensed |
| l | licensed with a node lock (fixed) license |
| f | licensed with a floating license |

resources_assigned List of resources in use on node.
Format: resource list

reservations List of reservations pending on the node.
Format: reservation specification

### 6.7.1 Setting Node Limits

It is possible to set limits on queues (and the Server) as to how many nodes a job can request. The `nodes` resource itself is a text string and difficult to limit. Thus, `nodect` (node count) is set by the Server to the integer number of nodes desired by the user as declared in the `nodes` resource specification. That declaration is parsed and the resulting total number of nodes is set in `nodect`. This is useful when an Administrator wishes to place an integer limit, `resources_min` or `resources_max` on the number of nodes used by a job entering a queue. Based on the earlier example of declaring nodes, if a user requested a nodes specification of: `3:inner+2:outer`, then `nodect` would get set to 5 (i.e. 3+2). Another read-only job attribute, `neednodes`, is initially set by the Server to the same value as `nodes`. `Neednodes` may be modified by the job Scheduler for special policies. The contents of `neednodes` determines which nodes are actually assigned to the job. `neednodes` is visible to the Manager but not to an unprivileged user.

If you wish to set up a queue default value for `nodes` (a value to which the resource is set if the user does not supply one), corresponding default values must also be set for `nodect` and `neednodes`. For example:

```
% qmgr
Qmgr: set queue small resources_default.nodes=1:inner
Qmgr: set queue small resources_default.nodect=1
Qmgr: set queue small resources_default.neednodes=1:inner
```

Minimum and maximum limits are set for “nodect” only. For example:

```
% qmgr
Qmgr: set queue small resources_min.nodect=1
Qmgr: set queue small resources_max.nodect=15
Qmgr:
```

**Important:** Minimum and maximum values must **not** be set for either nodes or neednodes as their value are strings.

### 6.7.2 Node Specification Information

This section provides additional information about working with the PBS nodes file and what guides the selection of a set of nodes for the job. This is based on Server configuration and the user’s actual job specification. (Detailed information on node specification, from the user perspective, is provided in Chapter 9 of the **PBS Pro User Guide** under the heading “Node Specification Syntax”.)

Step 1 If a single specific host is named in the request to run a given job and the host is specified in the nodes file as a *time-shared* host, the Server will attempt to run the job on that host.

Step 2 If either:

(a) a specific host is named in the Run Job and the named node does not appear in the Server’s nodes file as a time-shared host;

or

(b) a “+” separated list of hosts [or node properties] is specified in the Run Job request;

then, the Scheduler attempts to allocate one (or more as requested) virtual processor on the named *cluster* node(s) named in the job. All of the named nodes have to appear in the Server’s nodes file. If the allocation succeeds, the job [shell script] is run on the first of the nodes allocated.

Step 3 If no location was specified on the Run Job request, but the job requests nodes, then the required number of virtual processors on cluster nodes which match the request are allocated if possible. If the allocation succeeds, the job is run on the node allocated to match the first specification in the node request. Note, the Scheduler may modify the job's original node request, see the job attribute `neednodes.`

Step 4 If the Server parameter `default_node` is set, its value is used. If it matches the name of a time-shared node, the job is run on that node. If the value can be mapped to a set of one or more free cluster nodes, they are allocated to the job.

Step 5 If `default_node` is not set, and at least one time-shared node is defined, that node is used. If more than one is defined, the first is selected for the job.

Step 6 The last choice is to act as if the job has requested `1#shared`. The job will have allocated to it an existing job-shared VP, or if none exists, then a free VP is allocated as job-shared.

The `exec_host` string and the `runjob` destination string is now of the form:

```
nodename/P[*C][+nodename/P[*C]...]
```

where P is the process index ranging from 0 to P-1, the number of requested processes on the node; and C is the number of CPUs per process. For example, a request of

```
-l nodes=alpha:ppn=2:ncpus=2+beta
```

would result in an execution host string of

```
alpha/0*2+beta/0+alpha/1*2
```

### 6.7.3 Node Comments

Nodes have a "comment" attribute which can be used to display information about that node. If the comment attribute has not been explicitly set by the PBS Manager and the node is down, it will be used by the PBS Server to display the reason the node was marked down. If the Manager has explicitly set the parameter, the Server will not overwrite the comment. The comment attribute may be set via the `qmgr` command:

```
% qmgr
Qmgr: set node pluto comment=”node will be up at 5pm”
```

## 6.8 Node Grouping

The *node grouping* feature allows the PBS Administrator to partition a PBS cluster into logical node groups and permit users to run multi-node jobs within one partition. In general, when a job is put into execution it will be assigned to nodes that all belong to the same partition, as chosen by the Scheduler. Similarly, when an advance reservation is confirmed, it is given a set of nodes all of which belong to a single partition. This is of particular interest to those sites which have multiple high-performance network switches within a large cluster. Jobs should be assigned onto the same switch, rather than given nodes that span multiple switches (unless that is the only way to ever satisfy the job's or reservation's nodes request).

The partitioning is implemented by using a non-consumable node resource. The Administrator defines a particular node resource as the basis of the grouping. All nodes having the same value for the grouping resource are considered to be in the same partition.

The Administrator enables node grouping via the qmgr command. There are two Server parameters associated with node grouping. One enables node grouping and the other specifies the name of the grouping resource.

```
node_group_enable = TRUE | FALSE
node_group_key = resource_name
```

If `node_group_enable` is not set it defaults to false. If `node_group_enable` is set and `node_group_key` is not, then node grouping will be disabled as well. It is not permitted to set `node_group_key` to an invalid resource name or a non-string resource.

**Important:** Note that these are Server object attributes only and therefore cannot be set at the queue level.

If the resource ***resource_name*** is not already defined, the Administrator should define this resource in the Server's *PBS_HOME/server_priv/resourcedef* file (and then restart the Server). The resource should be specified as type *string* and there should be no flags specified. Defining a new resource is discussed in detail in section 9.2 "Defining New Custom Resources" on page 150.

The nodes get partitioned according to the actual value given to the grouping resource on each node, i.e. the value of *resources_available.RES* on the node. This means that the Administrator needs to use qmgr to add the grouping resource to the node's *resources_available* parameter and set its value (see below). Not all nodes need to have the grouping resource appear in their *resources_available* parameter, just those that the Administrator wants to participate in the node grouping.

For example, say you have a cluster with two high-performance switches each with half the nodes connected to it. Now you want to set up node grouping so that jobs will be scheduled only onto the same switch.

First, create a new resource called “switch”, and define it in the Server’s *PBS_HOME/server_priv/resourcedef* file, as follows:

```
switch    type=string
```

Next, we need to enable node grouping, specify the resource to use for node grouping (i.e. our new resource “switch”), and assign to “switch” a node-specific value. We can do all of this via the qmgr command, as the following example shows.

```
% qmgr
Qmgr: set server node_group_enable=true
Qmgr: set server node_group_key=switch
Qmgr: set node node1 resources_available.switch=A
Qmgr: set node node2 resources_available.switch=A
Qmgr: set node node3 resources_available.switch=A
Qmgr: set node node4 resources_available.switch=B
Qmgr: set node node5 resources_available.switch=B
Qmgr: set node node6 resources_available.switch=B
```

For ease of use with qmgr, this attribute can be set on many nodes at once. This is done through the use of the "active" qmgr command. You can use this command to create a set of nodes, and after the set is created all node commands will act upon this set if no specific node name is specified. The usage is as following:

```
     active node "comma,separated,list,of,nodes"
```

There can be no spaces between the list of nodes and commas. After the set is defined, you can type commands by leaving the node name off and the command will act upon the entire set. The following example is equivalent to the longer example above.

```
% qmgr
Qmgr: active node "node1,node2,node3"
Qmgr: set node resources_available.switch=A
Qmgr: active node "node4,node5,node6"
Qmgr: set node resources_available.switch=B
```

## 6.9 PBS Resources

PBS has a standard set of "resources" which may be specified as a job requirement. Common examples of the predefined standard resources are:

| | |
|---|---|
| cput | amount of CPU time |
| mem | amount of real memory |
| ncpus | number of CPUs |
| nodes | number and type of execution nodes |
| walltime | amount of real clock time |

**Important:** A complete listing of available PBS resources is given in section 4.7 “PBS System Resources” of the **PBS Pro User Guide**.

Depending on site policy, the required resources may be considered against the availability of the resources to determine ordering of jobs for execution and placement of jobs on an execution host.

### 6.9.1 Defining New Resources

It is possible for the PBS Manager to define new resources within PBS Pro. Once created, jobs may request these new resources and the Scheduler can be directed to consider the new resources in the scheduling policy. For detailed discussion of this capability, see Chapter 9, “Customizing PBS Resources” on page 149

## 6.10 Resource Default/Min/Max Attributes

Default, minimum and maximum queue and Server limits work with numeric valued resources, including time and size values. Generally, they do not work with string valued resources because of character comparison order. However, setting the `min` and `max` to the same value to force an exact match will work even for string valued resources, as the following example shows.

```
% qmgr
Qmgr: set queue big resources_max.arch=unicos8
Qmgr: set queue big resources_min.arch=unicos8
```

The above example can be used to limit jobs entering queue `big` to those specifying `arch=unicos8`. Again, remember that if `arch` is not specified by the job, the tests pass automatically and the job will be accepted into the queue.

Note however that if a job does not request a specific resource, then the enforcement of the corresponding limit will not occur. To prevent such cases, the Administrator is advised to set queue and/or server defaults. The following example sets a maximum limit on the amount of cputime to 24 hours; but it also has a default of 1 hour, to catch any jobs that do not specify a `cput` resource request.

```
% qmgr
Qmgr: set queue big resources_max.cput=24:00:00
Qmgr: set queue big resources_default.cput=1:00:00
```

With this configuration, any job that requests more than 24 hours will be rejected. Any job requesting 24 hours or less will be accepted, but will have this limit enforced. And any job that does not specify a `cput` request will receive a default of 1 hour, which will also be enforced.

## 6.11 Selective Routing of Jobs into Queues

Often it is desirable to route jobs to various queues on a Server, or even between Servers, based on the resource requirements of the jobs. The queue *resources_min* and *resources_max* parameters discussed above make this selective routing possible. As an example, let us assume you wish to establish two execution queues, one for short jobs of less than one minute cpu time, and the other for long running jobs of one minute or longer. Let's call them `short` and `long`. Apply the `resources_min` and `resources_max` parameter as follows:

```
% qmgr
Qmgr: set queue short resources_max.cput=59
Qmgr: set queue long resources_min.cput=60
```

When a job is being enqueued, it’s requested resource list is tested against the queue limits: `resources_min` <= job_requirement <= `resources_max`. If the resource test fails, the job is not accepted into the queue. Hence, a job asking for 20 seconds of cpu time would be accepted into queue `short` but not into queue `long`.

**Important:** Note, if the `min` and `max` limits are equal, only that exact value will pass the test.

**Important:** The boolean resource specification "`resc`" cannot be used as a selection criteria for choosing a route queue destination. This is because `resc` is used only for selection of the nodes on which to execute the job.

You may wish to set up a routing queue to direct jobs into the queues with resource limits. For example:

```
% qmgr
Qmgr: create queue funnel queue_type=route
Qmgr: set queue funnel route_destinations =”short,long”
Qmgr: set server default_queue=funnel
```

A job will end up in either `short` or `long` depending on its cpu time request.

**Important:** You should always list the destination queues in order of the most restrictive first as the first queue which meets the job’s requirements will be its destination (assuming that queue is enabled).

Extending the above example to three queues:

```
% qmgr
Qmgr: set queue short resources_max.cput=59
Qmgr: set queue long resources_min.cput=1:00
Qmgr: set queue long resources_max.cput=1:00:00
Qmgr: create queue huge queue_type=execution
Qmgr: set queue funnel route_destinations=”short,long,huge”
Qmgr: set server default_queue=funnel
```

A job asking for 20 minutes (20:00) of cpu time will be placed into queue `long`. A job asking for 1 hour and 10 minutes (1:10:00) will end up in queue `huge`, because it was not accepted into the first two queues, and nothing prevented it from being accepted into `huge`.

**Important:** If a test is being made on a resource as shown with `cput` above, and a job does not specify that resource item (it does not appear in the `-l resource=`*`valuelist`* on the `qsub` command, the test will pass. In the above case, a job without a cpu time limit will be allowed into queue `short.` For this reason, together with the fact that an unset limit is considered to be an infinite limit, you may wish to add a default value to the queues or to the Server.

```
% qmgr
Qmgr: set queue short resources_default.cput=40
or
Qmgr: set server resources_default.cput=40
```

Either of these examples will ensure that a job without a cpu time specification is limited to 40 seconds. A `resources_default` parameter at a queue level only applies to jobs in that queue.

**Important:** Be aware of several important facts:

If a default value is assigned, it is done so after the tests against `min` and `max`. Default values assigned to a job from a queue `resources_default` are not carried with the job if the job moves to another queue. Those resource limits becomes unset as when the job was specified. If the new queue specifies default values, those values are assigned to the job while it is in the new queue. Server level default values are applied if there is no queue level default.

## 6.12 SGI Weightless CPU Support

Submitting a job and requesting `-l ncpus=0` is now legal. In a non-cpuset SGI IRIX 6.x environment, the job's kernel scheduling priority will be set "weightless". There will no allocation at the Server, Queue, or Node level of CPUs; i.e. `resources_assigned.ncpus` will not be incremented for this job.

**Important:** Because `ncpus=0` has no useful effect on any other system and can result in allowing too many jobs to be run, it is **strongly** recommended that jobs not be allowed to be submitted with `ncpus=0`. This may be done by setting a Server level resource default and a resources minimum via the `qmgr` command:

```
% qmgr
Qmgr: set server resources_default.ncpus=1
Qmgr: set queue q1 resources_min.ncpus=1
Qmgr: set queue q2 resources_min.ncpus=1
```

## 6.13 Recording Server Configuration

If you should you wish to record the configuration of a PBS Server for re-use later, you may use the `print` subcommand of `qmgr`(8B). For example,

```
# qmgr -c “print server” > /tmp/server.out
# qmgr -c “print node @ default” > /tmp/nodes.out
```

will record in the file `/tmp/server.out` the `qmgr` subcommands required to recreate the current configuration including the queues. The second file generated above will contain the nodes and all the node properties. The commands could be read back into `qmgr` via standard input:

```
# qmgr < /tmp/server.out
# qmgr < /tmp/nodes.out
```

## 6.14 Server Support for Globus

If Globus support is enabled, then an entry must be manually entered into the PBS nodes file (*PBS_HOME*/`server_priv/nodes`) with `:gl` appended to the name. This is the only case in which two nodes may be defined with the same node name. One may be a Globus node (MOM), and the other a non-Globus node.

**Important:** If you run both a Globus MOM and a normal MOM on the same site, the normal PBS MOM must be listed first in your nodes file. If not, some scheduling anomalies could appear.

## 6.15 Configuring PBS Redundancy and Failover

The redundancy-failover feature of PBS Pro provides the capability for a backup Server to assume the work load of a failed Server, thus eliminating the one single point of failure in PBS Pro. If the Primary Server fails due to a hardware or software error, the Secondary Server will take over the work load and communications automatically. No work is lost in the transition of control.

The following terms are used in this manual section: *Active Server* is the currently running PBS Pro Server process. *Primary Server* refers to the Server process which under normal circumstances is the active Server. *Secondary Server* is a Server which is inactive (idle) but which will become active if the Primary Server fails.

### 6.15.1 Failover Requirements

The following requirements must be met to provide a reliable fail-over service:

1. The Primary and Secondary Servers must be run on different hosts. At the current time, the feature allows only one Secondary Server.

2. The Primary and Secondary Server hosts must be the same architecture.

3. Both the Primary and Secondary Server host must be able to communicate over the network with all execution hosts where `pbs_mom` is running.

4. The directory and subdirectories used by the Server, `PBS_HOME`, must be on a file system which is available to both the Primary and Secondary Servers. Both must have read/write access as root on UNIX or "Administrator" on Windows.

   The best solution is a high availability file server, such as a dual ported RAID system connected to both the Primary and Secondary Server hosts. A workable, but not ideal, solution is to use an NFS file server with the file system exported to and mounted by both the Primary and Secondary Server hosts. The file server must not be either the Primary or Secondary Server host, as that introduces a single point of failure effecting both Servers.

In a Microsoft Windows environment, a workable solution is to use the network share facility; that is, use as `PBS_HOME` a directory on a remote windows host that is shared among primary server and secondary hosts.

**Important:** Note, that a failure of the NFS server will prevent PBS from being able to continue.

5. A MOM, `pbs_mom`, may run on either the Primary or the Secondary hosts, or both. It is strongly recommended that the directory used for "`mom_priv`" be on a local, non-shared, file system. It is critical that the two MOMs do not share the same directory. This can be accomplished by using the `-d` option when starting `pbs_mom`, or with the `PBS_MOM_HOME` entry in `pbs.conf`. The `PBS_MOM_HOME` entry points to a directory which has the following contents:

Under UNIX:

| Directory Contents | Description |
|---|---|
| `aux` | Directory with permission 0755 |
| `checkpoint` | Directory with permission 0700 |
| `mom_logs` | Directory with permission 0755 |
| `mom_priv` | Directory with permission 0755 |
| `mom_priv/jobs` | Subdirectory with permission 0755 |
| `mom_priv/config` | File with permission 0644 |
| `pbs_environment` | File with permission 0644 |
| `spool` | Directory with permission 1777 (drwxrwxrwt) |
| `undelivered` | Directory with permission 1777 (drwxrwxrwt) |

Under Microsoft Windows:

| Directory Contents | Description |
|---|---|
| `auxiliary` | Directory with full access to Administrators and read-only access to Everyone |
| `checkpoint` | Directory with full access only to Administrators |
| `mom_logs` | Directory with full access to Administrators and read-only access to Everyone |
| `mom_priv` | Directory with full access to Administrators and read-only access to Everyone |
| `mom_priv/jobs` | Subdirectory with full access to Administrators and read-only access to Everyone |
| `mom_priv/config` | File with full access-only to Administrators |
| `pbs_environment` | File with full access to Administrators and read-only to Everyone |
| `spool` | Directory with full access to Everyone |
| `undelivered` | Directory with full access to Everyone |

If `PBS_MOM_HOME` is present in `pbs.conf`, `pbs_mom` will use that directory for its "home" instead of `PBS_HOME`.

6. The version of the PBS Pro commands installed everywhere must match the version of the Server, in order to provide for automatic switching in case of failover.

### 6.15.2 Failover Configuration for UNIX

The steps below outline the process for general fail-over setup, and should be sufficient for configuration under UNIX. If you wish to configure PBS failover using Microsoft Windows 2000 and XP-based systems, you are advised to review the material below to familiarize yourself with the general failover setup, and then to read the "Failover Configuration on Windows" section below for the actual steps to follow.

To configure PBS Pro for a failover operation, follow these steps:

1. Select two systems of the same architecture to be the Primary and Secondary Server systems.

   For Windows, please use Windows 2000 Pro or Windows 2000 Server machines.

2. Configure a file system (or at least a directory) that is read/write accessible by root (UNIX) or “Administrator” (Windows) from both systems. If an NFS file system is used, root or Administrator must have access to read and write as “root” or as “Administrators” on both systems. Under UNIX, the directory tree must meet the security requirements of PBS:

   Each parent directory above `PBS_HOME` must be owned by “root” (“Administrators”) and be writable only by “root” (Administrators”).

3. Install PBS Pro on both systems, specifying the shared file system location for the `PBS_HOME` directory. Do not start either `pbs_server` at this point.

4. Modify `pbs.conf` on both systems, as follows:

4a. Change `PBS_SERVER` on both systems to have the same value. The value must be a valid hostname. The short form (alias) of the Primary Server host name is preferred.

4b. Add the following entries to both `pbs.conf` files, they must have the same value in both files:

```
PBS_PRIMARY=primary_host
PBS_SECONDARY=secondary_host
```

where "primary_host" is the primary formal name of the Primary Server host, and "secondary_host" is the primary formal name of the Secondary Server host. It is important that these entries be correct and distinct as they are used by the Servers to determine their status at start up.

These entries must also be added to pbs.conf on any system on which the PBS commands are installed, and on any compute node in the cluster.

4c. Ensure that the PBS_HOME entry on both systems names the shared PBS directory, using the specific path on that host.

4d. On the Secondary host, modify pbs.conf to not start the Scheduler by setting

```
PBS_START_SCHED=0
```

If needed, the Secondary Server will start a Scheduler itself.

5. If you are running a pbs_mom on both the Primary and Secondary Server hosts, modify the PBS init script under UNIX or the Start-> Control Panel-> Administrator Tools-> Services-> PBS_MOM dialog under Windows to ensure that pbs_mom on one of the system is started with a different PBS_HOME using the -d option. You will need to replicate the PBS_HOME directory structure at the place specified by the PBS_HOME. It is not recommended to run pbs_mom on both systems.

6. PBS has a standard delay time from detection of possible Primary Server failure until the Secondary Server takes over. This is discussed in more detail in the "Normal Operation" section below. If your network is very reliable, you may wish to decrease this delay. If your network is unreliable, you may wish to increase this delay. The default delay is 30 seconds. To change the delay, use the "-F seconds" option on the Secondary Server's command line.

7. In the typical PBS environment, the Job Scheduler, pbs_sched, is run on the same host as the PBS Server. The Secondary Server will start a Scheduler on its (secondary) host only if the Secondary Server cannot contact the Scheduler on the primary host. This is handled automatically; see the discussion under "Normal Operation" section below.

8. Once the Primary Server is started, use the qmgr command to set or modify the Server's "mail_from" parameter to an email

address which is monitored. If the Primary Server fails and the Secondary becomes active, an email notification of the event will be sent to the "mail_from" address.

### 6.15.3 Failover Configuration for Windows

The following illustrates how PBS can be setup on Windows with the Server failover capability using the network share facility. That is, the primary Server/Scheduler, secondary Server/Scheduler will share a PBS_HOME directory that is located on a network share file system on a remote host.

The scenario is that we have a primary pbs_server to run on hostA, a secondary Server to run on hostB, and the shared PBS_HOME be setup on hostC using Windows network share facility. hostA, hostB, and hostC must be setup on a Windows 2000 Pro or Windows 2000 Server platform. Windows XP has limitations on number of outgoing network connections as well as issues with accessing network shared folders from a service application which proved problematic with the failover setup.

1. Install PBS Windows on hostA and hostB accepting the default destination location of "C:\Program Files\PBS Pro".

2. Then stop all the PBS services on both hostA and hostB:

```
net stop pbs_server
net stop pbs_mom
net stop pbs_sched
net stop pbs_rshd
```

3. Perform the following on both hostA and hostB:

a. Go to Start Menu->Control Panel->Administrative Tools->Services.

b. Select PBS_SERVER, choose the "Log On" tab, and click to select the checkbutton "Logon as this account:", specifying "Administrator" and entering its password. (However, if you used a domain admin account to install PBS, please specify this same account and its password here if not the Administrator account.)

c. Repeat step b. for PBS_SCHED.

This configures the pbs_server and pbs_sched services to be run under Administrator instead of Local System Account (SYSTEM) so that pbs_server and pbs_sched can access shared folder PBS_HOME; SYSTEM is not permitted to access network resources.

4. Now configure a shared PBS_HOME by doing the following:

a. Go to hostC; create a folder named e.g., C:\pbs_home. If you installed PBS using a domain admin account, be sure to create the folder using the same account. Otherwise, PBS may have permission problems accessing this shared folder.

b. Using Windows Explorer, right click select the C:\pbs_home file, and choose "Properties".

c. Then select the "Sharing" tab, and click the checkbutton that says "Share this folder"; specify "Full Control" permission for "Administrators"; and remove Everyone access from the Permission list.

d. Using Tools->Map Network Drive dialog in "My Computer" from hostA and hostB, map \\hostC\pbs_home as a network drive P:.

5. Next specify PBS_HOME for primary pbs_server on hostA and secondary Server on hostB by running the following on both hosts:

```
pbs-config-add "PBS_HOME=P:"
```

Now on hostA, copy the files from the local PBS home directory onto the shared PBS_HOME as follows:

```
xcopy /o /e "\Program Files\PBS Pro\home" P:
```

6. pbs_mom's home directory must still be on a local directory because it will continue to run under the SYSTEM account. It needs to run under this special account so that MOM inherits

the privilege that allows creation of user security tokens.

Set up a local PBS_MOM_HOME by running the following on both hosts:

**pbs-config-add "PBS_MOM_HOME=C:\Program Files\PBS Pro\home"**

7. Now create references to primary Server name and secondary Server name in the pbs.conf file by running on both hosts:

```
pbs-config-add "PBS_SERVER=hostA"
pbs-config-add "PBS_PRIMARY=hostA"
pbs-config-add "PBS_SECONDARY=hostB"
```

8. Now start all the PBS services on hostA:

```
net start pbs_mom
net start pbs_server
net start pbs_sched
net start pbs_rshd
```

9. Start the failover Server on hostB:

```
net start pbs_server
```

It's normal to get the following message:

```
"PBS_SERVER could not be started"
```

This is because the failover Server is inactive waiting for the primary Server to go down. If you need to specify a delay on how long the secondary Server will wait for the primary Server to be down before taking over, then you use Start Menu->Control Panel->Administrative Tools->Services, choosing PBS_SERVER, and specify under the "Start Parameters" entry box the value,

```
"-F <delay_secs>"
```

Then restart the secondary pbs_server. Keep in mind that the Services dialog does not remember the "Start Parameters" value for future restarts. The old default delay value will be in effect on the next restart.

10. Set the managers list on the primary Server so that when the secondary Server takes over, you can still do privilege tasks under the Administrator account or from a peer `pbs_server`:

```
qmgr> set server managers = "Administrator@*,SYSTEM@*”
```

### 6.15.4 Failover: Normal Operation

Either the Primary Server and/or the Secondary Server may be started by hand or started via the system `init.d` script under UNIX, or using the Services facility under Windows. If you are starting the Secondary Server from the `init.d` script (UNIX only) and wish the change the failover delay, be sure to add the `-F` option to the `pbs_server`’s entry in the `init.d` script. Under windows, specify the -F as a start parameter given by the Start-> Control Panel-> Administrator Tools-> Services-> PBS_SERVER dialog.

It does not matter in which order the Primary and Secondary Servers are started.

**Important:** If the primary or secondary Server fails to start with the error:

```
another server running
```

then check for the following conditions:

1. There may be lock files (`server.lock`, `server.lock.secondary`) left in *PBS_HOME/*`server_priv` that need to be removed,

2 The RPC `lockd` daemon may not be running. For instance, on an IRIX system, you can manually start this daemon by running as root:

```
/usr/etc/rpc.lockd
```

When the Primary and Secondary Servers are initiated, the Secondary Server will periodically attempt to connect to the Primary. Once connected, it will send a request to "register" itself as the Secondary. The Primary will reply with information to allow the Secondary to use the license keys should it become active.

**Important:** Backup license keys or keys tied to the Secondary host are not required.

The Primary Server will then send "handshake" messages every few seconds to inform the Secondary Server that the Primary is alive. If the handshake messages are not received for the "take over" delay period, the Secondary will make one final attempt to reconnect to the Primary before becoming active. If the "take over" delay time is long, there may be a period, up to that amount of time, when clients cannot connect to either Server. If the delay is too short and there are transient network failures, then Secondary Server may attempt to take over while the Primary is still active.

While the Primary is active and the Secondary Server is inactive, the Secondary Server will not respond to any network connection attempts. Therefore, you cannot status the Secondary Server to determine if it is up.

If the Secondary Server becomes active, it will send email to the address specified in the Server parameter "`mail_from`". The Secondary will inform the execution daemons, `pbs_mom`, on the configured nodes that it has taken over. The Secondary will attempt to connect to the Scheduler on the Primary host. If it is unable to do so, the Secondary Server will start a Scheduler on its host. The Secondary Server will then start responding to network connections and accepting requests from client commands such as `qstat` and `qsub`.

JobIDs will be identical regardless of which Server was running when the job was created, and will contain the name specified by `PBS_SERVER` in `pbs.conf`.

In addition to the email sent when a Secondary Server becomes active, there is one other method to determine which Server is running. The output of a "`qstat -Bf`" command includes the "`server_host`" parameter whose value is the name of the host on which the Server is running.

When a user issues a PBS command directed to a Server that matches the name given by `PBS_SERVER`, the command will normally attempt to connect to the Primary Server. If it is unable to connect to the Primary Server, the command will attempt to connect to the Secondary Server (if one is configured). If this connection is successful, then the command will create a file referencing the user executing the command. (Under UNIX, the file is named "`/tmp/.pbsrc.UID`" where "UID" is the user id; under Windows the file is named `%TEMP\.pbsrc.USERNAME` where “USERNAME” is the user login name.) Any future command execution will detect the presence of that file and attempt to connect to the Secondary Server first. This eliminates the delay in attempting to connect to the down

Server. If the command cannot connect to the Secondary Server, and can connect to the Primary, the command will remove the above referenced file.

### 6.15.5 Failover: Manual Shutdown

Any time the Primary Server exits, because of a fault, or because it was told to shut down by a signal or the `qterm` command, the Secondary Server will become active.

If you wish to shut down the Primary Server and not have the Secondary Server become active, you must either:

1. Use the `-f` option on the `qterm` command. This causes the Secondary Server to exit as well as the Primary; or

2. Use the `-i` option on the `qterm` command, this causes the Secondary Server to remain running but inactive (stand by state); or

3. Manually kill the Secondary Server before terminating the Primary Server (via sending any of `SIGKILL`, `SIGTERM`, or `SIGINT`).

If the Primary Server exits causing the Secondary Server to become active and you then restart the Primary Server, it will notify the Secondary Server to restart and become inactive. You need not terminate the active Secondary Server before restarting the Primary. However, be aware that if the Primary cannot contact the Secondary due to network outage, it will assume the Secondary is *not* running. The Secondary will remain active resulting in two active Servers.

If you need to shutdown and restart the Secondary Server while it is active, then use the `pbs_server` with the `-F` option and a delay value of "-1":

```
pbs_server -F -1
```

The negative one value directs the Secondary Server to become active immediately. It will still make one attempt to connect to the Primary Server in case the Primary is actually up.

### 6.15.6 Failover and Route Queues

When setting up a Server route queue whose destination is in a failover configuration, it is necessary to define a second destination that specifies the same queue on the Secondary Server.

For example, if you already have a routing queue created with a destination as shown:

```
Qmgr: set queue r66 route_destinations=workq@primary.xyz.com
```

you need to add the following additional destination, naming the secondary Server host:

```
Qmgr: set queue r66 route_destinations+=workq@secondary.xyz.com
```

### 6.15.7 Failover and Peer Scheduling

If the Server being configured is also participating in Peer Scheduling, both the Primary and Secondary Servers need to be identified as peers to the Scheduler. For details, see section 8.11.1 “Peer Scheduling and Failover Configuration” on page 147.

# Chapter 7
# Configuring MOM

The execution server daemons, MOMs, require much less configuration than does the Server. The installation process creates a basic MOM configuration file which contains the minimum entries necessary in order to run PBS jobs. This chapter describes the MOM configuration file, and explains all the options available to customize the PBS installation to your site.

## 7.1 MOM Config File

The behavior of each MOM daemon/service is controlled via its configuration file which is read upon initialization (start-up) and upon re-initialization (on UNIX, when `pbs_mom` receives a SIGHUP signal; on Windows, by starting and restarting MOM).

When MOM is started, it will open the `mom_priv/config` file in the path specified in `pbs.conf`, if the `config` file exists. If it does not, MOM will continue anyway. The `config` file may be placed elsewhere or given a different name, by starting `pbs_mom` using the `-c` option with the new file and path specified. (See also section 10.2.1 “Manually Starting MOM” on page 162.)

The configuration file provides several types of run time information to MOM: access control, static resource names and values, external resources provided by a program to be run on request via a shell escape, and values to pass to internal functions at initialization (and re-initialization). Each entry in the configuration file should be on a separate line with the component parts separated by white space. If the line starts with a pound sign ("#"), the line is considered to be a comment and is ignored.

The installation process creates a MOM configuration file with the following entries, which are explained in detail in the subsequent sections of this chapter.

```
$logevent 0x1ff
$clienthost server-hostname
```

## 7.2 MOM Configuration Parameters

Most MOM configuration parameters have a name which starts with a dollar sign ("$"). Directives that are specific to a single operating system, such as SGI IRIX, do not start with a dollar sign "$". Configuration parameters that apply to all systems are first discussed below, followed by system-specific configuration parameters.

### 7.2.1 Standard Configuration Parameters

$action *action-to-modify timeout new-action*

If present, directs MOM to replace the default action for a particular event with a site-specified action. The current list of modifiable actions are:

| | |
|---|---|
| checkpoint | Run the specified program *new-action* in place of the "periodic" job checkpoint action (after which the job continues to run). For details see section 7.6 "Site-Specific Job Checkpoint / Restart" on page 109. |
| checkpoint_abort | Run the specified program *new-action* to checkpoint the job, after which the job is to be terminated. For details see section 7.6 "Site-Specific Job Checkpoint / Restart" on page 109. |
| multinodebusy | When cycle harvesting multi-node jobs, change the default action when a node become busy from "allow job to run" to "requeue job". For details, see section 7.7.3 "Cycle Harvesting and Multi-node Jobs" on page 114. |

| | |
|---|---|
| restart | Run the specified program `new-action` in place of the default job restart action. For details see section 7.6 “Site-Specific Job Checkpoint / Restart” on page 109. |
| terminate | Run the specified program `new-action` in place of the normal SIGTERM / SIGKILL action when MOM is terminating a job. For details, see section 7.5 “Site-specific Job Termination Action” on page 107. |

`$checkpoint_path`

If present, directs MOM to use the pathname following this directive as the location into which to write checkpoint files. (See also section 10.5 “Checkpoint / Restart Under PBS” on page 173.)

```
$checkpoint_path /path/to/ckpdevice
```

`$clienthost` Causes a host name to be added to the list of hosts which will be allowed to connect to MOM. Two host names are always allowed to connect to MOM: “localhost” and the name returned by the system call gethostname(). These names need not be specified in the configuration file. The Server’s host must be either the same as MOM or be listed as a `clienthost` entry in each MOM’s `config` file. Upon startup, MOM will send a restart notice to the Server. The IP addresses of all the hosts (nodes) in the Server `nodes` file will be forwarded by the Server to the MOM on each host listed in the `nodes` file. These hosts need not be in the various MOM’s configuration file as they will be added internally when the list is received from the Server.

The hosts which are provided by the Server to MOM comprise a *sisterhood* of hosts. Any one of the sisterhood will accept connections from a Scheduler [*Resource Monitor* (RM) requests] or Server [jobs to execute] from within the sisterhood. They will also accept *Internal MOM* (IM) messages from within the sisterhood. For a sisterhood to be able to communicate IM messages to each other, they must all share the same RM port. For example, here are two lines for the configuration file which will allow the hosts “phobos” and “deimos” to connect to MOM:

```
$clienthost phobos
$clienthost deimos
```

`$cputmult` Sets a factor used to adjust cpu time used by a job. This is provided to allow adjustment of time charged and limits enforced where the job might run on systems with different cpu performance. If MOM's system is faster than the reference system, set `$cputmult` to a decimal value greater than 1.0. If MOM's system is slower, set `$cputmult` to a value between 1.0 and 0.0. The value is given by

value = speed_of_this_system / speed_of_reference_system

For example:

```
$cputmult 1.5
or
$cputmult 0.75
```

`$enforce` *`arg`* Specifies site-specific resource enforcement behavior of `mem` or `ncpus`. (For details, see section 7.8 "Job Memory Limit Enforcement" on page 116 and section 7.9 "Job NCPUS Limit Enforcement" on page 118.) Arguments to the `enforce` directive take the form:

```
attribute [ value]
```

and include the following:

| Attribute | Type | Description |
|---|---|---|
| `average_cpufactor` | float | ncpus factor to allow; default: 1.025 |
| `average_percent_over` | int | percentage over the limit to allow; default: 50 |
| `average_trialperiod` | int | minimum walltime before enforcement; default: 120 seconds |
| `cpuaverage` | boolean | if present, enforce this limit; default: off |
| `cpuburst` | boolean | if present, enforce this limit; default: off |
| `complexmem` | boolean | if present, enforce this limit; default: off |
| `delta_cpufactor` | float | ncpus factor to allow; default: 1.5 |

| | | |
|---|---|---|
| delta_percent_over | int | percentage over the limit to allow; default: 50 |
| delta_weightup | float | weighting when average is moving up; default: 0.4 |
| delta_weightdown | float | weighting when average is moving down; default: 0.1 |
| mem | boolean | if present, enforce this limit; default: off |

$ideal_load Declares the low water mark for load on a node. It works in conjunction with a $max_load directive. When the load average on the node drops below the ideal_load, MOM on that node will inform the Server that the node is no longer busy.

$kbd_idle Enables support for “idle workstation cycle harvesting” as described later in this chapter. Details are provided on page 111.

```
$kbd_idle 1800 10 5
```

$logevent Sets the mask that determines which event types are logged by pbs_mom. For example:

```
$logevent 0x1ff

$logevent 255
```

The first example would set the log event mask to 0x1ff (511) which enables logging of all events including debug events. The second example would set the mask to 0x0ff (255) which enables all events except debug events. The values of events are listed in section 10.12 “Use and Maintenance of Logfiles” on page 183.

$max_load Declares the high water mark for load on a node. It is used in conjunction with a $ideal_load directive. When the load average exceeds the high water mark, MOM on that node will notify the Server that the node is busy. The state of the node will be shown as busy. A busy cluster node will not be allocated to jobs. This is useful in preventing allocation of jobs to nodes which are busy with interactive sessions.

A busy time-shared node may still run new jobs under the direction of the Scheduler. Both the $ideal_load and $max_load directives add a static resource, ideal_load and max_load, which may be queried by the Scheduler. These static resources are supported by the Standard scheduler when load-balancing jobs. Note that the Scheduler will attempt to predict the load on the node prior to running a job, by taking the current load, and adding the number of CPUs requested by a job being considered. If the result is greater than max_load, then the job will not be run. For example:

```
$ideal_load 2.0
$max_load 3.5
```

$prologalarm

Sets the time-out period in seconds for the prologue and epilogue scripts. (See discussion of the prologue and epilogue in section 10.11 “Job Prologue / Epilogue Programs” on page 180.) An alarm is set to prevent the script from locking up the job if the script hangs or takes a very long time to execute. The default value is 30 seconds. An example:

```
$prologalarm 60
```

$restricted

Causes a host name to be added to the list of hosts which will be allowed to connect to MOM without needing to use a privileged port. The name specification allows for wildcard matching. Connections from the specified hosts are restricted in that only internal queries may be made. Only static resources from the config file will be reported and no control requests can be issued. This is to prevent any shell commands from being run by a non-root process.

This type of entry is typically used to specify hosts on which a monitoring tool, such as xpbsmon, can be run. xpbsmon will query MOM for general resource information.

For example, here is a configuration file line which will allow queries from any host from the domain “pbspro.com”:

```
$restricted *.pbspro.com
```

$suspendsig Specify an alternative signal to be used for suspending jobs.

```
$suspendsig signal_number
```

$usecp Directs MOM to use /bin/cp instead of rcp or scp for delivery of output files. If MOM is to move a file to a host other than her own, MOM normally uses a remote copy command (scp or rcp) to transfer the file. This applies to stage-in/out and delivery of the job's standard output/error. The destination is recorded as hostx:/full/path/name. So if hostx is not the same system on which MOM is running, she uses scp or rcp; if it is the same system, MOM uses /bin/cp. (To enable scp, set the PBS_SCP parameter as described in the discussion of "pbs.conf" on page 159.)

However, if the destination file system is NFS mounted among all the systems in the PBS environment (cluster), cp may work better than scp/rcp. One or more $usecp directives in the config file can be used to inform MOM about those file systems where the cp command should be used instead of scp/rcp. The $usecp entry has the form:

$usecp *hostspec*:*path_prefix new_prefix*

The *hostspec* is either a fully qualified host–domain name or a wild-carded host–domain specification as used in the Server's host ACL parameter. The *path_prefix* is the leading (root) component of the fully qualified path for the NFS files as visible on the specified host. The *new_prefix* is the initial components of the path to the same files on MOM's host. If different mount points are used, the path_prefix and the new_prefix will be different. If the same mount points are used for the cross mounted file system, then the two prefixes will be the same. When given a file destination, MOM will:

Step 1 Match the hostspec against her host name. If they match, MOM will use the cp command to move the file. If the hostspec is "localhost" then MOM will also use cp.

Step 2 If the match in step one fails, MOM will match the host portion of the destination against each $usecp hostspec in turn. If the host matches, MOM matches the path_prefix against the initial segment of the destination name. If this matches, MOM will discard the host name, replace the initial segment of the path that matched against `path_prefix` with the `new_prefix` and use `cp` with the resulting destination.

Step 3 If the host is neither the local host nor matches any of the `$usecp` directives, MOM will use the `scp` or `rcp` command to move the file. For example, a user named Beth on host phobos.pbspro.com submits a job while her current working directory is:

```
   /u/wk/beth/proj
```

The destination for her output would be given by PBS as:

`phobos.pbspro.com:/u/wk/beth/proj/123.OU`.

The job runs on node jupiter.pbspro.com which has the user's home file system cross mounted as `/r/home/beth`. Either of the following entries in the config file on node jupiter will result in a `cp` copy to `/r/home/beth/proj/123.OU` instead of an `rcp` copy to `phobos.pbspro.com:/u/wk/beth/proj/123.OU`

```
$usecp phobos.pbspro.com:/u/wk/ /r/home/

$usecp *.pbspro.com:/u/wk/ /r/home/
```

Note that the destination is matched against the `$usecp` entries in the order listed in the config file. The first match of host and file prefix determines the substitution. Therefore, if you have the same physical file system mounted as `/scratch` on node mars and as `/workspace` on every other host, then the entries in the config file on jupiter should be in the following order:

```
$usecp mars.pbspro.com:/scratch /workspace
$usecp *.pbspro.com:/workspace /workspace
```

`$wallmult` Sets a factor used to adjust wall time usage by a job to a common reference system. The factor limits in the same way as `$cputmult` is used for cpu time.

`$tmpdir` Specifies the top level location under which each job will have its own directory set up, where the PATH argument names a secure (not world writable) directory, such as /usr/tmp. The job-specific sub-directory will be named using the jobid of the job. In addition, the job’s TMPDIR environment variable will be set to the full path of this job-specific sub-directory. When the job terminates, the subdirectory and its contents are deleted.

```
$tmpdir PATH
```

### 7.2.2 SGI-Specific Configuration Parameters

Directives that are specific to a single operating system do not start with a dollar sign “$”. This section discusses parameters that are specific to SGI systems. See also section 7.10 “Enhanced SGI “cpusets” Support” on page 119.

`alloc_nodes_greedy`

On "cpuset"-enabled SGI IRIX systems, requests to allocate nodes close together (0) or anywhere (1, the default).

```
alloc_nodes_greedy 1
```

`checkpoint_upgrade`

If present, causes PBS to pass a special upgrade checkpoint flag to the SGI IRIX checkpoint system for use immediately prior to an IRIX operating system upgrade. For details on use, see section 10.5.4 “Checkpointing Jobs Prior to SGI IRIX Upgrade” on page 174.

`cpuset_create_flags` *`arg`*

On "cpuset"-enabled SGI systems, defines whether to allocate either memory or CPUs in an exclusive fashion.

```
cpuset_create_flags CPUSET_CPU_EXCLUSIVE
cpuset_create_flags CPUSET_MEMORY_EXCLUSIVE
cpuset_create_flags CPUSET_POLICY_PAGE
```

Optionally, the list of flags can be “OR’d” together on a single line, as shown in the example below (but there can be no whitespace between the OR bars, i.e. “|”s).

```
cpuset_create_flags CPUSET_CPU_EXCLUSIVE|CPUSET_MEMORY_EXCLU
```

`cpuset_destroy_delay` *`seconds`*

On "cpuset"-enabled SGI systems, specifies the number of seconds to wait before tearing down (deleting) a "cpuset".

`cpuset_small_mem` *`arg`*

On "cpuset"-enabled SGI systems, disables running jobs on shared "cpusets” when set to “0”. Otherwise, this sets the maximum amount of memory (in kilobytes) that a small job, which shares a cpuset, can request.

`cpuset_small_ncpus` *`arg`*

On “cpuset”-enabled SGI systems, disables running jobs on shared “cpusets” when set to “0”. Otherwise, this sets the maximum number of CPUs that a small job, which shares a cpuset, can request.

`max_shared_nodes`

Specifies the number of nodes that should be allocated to shared jobs with cpuset-enabled SGI IRIX systems.

## 7.3 Static Resources

To identify static resource names and values, the MOM configuration file can contain a list of resource name/value pairs, one pair per line, separated by white space. These are most often used by the alternate schedulers, but are listed here for completeness. The names can be anything and are not restricted to actual hardware. For example the entry `“pongsoft 1”` could be used to indicate to the Scheduler that a certain piece of software (“pong”) is available on this system. Another example could be the number of tape drives of different types.

```
pongsoft 1
tapedat  1
tape8mm  1
```

For more information on creating site-specific resources, see Chapter 9, “Customizing PBS Resources” on page 149.

## 7.4 Dynamic Resources

PBS provides the ability to extend the resource query capabilities of MOM by adding shell escapes to the MOM configuration file. The primary use of this feature is to add site-specific resources, such as to manage software application licenses. For a full discussion of this capability, and examples of use, see Chapter 9, “Customizing PBS Resources” on page 149.

## 7.5 Site-specific Job Termination Action

The default behavior of PBS is for MOM to terminate a job when the job's usage of a resource exceeds the limit requested or when the job is deleted by the Server on shutdown or because of a `qdel` command. However, a site may specify a script (or program) to be run by `pbs_mom` in place of the normal `SIGTERM`/`SIGKILL` action when MOM is terminating a job under the above conditions. This action takes place on terminate from exceeding resource limits or from usage of the `qdel` command. The script is defined by adding the following directive to MOM's config file:

```
$action terminate TIME_OUT !SCRIPT_PATH [ARGS]
```

Where *`TIME_OUT`* is the time, in seconds, allowed for the script to complete.

*`SCRIPT_PATH`* is the path to the script. If it is a relative path, it is evaluated relative to the `PBS_HOME/mom_priv` directory.

**Important:** Under windows, *`SCRIPT_PATH`* must have a “`.bat`” suffix since it will be executed under the Windows command prompt `cmd.exe`. The script must only be writable by an Administrator-type account. If the script is in *`PBS_HOME`*`\mom_priv\`, then the Administrator can run the following command to correct the permissions and ownership of the file.

```
pbs_mkdirs mom
```

*`ARGS`* are optional arguments to the script. Values for *`ARGS`* may be: `STRING` (any string not starting with '%') or `%keyword` (which is replaced by MOM with the corresponding value).

| | |
|---|---|
| `%jobid` | job id |
| `%sid` | session id of task (job) |
| `%uid` | execution uid of job |
| `%gid` | execution gid of job |
| `%login` | login name associated with uid |
| `%owner` | job owner "name@host" |
| `%auxid` | aux id (system dependent content) |

If the script exits with a zero exit status (before the time-out period), PBS will not send any signals or attempt to terminate the job. It is the responsibility of the termination script in this situation to insure that the job has been terminated. If the script exits with a non-zero exit status, the job will be sent `SIGKILL` by PBS. If the script does not complete in the time-out period, it is aborted and the job is sent `SIGKILL`. A *`TIME_OUT`* value of 0 is an infinite time-out.

A UNIX example:

```
$action terminate  60   !endjob.sh %sid %uid %jobid
or
$action terminate   0   !/bin/kill -13 %sid
```

A similar Windows example:

```
$action terminate 60 !endjob.bat %sid %uid %jobid
or
$action terminate 0 !”C:/Program Files/PBS Pro/exec/bin/pbskill” %sid
```

The first line in both examples above sets a 60 second time-out value, and specifies that `PBS_HOME/mom_priv/endjob.sh` (`endjob.bat` under Windows) should be executed with the arguments of the job’s session ID, user ID, and PBS jobs ID. The third line in the first (UNIX) example simply calls the system `kill` command with a specific signal (13) and the session ID of the job. The third line of the Windows example calls the PBS-provided `pbskill` command to terminate a specific job, as specified by the session id (`%sid`) indicated.

## 7.6 Site-Specific Job Checkpoint / Restart

The PBS Pro site-specific job checkpoint facility allows an Administrator to replace the built-in checkpoint facilities of PBS Pro with a site-defined external command. This is most useful on computer systems that do not have OS-level checkpointing. This feature is implemented by setting five MOM configuration parameters. The first two parameters control the behavior of MOM in regards to checkpointing and the remaining three parameters pertain to restart activities.

```
$action checkpoint        TIME_OUT !SCRIPT_PATH ARGS [...]
$action checkpoint_abort  TIME_OUT !SCRIPT_PATH ARGS [...]
$action restart           TIME_OUT !SCRIPT_PATH [ARGS ...]
```

*TIME_OUT* is the time (in seconds) allowed for the script (or program) to complete. If the script does not complete in this period, it is aborted and handled in the same way as if it returned a failure. This does not apply if "restart_transmogrify" is "true" (see below), in which case, no time check is preformed.

*SCRIPT_PATH* is the path to the script. If it is a relative path, it is evaluated relative to the `PBS_HOME/mom_priv` directory.

*ARGS* are the arguments to pass to the script. The following *ARGS* are expanded by PBS:

| | |
|---|---|
| `%globid` | Global ID |
| `%jobid` | Job ID |
| `%sid` | Session ID |
| `%taskid` | Task ID |
| `%path` | File or directory name to contain checkpoint files |

```
$restart_background   (true|false)
$restart_transmogrify (true|false)
```

The MOM configuration parameter `restart_background` is a boolean flag that modifies how MOM performs a restart. When the flag is "false" (the default), MOM runs the restart operation and waits for the result. When the flag is "true", restart operations are done by a child of MOM which only returns when all the restarts for all the local tasks of a job are done. The parent (main) MOM can then continue processing without being blocked by the restart.

The MOM configuration parameter `restart_transmogrify` is a boolean flag that controls how MOM launches the restart script/program. When the flag is “false” (the default) MOM will run the restart script and block until the restart operation is complete (and return success or appropriate failure). In this case the restart action must restore the original session ID for all the processes of each task or MOM will no longer be able to track the job. Furthermore, if `restart_transmogrify` is "false" and restart is being done with an external command, the configuration parameter `restart_background` will be ignored and the restart will be done as if the setting of `restart_background` was "true". This is to prevent a script that hangs from causing MOM to block. If `restart_transmogrify` is "true", MOM will run the restart script/program in such a way that the script will "become" the task it is restarting. In this case the restart action script will replace the original task's top process. MOM will replace the session ID for the task with the session ID from this new process. If a task is checkpointed, restarted and checkpointed again when `restart_transmogrify` is "true", the session ID passed to the second checkpoint action will be from the new session ID.

### 7.6.1 Guidelines for Creating Local Checkpoint Action

This section provides a set of guidelines the Administrator should follow when creating a site-specific job checkpoint / restart program (or script). PBS will initiate the checkpoint program/script for each running task of a job. This includes all the nodes where the job is running. The following environment variables will be set:

| | | | |
|---|---|---|---|
| GID | HOME | LOGNAME | PBS_GLOBID |
| PBS_JOBCOOKIE | PBS_JOBID | PBS_JOBNAME | PBS_MOMPORT |
| PBS_NODEFILE | PBS_NODENUM | PBS_QUEUE | PBS_SID |
| PBS_TASKNUM | SHELL | UID | USER |

The checkpoint command should expect and handle the following inputs:

Global ID
Job ID
Session ID
Task ID
Filename or Directory name to contain checkpoint files

The restart command should return success or failure error codes, and expect and handle as input a file/directory name.

Both the checkpoint and restart scripts/programs should block until the checkpoint/restart operation is complete. When the script completes, it should indicate success or failure by returning an appropriate exit code and message. To PBS, an exit value of 0 indicates success, and a non-zero return indicates failure.

Note that when the MOM configuration parameter `restart_transmogrify` is set to "false" the restart action must restore the original session ID for all the processes of each task or MOM will no longer be able to track the job. If the parameter `restart_transmogrify` is set to "true", when the restart script for a task exits, the task will be considered done, and the restart action *`TIME_OUT`* will not be used.

## 7.7 Idle Workstation Cycle Harvesting

“Harvesting” of idle workstations is a method of expanding the available computing resources of your site by automatically including in your cluster unused workstations that otherwise would have sat idle. This is particularly useful for sites that have a significant number of workstation that sit on researchers’ desks and are unused during the nights and weekends. With this feature, when the “owner” of the workstation isn’t using it, the machine can be configured to be used to run PBS jobs.

### 7.7.1 Supported Platforms for Cycle Harvesting

Due to different operating system support for tracking mouse and keyboard activity, the availability and method of support for cycle harvesting varies based on the computer platform in question. The following table illustrates the method and support per system.

| System | Status | Method | Reference |
|---|---|---|---|
| AIX | supported | pbs_idled | See section 7.7.2. |
| FreeBSD | unsupported | pbs_idled | See section 7.7.2. |
| HP/ux 10 | supported | device | See section 7.7.4. |
| HP/ux 11 | supported | device | See section 7.7.4. |
| IRIX | supported | pbs_idled | See section 7.7.2. |
| Linux | supported | device | See section 7.7.4. |

| System | Status | Method | Reference |
|---|---|---|---|
| MacOS-X | unsupported | pbs_idled | See section 7.7.2. |
| Solaris | supported | device | See section 7.7.4. |
| Tru64 | supported | pbs_idled | See section 7.7.2. |
| Windows XP Pro | supported | other | See section 7.7.2. |
| Windows<br>2003 Advanced Server | supported | other | See section 7.7.2. |
| Windows<br>2000 Pro | supported | other | See section 7.7.2. |
| Windows<br>2000 Server | supported | other | See section 7.7.2. |

### 7.7.2 Configuring Cycle Harvesting

If a system is configured for cycle harvesting, it becomes available for batch usage by PBS if its keyboard and mouse remain unused or idle for a certain period of time. The workstation will be shown in state "free" when the status of the node is queried. If the keyboard or mouse is used, the workstation becomes unavailable for batch work and PBS will suspend any running jobs on that workstation and not attempt to schedule any additional work on that workstation. The workstation will be shown in state "busy", and any suspended jobs will be shown in state "U", a new state.

**Important:** Jobs on workstations that become *busy* will not be migrated; they remain on the workstation until they complete execution, are rerun, or are deleted.

The cycle harvesting feature is enabled via a single entry in `pbs_mom`'s `config` file, `$kbd_idle`, and takes up to three parameters, as shown below.

```
$kbd_idle idle_available [ idle_busy [ idle_poll ] ]
```

These three parameters, representing time specified in seconds, control the transitions between *free* and *busy* states. Definitions follow.

`idle_available` is the time that the workstation keyboard and mouse must be idle before the workstation is available to PBS.

`idle_busy` is a time period during which the keyboard or mouse must remain busy before the workstation "stays" unavailable. This is used to keep a single key stroke or mouse movement from keeping the workstation busy.

`idle_poll` is how often the state of the keyboard and mouse are checked.

Let us consider the following example.

```
$kbd_idle 1800 10 5
```

Adding the above line to MOM's config file directs PBS to mark the workstation as *free* if the keyboard and mouse are idle for 30 minutes (1800 seconds), to mark the workstation as *busy* if the keyboard or mouse are used for 10 consecutive seconds, and the state of the keyboard/mouse is to be checked every 5 seconds.

The default value of *idle_busy* is 10 seconds, the default for *idle_poll* is 1 second. There is no default for *idle_available*; setting it to non-zero is required to activate the cycle harvesting feature.

Elaborating on the above example will help clarify the roll of the various times. Lets start with a workstation that has been in use for some time by its owner. The workstation is shown in state *busy*. Now the owner goes to lunch. After 1800 seconds (30 minutes), the system will change state to *free* and PBS may start assigning jobs to run on the system. At some point after the workstation has become *free* and a job is started on it, someone walks by and moves the mouse or enters a command. Within the next 5 seconds, `pbs_mom` notes the activity. The job is suspended and shown being in state "U" and the workstation is marked *busy*. If, after 10 seconds have passed and there is no additional keyboard/mouse activity, the job is resumed and the workstation again is shown as *free*. However, if keyboard/mouse activity continued during that 10 seconds, then the workstation would remain *busy* and the job would remain suspended for at least the next 1800 seconds.

A site may wish to enable cycle harvesting, but only for single-node jobs. If this is the case, the `no_multinode_jobs` parameter can be set. For details, see the entry for `no_multinode_jobs` entry on page 72.

**Important:** Some computer operating systems do not update the keyboard and mouse access times, causing PBS not to be able to query keyboard/mouse activity. For details, and for alternative methods of using cycle harvesting, see section 7.7.1 “Supported Platforms for Cycle Harvesting” on page 111.

### 7.7.3 Cycle Harvesting and Multi-node Jobs

When a job which only uses a single node is running on a workstation configured for cycle harvesting, and that node becomes "busy", the job is suspended. However, suspending a multi-node parallel job may have undesirable side effects because of the inter-process communications. Thus the default action for job which uses multiple nodes when one or more of the nodes become busy is to leave the job running.

It is possible, however, to specify that the job should be requeued (and subsequently rescheduled to run elsewhere) when any of the nodes on which the job is running become *busy*. To enable this action, the Administrator must add the following directive to MOM’s configuration file:

```
$action multinodebusy 0 requeue
```

where "`multinodebusy`" is the action to modify; "`0`" (zero) is the action time out value (it is ignored for this action); and "`requeue`" is the new action to perform.

**Important:** Jobs which are not rerunnable (i.e. those submitted with the `qsub -rn` option) will be killed if the requeue action is configured and a node becomes busy.

### 7.7.4 Cycle Harvesting on Machines with X-Windows

If you wish to enable cycle harvesting on machines which run X-Windows, special steps need to be taken. On Solaris and Linux, the console, keyboard, and mouse device access times are updated periodically. The PBS MOM process takes note of that and marks the node busy if any of the input devices are in use. On others Unixes this data is not available. In such cases, PBS must monitor the X-Window System in order to obtain interactive idle time. To support this, there is a PBS X-Windows monitoring daemon called `pbs_idled`. This program runs in the background and monitors X and reports to the `pbs_mom` whether the node is idle or not. Because of X-Windows security, running `pbs_idled` requires more modification then just installing PBS. First, a directory must be made for `pbs_idled`. This directory must have the same permissions as `/tmp` (i.e. mode 1777). This will allow the `pbs_idled` to create and update files as the user, which is necessary because the program will be running as the user. For example:

```
on Linux:
# mkdir /var/spool/PBS/spool/idledir
# chmod 1777 /var/spool/PBS/spool/idledir

on UNIX:
# mkdir /usr/spool/PBS/spool/idledir
# chmod 1777 /usr/spool/PBS/spool/idledir
```

Next, turn on keyboard idle detection in the MOM `config` file:

```
$kbd_idle 300
```

Lastly, `pbs_idled` needs to be started as part of the X-Windows startup sequence. The best and most secure method of installing `pbs_idled` is to insert it into the system wide `Xsession` file. This is the script which is run by `xdm` (the X login program) and sets up each user's X-Windows environment. The startup line for `pbs_idled` must be before the that of the window manager. It is also very important that `pbs_idled` is run in the background. On systems that use `Xsession` to start desktop sessions, a line invoking `pbs_idled` should be inserted near the top of the file. `pbs_idled` is located in `$PBS_EXEC/sbin`. For example, the following line should be inserted in a Linux `Xsession` file:

```
/usr/pbs/sbin/pbs_idled &
```

**Important:** On a Tru64 system running CDE, inserting `pbs_idled` into an Xsession file will *not* result in the executable starting. Rather, it needs to be added to the `dtsession_res` file. which typically has the following path:

```
/usr/dt/bin/dtsession_res
```

Note that if access to the system-wide Xsession file is not available, pbs_idled may be added to every user's personal .xsession, .xinitrc, or .sgisession file (depending on the local OS requirements for starting X-windows programs upon login).

**Important:** OS-X does not run X-Windows as its primary windowing system, and therefore does not support cycle harvesting.

### 7.7.5 Cycle Harvesting and File Transfers

The cycle harvesting feature interacts with file transfers in one of two different ways, depending on the method of file transfer. If the user’s job includes file transfer commands (such as rcp or scp), and such a command is running when PBS decides to suspend the job on the node, then the file transfer will be suspended as well.

However, if the job has PBS file staging directives (i.e. stageout=file1...), the file transfer will not be suspended. This is because the file staging occurs as part of the post-execution (or “E”xiting state, after the epilogue is run), and is not subject to suspension. (For more information on PBS file staging, see the **PBS Pro User Guide**.)

## 7.8 Job Memory Limit Enforcement

Enforcement of the "mem" (physical memory) resource usage is available on all platforms. Enforcement is configurable by the $enforce entry in MOM’s config file (see “$enforce arg” on page 100). By default, enforcement is off. If a $enforce mem statement appears in the config file, then jobs that exceed their specified amount of physical memory will be killed. There are two items to be aware of:

1. "mem" is a per job/ per node limit.
2. "mem" enforcement is polled, therefore a job may exceed its limit for up to two minutes before it is detected.

### 7.8.1 SUN Solaris-specific Memory Enforcement

If a job is submitted with a "pmem" limit or without "pmem" and with a "mem" limit, PBS uses the setrlimit(2) call to set the limit. For most Operating Systems, setrlimit() is called with RLIMIT_RSS which limits the Resident Set (working set size). This is not a hard limit, but advice to the kernel. This process becomes a prime candidate to have memory pages reclaimed. Solaris does not support RLIMIT_RSS, but instead has RLIMIT_DATA and RLIMIT_STACK, which are hard limits. Therefore, under Solaris, a malloc() call that exceeds the limit will return NULL. This behavior is different than under other operating systems and may result in the program (such as a user's application) receiving a SIGSEGV signal.

### 7.8.2 SGI Non-"cpuset" Memory Enforcement

Under IRIX 6.5.x, there are two ways to determine the amount of real memory a set of processes are using. The "simple" way, as used by the ps(1) command, looks solely at the pr_rssize field of the /proc/pinfo/ entry for each process. The "complex" method uses special SGI calls to determine the "shared" state of each memory segment in each process.

The "simple" method is quick and clean. However, this method does not factor in shared memory segments, so the resulting usage figure for processes that are started by the sproc(2) call is too high. The shared segments are counted fully against each process. This "apparent" over usage can result in under loading of the physical memory in the system.

The "complex" method correctly factors in the shared memory segments and yields a more accurate report on the amount of physical memory used. However, the SGI ioctl(PIOCMAP_SGI) call requires that the kernel look at each memory segment. This can result in the calling program, pbs_mom, being blocked for an extended period of time on larger systems. Systems smaller than 32 CPUs are not likely to see a problem.

By default, the "simple" option is enabled. With the addition of a $enforce complexmem statement in MOM's config file, the "complex" memory usage calculation is selected.

**Important:** If the "complex" method is selected, the Administrator needs to monitor the MOM logs for a warning of the form "time lag N secs" where N is a number of seconds greater than five. If this message appear frequently, it means the IRIX kernel is taking that long to respond to the ioctl call and the performance of pbs_mom may suffer. In that case, it is recommended that the site revert to the "simple" calculation or run the "cpuset" version of MOM.

## 7.9 Job NCPUS Limit Enforcement

Enforcement of the ncpus (number of CPUs used) is now available on all platforms. Enforcement is configurable by a set of entries in MOM's config file. (See description of $enforce on page 100.) By default, enforcement is off since it has not been enforced in the past.

Associated with this enforcement is the read-only resource cpupercent. This is a report of the average percentage usage of one CPU. For example, a value of 50 means that during a certain period, the job used 50 percent of one CPU. A value of 300 means over the period, the job used an average of three CPUs. Enforcement is based on two separate checks. The first check is based on the polled sum of CPU time for all processes in the job. Each poll period, the total CPU time used, divided by the total walltime, is checked against the value specified for ncpus * 100. If this "average" value exceeds the number of CPUs requested and if the cpuaverage enforcement is turned on, the job is aborted. The second check during each poll period looks for sudden bursts of CPU activity. cpupercent is a moving weighted average based on the prior cpupercent and the amount of new CPU time used divided by the walltime for this period. This value can be weighted to ignore or punish sudden bursts of CPU activity. This enforcement is available if cpuburst is set in MOM's config file.

The following parameters, set via a $enforce statement in the config file, control the enforcement, weighting, and allowed overrun. (Types and description of each are given in the discussion of "$enforce arg" on page 100 above.)

For the absolute cpu time / walltime calculation, the following enforce arguments are used: cpuaverage, average_trialperiod, average_percent_over, average_cpufactor. Given the default values, a job will be killed if:

(cput / walltime) > (ncpus * average_cpufactor + average_percent_over / 100)

This enforcement only occurs after the job has had average_trialperiod seconds of walltime. For the weighted moving average, the following enforce arguments are used: cpuburst, delta_percent_over, delta_cpufactor, delta_weightup, delta_weightdown. The read-only cpupercent value is determined by weighing the new cput/walltime for the last measurement period and the old cpupercent:

new_percent = change_in_cpu_time / change_in_walltime
weight = delta_weight[up|down] * walltime/max_poll_period
new_cpupercent = (new_percent * weight) + (old_cpupercent * (1-weight))

`delta_weight_up` is used if the `new_percent` is higher than the old `cpupercent` value and `delta_weight_down` is used if `new_percent` is lower than the old value. If `delta_weight_`[`up`|`down`] is 0.0, then the value for `cpupercent` will not change with time. If it is 1.0, `cpupercent` will become the `new_percent` for the poll period; that means `cpupercent` changes quickly. (`max_poll_period` is the maximum time between samples, set to 120 seconds.) The job will be killed if

new_cpupercent > ((ncpus * 100 * delta_cpufactor) + delta_percent_over)

The following entries in MOM's `config` file would turn on enforcement of both average and burst with the default values:

```
$enforce cpuaverage
$enforce cpuburst
$enforce delta_percent_over 50
$enforce delta_cpufactor 1.05
$enforce delta_weightup 0.4
$enforce delta_weightdown 0.1
$enforce average_percent_over 50
$enforce average_cpufactor 1.025
$enforce average_trialperiod 120
```

## 7.10 Enhanced SGI "cpusets" Support

PBS Pro includes enhancements for running PBS in conjunction with SGI "cpusets". (An SGI "cpuset" is a named region of an SGI Origin or Altix system composed of one or more nodeboards, and associated memory and CPUs.) This section discusses these features in conjunction with previous versions' support for "cpusets".

**Important:** SGI "cpuset" support on Altix systems requires the SGI ProPack library. For details, see section 3.3.2 on page 16.

**Important:** Because "cpusets" cannot span systems, they can not be used by multi-node jobs. Jobs utilizating "cpusets" must therefore request CPUs and not nodes (i.e. via the `qsub` option "`-l ncpus=N`" rather than "`-l nodes=N`" as discussed in the **PBS Pro User Guide**.

### 7.10.1 SGI IRIX “cpuset” Caveats

Users of irix6cpuset MOMs running under OS version 6.5.21 need to override the “cpuset” creation flags default by defining explicitly a flags set that doesn't contain CPUSET_EVENT_NOTIFY. This flag confuses the “cpuset” creation process. For example, the following can be added to mom's config file (all on one line, without the “\”s):

```
cpuset_create_flags CPUSET_CPU_EXCLUSIVE|\
CPUSET_MEMORY_LOCAL|\
CPUSET_MEMORY_EXCLUSIVE|\
CPUSET_MEMORY_MANDATORY|\
CPUSET_POLICY_KILL
```

### 7.10.2 SGI Altix “cpuset” Caveats

The SGI Altix machine is supported with (or without) cpusets. The interface is similar to what is used for IRIX cpuset machines. The known differences follow:

The `altid` returned by MOM has the form "cpuset=name" where the cpuset name follows the equal sign. There is no walltime associated with a shared cpuset. Once the last job using a shared cpuset exits (or is suspended), the shared cpuset is cleared.

There are fewer flags that can be specified in the "cpuset_create_flags" MOM configuration parameter. The available flags are:

```
CPUSET_CPU_EXCLUSIVE
CPUSET_MEMORY_LOCAL
CPUSET_MEMORY_EXCLUSIVE
CPUSET_MEMORY_KERNEL_AVOID
CPUSET_MEMORY_MANDATORY
CPUSET_POLICY_PAGE
CPUSET_POLICY_KILL
CPUSET_EVENT_NOTIFY
```

The missing flag is:

```
CPUSET_KERN
```

Several IRIX “cpuset”-specific MOM configuration parameters are not available. The parameters that are available are:

```
cpuset_create_flags
cpuset_destroy_delay
cpuset_small_ncpus
cpuset_small_mem
max_shared_nodes
memreserved
```

The "small cpuset" functionality is on by default and requires a job have both ncpus and mem set so the MOM can decide if it will fit into a small cpuset.

### 7.10.3 IRIX OS-Level Checkpoint With “cpusets”

MOM supports use of new IRIX checkpointing features to allow the checkpointing and restart of jobs running within SGI “cpusets”. This capability requires SGI IRIX version 6.5.16 or later.

### 7.10.4 SGI Shared “cpusets” for Small Jobs

MOM supports two classes of jobs running within “cpusets”: a *small* job and a *multi-cpu* job. The small job, which is usually a single CPU job with memory requirements such that it will fit on a single nodeboard, will allow itself to be run within a “cpuset” where other similar jobs run. Multi-cpu jobs are those that require more resources than are available on a single nodeboard or which need to run exclusively within a "cpuset" for repeatability of performance. The small job runs on a shared "cpuset" whereas the multi-cpu job runs on an exclusive "cpuset". Also, up to `max_shared_nodes` (set in MOM’s `config` file) will be allowed to be assigned to shared “cpusets”. To find out which "cpuset" is assigned to a running job, the `alt_id` job attribute has a field called `cpuset` that will show this information.

**Important:** The SGI Altix system has shared "cpuset" support enabled by default. Therefore, if the Server does not have the two parameters
`resources_default.mem`, `resources_default.ncpus`
set, and if the job hasn't explicitly requested both of these values, the MOM will not be able to determine if the job will be a small "cpuset" candidate. MOM will therefore not run the job and, after several iterations on this interaction between Scheduler and MOM, the Scheduler will put a system hold on the job.

To set a threshold as to the number of nodeboards that can be assigned to shared "cpusets”, set the following in MOM’s `config` file (see also section 7.2.2 “SGI-Specific Configuration Parameters” on page 105):

```
max_shared_nodes <# of nodeboards>
```

To define the maximum number of CPUs and amount of memory (in kb) that a job can request and still be considered a small job, specify those amounts using the following parameters in MOM’s config file:

```
cpuset_small_ncpus 2
cpuset_small_mem 1024
```

If you do not want MOM to run small jobs within a shared "cpuset", set the following in MOM’s config file:

```
cpuset_small_ncpus 0
cpuset_small_mem 0
```

### 7.10.5 MOM “cpuset” Configuration Options

In order to make MOM `config` file options consistent, the format of `cpuset_create_flags` for specifying the flags to pass to `cpusetCreate()` is now changed from

```
    $cpuset_create_flags <flags>
```

to

```
    cpuset_create_flags <flags>
```

Similarly, the format for `cpuset_destroy_delay` for setting the number of seconds to sleep before tearing down (deleting) a "cpuset" is changed from

```
    $cpuset_destroy_delay <secs>
```

to

```
    cpuset_destroy_delay <secs>
```

In addition, a new option is available:

```
    alloc_nodes_greedy <0 or 1>
```

If set to 0, for job requests of 64 or fewer ssinodes, MOM will only allocate eligible nodes that are within one router hop in the hypercube architecture, that is, those nodes that are physically sitting close to each other. Otherwise if set to 1, MOM will allocate nodes from all those available regardless of the distance (number of router hops) between them. This latter case is the default behavior.

**Important:** Upon daemon startup, MOM will not remove any "cpusets" that it did not create. The nodes in these "cpusets" will be removed from MOM's available nodepool.

### 7.10.6 Enhanced "cpuset" Resource Reporting

MOM will report to the Server the actual number of CPUs and memory that are under the control of PBS. This allows the node's resources_available.{ncpus,mem} to reflect the amount of resources that come from nodeboards that are not part of the reserved, system "cpusets" (e.g. boot) and stuck "cpusets". Be sure to unset any manual settings of resources_available.{ncpus,mem} in both the node and the Server to get this count automatically updated by MOM. Manual settings (i.e. those either put in the server's nodes file or via the qmgr set node construct) take precedence. If manually setting the server's resources_available.ncpus parameter, be sure to use a value that is a multiple of the nodeboard size. This value should not be less than one nodeboard size, otherwise no jobs (including shareable jobs) will run. For example, if there are four cpus per nodeboard, don't set resources_available.ncpus=3, instead set resources_available.ncpus=4 (8, 12, 16, and so on).

### 7.10.7 Node Allocation Rules with "cpusets"

Some special node and CPU allocation rules are now enforced by MOM on "cpuset" enabled systems:

If cpuset_create_flags set during cpusetCreate() contains a flag for CPUSET_CPU_EXCLUSIVE or CPUSET_MEMORY_EXCLUSIVE, then CPU 0 will not be allowed to be part of a "cpuset".

During allocation of nodes, nodeboard 0 will only be allocated if no other nodeboards are available to satisfy the request. Use of nodeboard 0 for jobs can be a source of performance degradation as the kernel heavily uses this node for system daemons. Usually, PBS with "cpusets" is used in conjunction with a boot "cpuset" which the system administrator creates which includes nodeboard 0.

## 7.11 MOM Globus Configuration

For the optional Globus MOM, the same configuration mechanism applies as with the regular MOM except only three initiation value directives are applicable: `$clienthost`, `$restricted`, `$logevent`. For details, see the description of these configuration directive earlier in this chapter.

## 7.12 Examples

Examples of different MOM configurations are included in Chapter 12 “Example Configurations” on page 221.

# Chapter 8
# Configuring the Scheduler

Now that the Server and MOMs have been configured, we turn our attention to the Scheduler. As mentioned previously, the Scheduler is responsible for implementing the local site policy by which jobs are run, and on what resources. This chapter discusses the default configuration created in the installation process, and describes the full list of tunable parameters available for the PBS Standard Scheduler.

## 8.1 Default Configuration

This Standard Scheduler provides a wide range of scheduling policies. It provides the ability to sort the jobs in several different ways, in addition to FIFO order, such as on user and group priority, fairshare, and preemption. As distributed, it is configured with the following options (which are described in detail below).

1. Three specific system resources are checked to make sure they are available: `mem` (memory requested), `ncpus` (number of CPUs requested) and `arch` (architecture requested).

2. Queues are sorted into descending order by queue priority attribute to determine the order in which they are to be considered.

3. All jobs in the highest priority queue will be considered for execu-

tion before jobs from the next highest priority queue.

4. The jobs within each queue are sorted into ascending order by requested CPU time (`cput`). The shortest job is placed first.

5. Jobs which have been queued for more than one day will be considered *starving* and extra measures will be taken to attempt to run them.

6. Any queue whose name starts with "`ded`" is treated as a dedicated time queue (see discussion below). Sample dedicated time file (*PBS_HOME/*`sched_priv/dedicated_time`) is included in the installation.

7. Prime time is set to 6:00 AM - 5:30 PM. Any holiday is considered non-prime. Standard U.S. Federal holidays for the year 2004 are provided in the file *PBS_HOME/*`sched_priv/holidays`. These dates should be adjusted yearly to reflect your local holidays.

8. In addition, the scheduler utilizes the following parameters and resources in making scheduling decisions:

| **Object** | **Attribute/Resource** | **Comparison** |
|---|---|---|
| server, queue & node | `resources_available` | >= resources requested by job |
| server, queue & node | `max_running` | >= number of jobs running |
| server, queue & node | `max_user_run` | >= number of jobs running for a user |
| server, queue & node | `max_group_run` | >= number of jobs running for a group |
| server & queue | `max_group_res` | >= usage of specified resource by group |

| Object | Attribute/Resource | Comparison |
|---|---|---|
| server & queue | `max_user_res` | >= usage of specified resource by user |
| server & queue | `max_user_res_soft` | >= usage of specified resource by user (see “Hard versus Soft Limits” on page 52) Not enabled by default. |
| server & queue | `max_user_run_soft` | >= maximum running jobs for a user (see “Hard versus Soft Limits” on page 52) Not enabled by default. |
| server & queue | `max_group_res_soft` | >= usage of specified resource by group (see “Hard versus Soft Limits” on page 52) Not enabled by default. |
| server & queue | `max_group_run_soft` | >= maximum running jobs for a group (see “Hard versus Soft Limits” on page 52) Not enabled by default. |
| queue | `started` | = true |
| queue | `queue_type` | = execution |
| job | `job_state` | = queued / suspended |
| node | `loadave` | < configured limit (default: not enabled) |
| node | `arch` | = type requested by job |
| node | `host` | = name or property requested by job |

## 8.2 New Scheduler Features

With each PBS Pro release, new features are added to the Standard Scheduler. This section discusses major new scheduler features available in PBS Pro 5.4, and how you may best take advantage of these changes.

### 8.2.1 Fairshare Enhancement

This new feature provides a site with control over the behavior of the Scheduler in the event that a submitted job possesses zero (0) shares. For details see the discussion on page 130 of the following Scheduler directive:

```
fairshare_enforce_no_shares
```

### 8.2.2 Per-resource Group Limits

A new maximum group limit that may be set on any resource is now enforced by the Scheduler. This new group limit can be specified as either a hard or a soft limit. For details review `max_group_res` in section 6.3 “Hard versus Soft Limits” on page 52.

## 8.3 Scheduler Configuration Parameters

To tune the behavior of the Standard Scheduler, change directory to *`PBS_HOME/`*`sched_priv` and edit the scheduling policy configuration file `sched_config`. This file controls the scheduling policy (the order in which jobs run). The format of the `sched_config` file is:

```
name: value [prime | non_prime | all | none]
```

*`name`* can not contain any whitespace, but *`value`* may if the string is double-quoted. *`value`* can be: true | false | number | string. Any line starting with a “#” is a comment, and is ignored. The third field allows you to specify that the setting is to apply during primetime, non-prime-time, or all the time. A blank third field is equivalent to “`all`” which is both prime- and non-primetime. Note that the *value* and *all* are case-sensitive, but common cases are accepted, e.g. “`TRUE`”, “`True`”, and “`true`”.

**Important:** Note that some Scheduler parameters have been deprecated, either due to new features replacing the old functionality, or due to automatic detection and configuration. Such deprecated parameters are no longer supported, and should ***not*** be used as they may cause conflicts with other parameters.

The available options for the Standard Scheduler, and the default values, are as follows.

`assign_ssinodes`

Deprecated. Such configuration now occurs automatically.

`backfill` boolean: Instead of draining the system until the starving job runs, the Scheduler will attempt to backfill smaller jobs around the starving jobs. It will first attempt to schedule other starving

jobs around it, before moving onto normal jobs. The help_starving_jobs parameter needs to be enabled in conjunction with this parameter. Note: this feature only applies to starving jobs, and is only useful if the users specify a walltime limit during submission.
Default: true all

backfill_prime

boolean: Directs the scheduler not to run jobs which will overlap into primetime or non-primetime. This will drain the system before primetime or non-primetime starts, assisting with the problem of jobs starting before prime/non-prime and impacting the job start time. See also prime_spill.
Default: false all

by_queue boolean: If true, the jobs will be run queue by queue; if false, and round_robin is enabled, then round_robin is enforced, otherwise the entire job pool in the Server is looked at as one large queue. See also round_robin.
Default: true all

cpus_per_ssinode

Deprecated. Such configuration now occurs automatically.

dedicated_prefix

string: Queue names with this prefix will be treated as dedicated queues, meaning jobs in that queue will only be considered for execution if the system is in dedicated time as specified in the configuration file *PBS_HOME*/sched_priv/dedicated_time. See also section 8.4 “Defining Dedicated Time” on page 136.
Default: ded

fair_share boolean: This will enable the fair share algorithm. It will also turn on usage collecting and jobs will be selected based on a function of their recent usage and priority(shares). See also section 8.9 “Enabling Fairshare” on page 141.
Default: false all

fairshare_entity

string: Specifies the job attribute to use as the “entity” for which fairshare usage data will be collected. (Can be any valid PBS job attribute, such as “euser”, “egroup”, “Account_Name”, or “queue”.)
Default: euser

`fairshare_enforce_no_shares`

boolean: If this option is enabled, jobs whose entity has zero shares will never run. Requires `fair_share` to be enabled.
Default: `false`

`fairshare_usage_res`

string: Specifies the resource to collect and use in fairshare calculations and can be any valid PBS resource, including user-defined resources. See also section 8.9.1 “Fairshare Usage Data Collection” on page 142.
Default: “`cput`”

`half_life`

time: The half life for fair share usage; after the amount of time specified, the fairshare usage will be halved. Requires that `fair_share` be enabled. See also section 8.9 “Enabling Fairshare” on page 141.
Default: `24:00:00`

`help_starving_jobs`

boolean: Setting this option will enable starving jobs support. Once jobs have waited for the amount of time given by `max_starve` they are considered starving. If a job is considered starving, then no jobs will run until the starving job can be run, unless backfilling is also specified. To use this option, the `max_starve` configuration parameter needs to be set as well. See also `backfill, max_starve`.
Default: `true all`

`job_sort_key`

string: Selects how the jobs should be sorted. (Replaced `sort_by` from previous versions of PBS Pro.) `job_sort_key` can be used to sort by either resources or by special case sorting routines. Multiple `job_sort_key` entries will be applied sequentially. The `HIGH` option implies descending sorting, `LOW` implies ascending. See example for details.
Syntax: `job_sort_key: “PBS_resource HIGH|LOW”`
Default: “`cput low`”

There are three special case sorting routines, that can be used instead of a specific PBS resource:

| Special Sort | Description |
|---|---|
| `fair_share_perc HIGH` | Sort based on the values in the resource group file. This should only used if strict priority sorting is needed. **Do not enable `fair_share_perc` sorting if using the `fair_share` scheduling option.** (This option was previously named “`fair_share`” in the deprecated `sort_by` parameter). See also section 8.10 “Enabling Strict Priority” on page 146 |
| `preempt_priority HIGH` | Sort jobs by preemption priority |
| `sort_priority HIGH\|LOW` | Sort by the job priority attribute (i.e. as set by the user via the “-p” option to the `qsub` command). |

The following example illustrates using resources as a sorting parameter. Note that for each, you need to specify HIGH (descending) or LOW (ascending). Also, note that *resources* must be a quoted string.

```
job_sort_key: "ncpus HIGH" all
job_sort_key: "mem LOW" prime
```

`key` Deprecated. Use `job_sort_key`.

`load_balancing`

boolean: If set, the Scheduler will balance the computational load across the nodes. The load balancing takes into consideration the load on each node as well as all resources specified in the “resource” list. This option only applies to SMP jobs, *not* multi-node jobs. See `smp_cluster_dist`.
Default: `false all`

`load_balancing_rr`

Deprecated. To duplicate this setting, enable `load_balancing` and set `smp_cluster_dist` to `round_robin`.

`log_filter` integer: Defines which event types to filter from the daemon logfile.

The value should be set to the bitwise OR of the event classes which should be filtered. (A value of 0 specifies maximum logging.) See also section 10.12 “Use and Maintenance of Logfiles” on page 183.
Default: `256` (DEBUG2)

`max_starve`
time: The amount of time before a job is considered starving. This variable is used only if `help_starving_jobs` is set.
Format: HH:MM:SS
Default: `24:00:00`

`mem_per_ssinode`
Deprecated. Such configuration now occurs automatically.

`mom_resources`
string: This option is used to be able to query the MOMs to set the value `resources_available.res` where res is a site-defined resource. Each mom is queried with the resource name and the return value is used to replace `resources_available.res` on that node. See also section 9.3 “Configuring Per-node Custom Resources” on page 151.

`node_sort_key`
string: Defines sorting on `resources_available` values on nodes (i.e. total amounts, not free amounts). Options and usage are same as for `job_sort_key`, except only one special case sorting algorithm is applicable here: `sort_priority`.
Default: “`sort_priority HIGH`”

`nonprimetime_prefix`
string: Queue names which start with this prefix will be treated as non-primetime queues. Jobs within these queues will only run during non-primetime. Primetime and nonprimetime are defined in the `holidays` file. See also “Defining Primetime and Holidays” on page 137.
Default: `np_`

`peer_queue`
string: Defines the mapping of a remote queue to a local queue for Peer Scheduling. For details, see section 8.11 “Enabling Peer Scheduling” on page 146.
Default: unset

`preemptive_sched`
string: Enable job preemption. See section 8.8 “Enabling Pre-

emptive Scheduling” on page 139 for details.
Default: `true all`

`preempt_checkpoint`

Deprecated. Add “C” to `preempt_order` parameter.

`preempt_fairshare`

Deprecated. Add “fairshare” to `preempt_prio` parameter.

`preempt_order`

quoted list: Defines the order of preemption methods which the scheduler will use on jobs. This order can change depending on the percentage of time remaining on the job. The ordering can be any combination of S C and R (for suspend, checkpoint, and requeue). The usage is an ordering (SCR) optionally followed by a percentage of time remaining and another ordering. Note, this has to be a quoted list("").
Default: `SCR`

```
preempt_order: "SR"
# or
preempt_order: "SCR 80 SC 50 S"
```

The first example above specifies that PBS should first attempt to use suspension to preempt a job, and if that is unsuccessful, then requeue the job. The second example says if the job has between 100-81% of requested time remaining, first try to suspend the job, then try checkpoint then requeue. If the job has between 80-51% of requested time remaining, then attempt suspend then checkpoint; and between 50% and 0% time remaining just attempt to suspend the job.

`preempt_prio` quoted list: Specifies the ordering of priority of different preemption levels. Two or more job types may be combined at the *same* priority level with a “+” between them (no whitespace). Comma-separated preemption levels are evaluated left to right, with each having lower priority than the preemption level preceding it. The table below lists the six preemption levels. Note that any level not specified in the `preempt_prio` list will be ignored.
Default: `“express_queue, normal_jobs”`

| | |
|---|---|
| express_queue | Jobs in the preemption (e.g. “express”) queue(s) preempt other jobs (see also `preempt_queue_prio`). |
| starving_jobs | When a job becomes starving it can preempt other jobs (requires `preempt_starving` be set to `true`). |
| fairshare | When a job exceeds it's fairshare. |
| queue_softlimits | Jobs which are over their queue soft limits |
| server_softlimits | Jobs which are over their server soft limits |
| normal_jobs | The preemption level into which a job falls if it does not fit into any other specified level. |

For example, the first line below states that starving jobs have the highest priority, then normal jobs, and jobs over their fairshare limit are third highest. The second example shows that starving jobs that are also over their fairshare limit are lower priority than normal jobs.

```
preempt_prio: "starving_jobs, normal_jobs, fairshare"
# or
preempt_prio: "normal_jobs, starving_jobs+fairshare"
```

`preempt_queue_prio`

integer: Specifies the minimum queue priority required for a queue to be classified as an express queue.
Default: `150`

`preempt_requeue` Deprecated. Add an “R” to `preempt_order` parameter.

`preempt_starving`

Deprecated. Add "starving_jobs" `preempt_prio` parameter.

`preempt_suspend` Deprecated. Add an “S” to `preempt_order` parameter.

`primetime_prefix`

string: Queue names which start with this prefix will be treated as primetime queues. Jobs will only run in these queues during primetime. Primetime and non-primetime are defined in the

holidays file. See also “Defining Primetime and Holidays” on page 137.
Default: p_

prime_spill time: Specifies the amount of time a job can "spill" over from non-primetime into primetime or from primetime into non-primetime. Can be set for any execution queue. For example, if a job is in a primetime queue and it is currently primetime, but the job would cross over into non-primetime by one hour, the job would not run if prime_spill were set to less than one hour. Note, this option is only meaningful if backfill_prime is true. Also note that this option can be separately specified for prime- and non-primetime.
Default: 00:00:00

For example, the first setting below means that non-primetime jobs can spill into prime time by 1 hour. However the second setting means that jobs in either prime/non-prime can spill into the other by 1 hour.

```
prime_spill: 1:00:00 prime
# or
prime_spill: 1:00:00 all
```

resources string: Specifies those resources which are to be considered when scheduling SMP jobs (not multi-node jobs). Limits are set by setting resources_available.resourceName on the Server objects (nodes, queues, and servers). The scheduler will consider numeric (integer or float) items as consumable resources and ensure that no more are assigned than are available (e.g. ncpus or mem). Any string resources will be compared using string comparisons (e.g. arch). See the description of the Server parameter resources; see also section “Dynamic Consumable Resources” on page 137.
Default: “ncpus, mem, arch” (number CPUs, memory, architecture)

round_robin boolean: If true, the queues will be cycled through in a circular fashion, attempting to run one job from each queue. If false, attempts to run all jobs from the current queue before processing the next queue. See by_queue.
Default: false all

`server_dyn_res` string: Directs the Scheduler use an external program and replace the server's `resources_available` values with new values returned by the external program. See section 8.5 "Dynamic Consumable Resources" on page 137 for details of usage.

`smp_cluster_dist`
string: Specifies how jobs should be distributed to all nodes of the cluster. Options are: `pack`, `round_robin`, and `lowest_load`. `pack` means keep putting jobs onto one node until it is "full" and then move onto the next. `round_robin` is to put one job on each node in turn before cycling back to the first one. `lowest_load` means to put the job on the lowest loaded node. See also section 8.7 "SMP Cluster Support" on page 138.
Default: `pack all`

`sort_by` Deprecated. Use `job_sort_key`.

`strict_fifo` boolean: If `true`, jobs will be run in a strict FIFO order. This means if a job fails to run for any reason, no more jobs will run from that queue/Server during that scheduling cycle. If `strict_fifo` order is desired, `job_sort_key` should not be used, as it will change the meaning from strict FIFO to strict ordering. Also see the Server parameter `resources_available`, and the parameters `help_starving_jobs` and `backfill` above.
Default: `false all`

`sync_time` time: The amount of time between writing the fair share usage data to disk. Requires `fair_share` to be enabled.
Default: `1:00:00`

`unknown_shares` integer: The amount of shares for the "unknown" group. Requires `fair_share` to be enabled. See also section 8.9 "Enabling Fairshare" on page 141.
Default: `10`

## 8.4 Defining Dedicated Time

The file *PBS_HOME*/`sched_priv/dedicated_time` defines the dedicated times for the scheduler. During dedicated time, only jobs in the dedicated time queues can be run (see `dedicated_prefix` in section 8.3 “Scheduler Configuration Parameters” on page 128). The format of entries is:

```
# From Date-Time   To Date-Time
# MM/DD/YYYY HH:MM MM/DD/YYYY HH:MM
# For example
04/15/2004 12:00 04/15/2004 15:30
```

**Important:** To force the Scheduler to re-read the dedicated time file (which is needed after modifying the file), send the Scheduler daemon process a SIGHUP.

## 8.5 Dynamic Consumable Resources

It is possible to schedule resources where the number or amount available is outside of PBS's control. Such resources are called “dynamic consumable resources”, and are discussed in detail, with example, in Chapter 9 “Customizing PBS Resources” on page 149.

## 8.6 Defining Primetime and Holidays

To have the scheduler enforce a distinction between primetime and non-primetime, as well as enforcing non-primetime scheduling policy during holidays, edit the *`PBS_HOME/`*`sched_priv/holidays` file to handle prime time and holidays. The ordering is important. Any line that begins with a "*" or a “#” is considered a comment. The format of the holidays file is:

```
YEAR YYYY     This is the current year.
<day> <prime> <nonprime>
<day> <prime> <nonprime>
```

*Day* can be `weekday`, `saturday`, or `sunday.` *Prime* and *nonprime* are times when prime or non-prime time start. Times can either be HHMM with no colons(:) or the word "`all`" or "`none`"

```
<day> <date> <holiday>
```

*Day* is the day of the year between 1 and 365 ( e.g. “1”). *Date* is the calendar date (e.g. “Jan 1”). *Holiday* is the name of the holiday (e.g. “New Year’s Day”).

```
HOLIDAYFILE_VERSION1
YEAR    2004
*            Prime   Non-Prime
* Day        Start   Start
*
  weekday    0600    1730
  saturday   none    all
  sunday     none    all
*
* Day of    Calendar      Company Holiday
* Year       Date          Holiday
    1        Jan 1         New Year's Day
   19        Jan 19        Dr. M.L. King Day
   47        Feb 16        President's Day
  152        May 31        Memorial Day
  186        Jul 4         Independence Day
  250        Sep 6         Labor Day
  285        Oct 11        Columbus Day
  316        Nov 11        Veteran's Day
  330        Nov 25        Thanksgiving
  360        Dec 25        Christmas Day
```

## 8.7 SMP Cluster Support

The Standard Scheduler schedules SMP clusters in an efficient manner. Instead of scheduling only via load average of nodes, it takes into consideration the resources specified at the server, queue, and node level. Furthermore, the Administrator can explicitly select the resources to be considered in scheduling via an option in the Scheduler's configuration file (`resources`). The configuration parameter `smp_cluster_dist` allows you to specify how nodes are selected. The available choices are `pack` (pack one node until full), `round_robin` (put one job on each node in turn), or `lowest_load` (put one job on the lowest loaded node). The `smp_cluster_dist` parameter should be used in conjunction with `node_sort_key` to ensure efficient scheduling.

To use these features requires two steps: setting resource limits via the Server, and specifying the scheduling options. Resource limits are set using the `resources_available` parameter of nodes via `qmgr` just like on the server or queues. For example, to set maximum limits on a node called "node1" to 10 CPUs and 2 GB of memory:

```
Qmgr: set node node1 resources_available.ncpus = 10
Qmgr: set node node1 resources_available.mem=2GB
```

**Important:** Note that by default both `resources_available.ncpus` and `resources_available.mem` are set to the physical number reported by MOM on the node. Typically, you do not need to set these values, unless you do not want to use the actual values reported by MOM.

Next, the Scheduler options need to be set. For example, to enable SMP cluster scheduler to use the "round robin" algorithm during primetime, and the "pack" algorithm during non-primetime, set the following in the Scheduler's configuration file:

```
smp_cluster_dist: round_robin prime
smp_cluster_dist: pack        non_prime
```

Finally, specify the resources to use during scheduling:

```
resources: "ncpus, mem, arch"
```

## 8.8 Enabling Preemptive Scheduling

PBS provides the ability to preempt currently running jobs in order to run higher priority work. Preemptive scheduling is enabled by setting several parameters in the Scheduler's configuration file (discussed below, and in "Scheduler Configuration Parameters" on page 128). Jobs utilizing advance reservations are not preemptable. If high priority jobs (as defined by your settings on the preemption parameters) can not run immediately, the Scheduler looks for jobs to preempt, in order to run the higher priority job. A job can be preempted in several ways. The Scheduler can suspend the job (i.e. sending a SIGSTOP signal), checkpoint the job (if supported by the underlying operating system, or if the Administrator configures site-specific checkpointing, as described in "Site-Specific Job Checkpoint / Restart" on page 109), or requeue the job (a requeue of the job terminates the job and places it back into the queued state). The Administrator can choose the order of these attempts via the `preempt_order` parameter.

There are several Scheduler parameters to control preemption. The preemptive_sched parameter turns preemptive scheduling on and off. You can specify the relative priority between different types of jobs using preempt_prio. You set what queue priority is preemptive with the preempt_queue_prio parameter. The parameter preempt_order allows you to specify the order in which these methods should be applied. If one preemption method fails, the Scheduler tries the next. If the scheduler cannot find enough work to preempt in order to run a given job, it will not preempt any work.

The preempt_prio parameter provides a means of specifying the order of precedence that preemption should take. The ordering is evaluated from left to right. One special name (normal_jobs) is the default (This is the preemption level into which a job falls if it does not fit into any other specified level.). If you want normal jobs to preempt other lower priority jobs, put normal_job before them in the preempt_prio list. If two or more levels are desired for one priority setting, the multiple levels may be indicated by putting a '+' between them. A complete listing of the preemption levels is provided in the scheduler tunable parameters section above.

Soft run limits can be set or unset via qmgr. If unset, any job that would have been subject to this limits will instead be considered a "normal job". The job sort preempt_priority will sort jobs by their preemption priority. Note: It is a good idea to put preempt_priority as the primary sort key (i.e. job_sort_key) if the preempt_prio parameter has been modified. This is especially necessary in cases of when soft limits are used.

**Important:** Note that any queue with a priority 150 (default value) or higher is treated as an express (i.e. high priority) queue.

Below is an example of (part of) the Scheduler’s configuration file showing how to enable preemptive scheduling and related parameters. Explanatory comments precede each configuration parameter.

```
# turn on preemptive scheduling
preemptive_sched:       TRUE ALL

# set the queue priority level for express queues
preempt_queue_prio:     150

# specific the priority of jobs as: express queue (highest)
# then starving jobs, then normal jobs, followed by jobs
# who are starving but the user/group is over a soft limit,
# followed by users/groups over their soft limit but not
# starving
#
preempt_prio: "express_queue, starving_jobs, normal_jobs,star
ving_jobs+server_softlimits, server_softlimits"

# specify when to each preemption method. If the first
# method fails, try the next method. If a job is has
# between 100-81% time remaining, try to suspend, then
# checkpoint then requeue. From 80-51% suspend and then
# checkpoint, but don't requeue. If between 50-0% time
# remaining, then just suspend it.
preempt_order: "SCR 80 SC 50 S"
```

## 8.9 Enabling Fairshare

In PBS fairshare “entities” (such as user names) are placed in a fairshare group file (*PBS_HOME*/sched_priv/resource_group). The file is read in and a tree is created. The tree consists of groups (nodes) and entities (leaves). Groups can contain groups. Every node and leaf has a number of shares associated with it. Priorities can be derived from these shares by taking a ratio of them to the rest of the shares in the same group.

The fairshare capability allows an Administrator to set which PBS resource is collected for fairshare usage. If unspecified, the default it is cput. The resource to use is specified in the sched_config file with the parameter: fairshare_usage_res. An Administrator can specify what the fairshare "entity” to use as well. It can be set to any PBS job attribute (i.e. euser, egroup, Account_Name, etc) via the sched_config file parameter: fairshare_entity. The Administrator can also use the pbsfs command to display the fairshare usage tree. (See the pbsfs(8B) manual page, or “The pbsfs (PBS Fairshare) Command” on page 199 for usage details.)

### 8.9.1 Fairshare Usage Data Collection

MOM polls for resource usage every two minutes and reports that data back to the Server. The Server then stores this in the `resources_used` attribute on the job (one `resources_used.`*`Res`* for each resource). The Scheduler queries this information from the Server and uses it in the calculation of `fairshare_entity` usage.

**Important:** Given the large granularity of the data that the scheduler use for fairshare (i.e. polled by MOM on a two-minute interval), the Administrator must be careful in setting the value of fairshare `half_life`. If `half_life` is set to under two minutes, the scheduler could decay the data several times before getting any new data. This could lead to unwanted results. For example, if the `half_life` is set to 10 seconds, the scheduler will get two minutes worth of data, and decay it 12 times before it gets more.

During each scheduling cycle in which the Scheduler runs a job, it will temporarily add the full requested resource usage amount to the jobs's `fairshare_entity`. This is applied to that scheduling cycle only. Every cycle after that, the scheduler will find the delta between the resources used from the previous cycle and the current cycle and add that to the entity's usage. The scheduler will write its fairshare usage data to the usage database periodically. The time between synchronization writes is set in the `sched_config` file (`sync_time`). The default is 1 hour.

**Important:** If the scheduler is killed, it will lose all data since the last synchronization. However, this data loss can be avoided if the Administrator first issues a kill -HUP prior to killing the Scheduler. Doing so will cause the scheduler to write out the fairshare usage data immediately.

The Administrator should note that the Scheduler tracks used resources for job on every scheduling cycle. If a job ends between two cycles, the scheduler will lose that small amount of data (between the previous cycle and the end of the job). This could affect the accuracy of fairshare usage data. (This only applies to fairshare usage. The PBS accounting records will have the correct job usage data.) If more accurate fairshare usage data is required, adjusting the `scheduler_iteration` Server parameter down is recommended. Doing so will force a scheduling iteration at the frequency specified.

### 8.9.2 Hierarchical Fairshare Tree

PBS Pro uses a hierarchical fairshare tree. This permits enforcing fairshare across groups rather than just individuals. For example, consider a situation where Sally and Bob are both in group A and Mark is in group B and Sally uses 100% of the machine for a month. In prior releases of PBS Pro, Bob and Mark would compete for cycles since neither of them had any usage (i.e. comparison of individuals within the fairshare tree). With a hierarchal fairshare tree, since group A has all of Sally's usage, group B has much higher priority. Thus Mark's jobs would run before Bob's.

**Important:** The fairshare implementation uses the "unknown" group to capture the usage of users who are not configured into the resources file. Set the number of shares for the "unknown" group using the `unknown_shares` configuration parameter, as discussed on page 136.

Here is another example. Say you have three nodes/leaves at one level with shares of 10, 20, and 10. The first entity/group has a priority of 25% or 10/40, the second has 50% or 20/40 and so on. A node with children can establish priorities among them via shares. So if in the example above the second group (50%) is actually a group with 4 users and all the users have equal shares, then each user has 1/4 of 50% or 12.5% of the machine.

If fairshare or strict priority is going to be used, the resource group file *`PBS_HOME/`*`sched_priv/resource_group` may need to be edited. (If all users are considered equal, this file doesn't need to be edited.) Each line of the file should use the following format:

```
name unique_ID parent_group shares
```

`name` The name of the specified entity or group

`unique_id` A unique numeric identifier for the group or entity

`parent_group` The name of the parent resource group to which this user/group belongs. The root of the share tree is called `root` and is added automatically to the tree by the Scheduler.

`shares` The number shares (or priority) the user/group has in the specified resource group.

If you wish to specify how individual users should be ranked against each other, only user entries are required in the `resources_group` file, as shown the following example:

```
tom    60      root    5
bob    61      root    15
roy    62      root    15
jim    63      root    10
rob    64      root    25
ann    65      root    10
dan    66      root    10
```

Another option is to divide shares into “groups”, and then name the users who are members of each group. The following example illustrates this configuration:

```
grp1    50      root    10
grp2    51      root    20
grp3    52      root    10
grp4    53      grp1    20
grp5    54      grp1    10
grp6    55      grp2    20
tom     60      root    5
bob     61      grp1    10
roy     62      grp2    10
jim     63      grp6    10
rob     64      grp6    10
ann     65      grp6    20
dan     66      grp3    10
sam     67      grp4    10
erin    68      grp4    10
will    69      grp5    10
```

The above example shows three top level groups (whose parent is the root group), three sub-groups (whose parent is a user-defined group), and 10 users belonging to different groups. The box below shows how the individual shares or percentages are calculated for groups and users, using the values from the above example. First we start with the root group as having 100% of the machine, but it has three member groups and one user, with share values of 10, 20, 10, and 5, respectively, for a total of 45 shares. The second line below shows calculating the actual percentage of the resources that group grp1 has. Specifically, the specified 10 shares are out of a total of 45 specified shares, or 22% of the total. The members of group grp1 (i.e. user bob and groups grp4 and grp5) each have specific shares of their group, all of which together total 40 shares specified for this group (i.e. 10+20+10). Thus their specified shares are of this total, resulting in these three entities having 25%, 50%, and 25%, respectively, of group grp1. We can further calculate how these share convert into shares of the total system by multiplying against the total percentage possessed by their group, grp1.

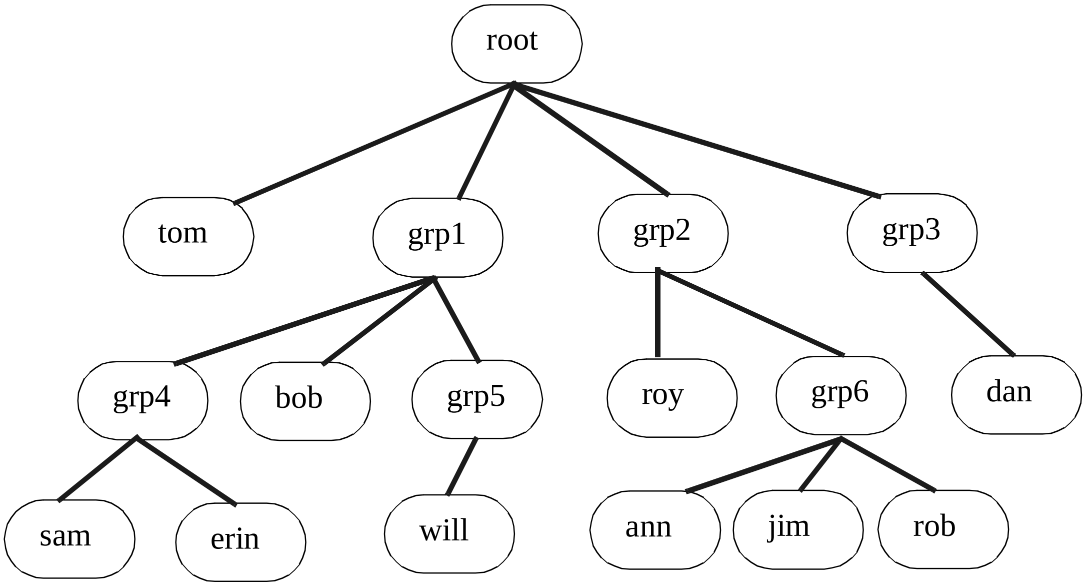
Two ways to visualize this share tree are given below, as a table of relative percentages, and as a graphical representation of the relationship between users and groups within the fairshare tree.

```
root 100%
    grp1 10 10/45 = 100% * 22% = 22%
        bob 10 10/40 (25%) * 22% = 5.5%
        grp4 20 20/40 (50%) * 22% = 11%
            sam  10 10/20 (50%) * 11% = 5.5%
            erin 10 10/20 (50%) * 11% = 5.5%
        grp5 10 10/40 (25%) * 22% = 5.5%
            will 10 100% * 5.5% = 5.5%

    grp2 20 20/45 = 100% * 44% = 44%
        roy 10 10/30 (33%) * 44% = 14.5%
        grp6 20 20/30 (66%) * 44% = 29%
             jim 10 10/40 (25%) * 29% = 7.2%
             rob 10 10/40 (25%) * 29% = 7.2%
             ann 20 20/40 (50%) * .29% = 14.5%

    grp3 10 10/45 = 22%
        dan 10 100% * 22% = 22%

    tom 5 5/45 = 11%
```



## 8.10 Enabling Strict Priority

Not to be confused with fairshare (which considers past usage of each entity in the selection of jobs), the Standard Scheduler offers a sorting key called “fair_share_perc” (see also section 8.3 “Scheduler Configuration Parameters” on page 128). Selecting this option enables the sorting of jobs based on the priorities specified in the fair share tree (as defined above in the resource_group file). A simple share tree will suffice. Every user’s parent_group should be root. The amount of shares should be their desired priority. unknown_shares (in the scheduler’s configuration file) should be set to one. Doing so will cause everyone who is not in the tree to share one share between them, making sure everyone else in the tree will have priority over them. Lastly, job_sort_key must be set to “fair_share_perc HIGH”. This will sort by the fair share tree which was just set up. For example:

```
usr1    60      root    5
usr2    61      root    15
usr3    62      root    15
usr4    63      root    10
usr5    64      root    25
usr6    65      root    30
```

## 8.11 Enabling Peer Scheduling

PBS Pro includes a feature to have different PBS installations automatically run jobs from each other’s queues. This provides the ability to dynamically load-balance across multiple, separate PBS installations. (These cooperating PBS installations are referred to as “Peers”.) When this feature is enabled and resources are available, PBS can “pull” jobs from one or more (remote) Peer Servers and run them locally. No job will be moved if it cannot run immediately.

**Important:** When configuring Peer Scheduling, it is *strongly* recommended to use the same versions of PBS Pro at all Peer locations.

This functionality is implemented by mapping a remote Peer’s queue to a local queue. When the scheduler determines that a remote job can run locally, it will move the job to the specified queue on the local Server and then run the job.

**Important:** Note that the Peer queue mapping must be between *execution* queues, not *route* queues.

This release of the Peer Scheduling features requires a flat user namespace, meaning that there is an assumption that user “joe” on the remote Peer system(s) is the same as user “joe” on the local system.

To configure Peer Scheduling, the following setting needs to be made in the local Server and all Peer Servers, via `qmgr`:

```
qmgr>  set server flatuid = true
```

Furthermore, in order to pull jobs from a remote Peer to the local Server, the remote Peer Server needs to grant manager access to the local Peer Server (and *vice versa* if you wish to permit jobs to move in the opposite direction).

```
qmgr> set server managers += root@localServer.domain.com
```

Lastly, you need to configure the local Scheduler. Add one or more `peer_queue` entries to the local Scheduler’s configuration file, mapping a remote queue to a local queue. The format is:

```
peer_queue: "local_queue remote_queue@remote_server.domain"
```

```
peer_queue: “workq workq@remote_server.domain.com”
peer_queue: “peerq workq@other_server.different.com”
```

Since the scheduler maps the remote jobs to a local queue, any moved jobs are subject to the policies of the queue they are moved into. If remote jobs are to be treated differently then local jobs, this can be done on the queue level. A queue can be created exclusively for remote jobs and this will allow queue level policy to be set for remote jobs. For example, you can set a priority value on your queues, and enable sorting by priority to ensure that remotely queued jobs are always lower (or higher!) priority than locally queued jobs.

### 8.11.1 Peer Scheduling and Failover Configuration

When setting up the Scheduler for Peer Scheduling, and the peer is in a Failover Server configuration, it is necessary to define two remote peer queues. For example, say the

Scheduler is set to pull jobs into local queue `workq` from the peer `workq@SVR1.example.com`. The `sched_config` file would have the following entry:

```
peer_queue: “workq workq@SVR1.example.com”
```

Now if you configure host SVR1 into a Failover configuration, where SVR1 is the Primary and the Secondary is SVR2, you will need to add an additional entry (for SVR2) to the Scheduler `sched_config` file, as follows:

```
peer_queue: “workq workq@SVR1.example.com”
peer_queue: “workq workq@SVR2.example.com”
```

### 8.11.2 Peer Scheduling and Group Limit Enforcement

There is a condition when using Peer Scheduling in which group hard limits could be exceeded. The situation occurs when the `egroup` attribute of the job changes when a job is moved from a remote (peer) system to the local system. All the hard limit checking is preformed on the old system prior to moving the job, and not on the new (local group). This means that if a job is in group `foo` on the remote system and group `foo` on the local system has not exceeded its limit, the job may be selected to run. But if it is selected to run, *and* the job’s default group on the local system is different (say its `bar`), the job will be run even if the limit on group `bar` has been exceeded. This situation can also occur if the user explicitly specifies a group via `qsub -W group_list`.

**Important:** Thus, it is recommended to advise users to *not* use the `qsub` options “`-u` *`user_list`*” or “`-W group_list=`*`groups`*” in conjunction with Peer Scheduling.

# Chapter 9
# Customizing PBS Resources

It is possible for the PBS Manager to define new resources within PBS. The primary use of this feature is to add site-specific resources, such as to manage software application licenses. This chapter discusses the steps involved in specifying such new resources to PBS, followed by several detailed examples of use.

Once new resources are defined, jobs may request these new resources and the Scheduler will consider the new resources in the scheduling policy. Using this feature, it is possible to schedule resources where the number or amount available is outside of PBS's control.

## 9.1 System-wide vs. Per-node Custom Resources

There are two types of custom resources: *system-wide* and *per-node*. As the names imply, you create a system-wide resource when there is a single entity to manage across the entire cluster (such as an external software license manger). You would create per-node resources when the resource you wish to manage exists locally on individual nodes (such as local scratch file system).

If you have a resource (such as a software license manager) that operates system-wide you should use the system-wide custom resource feature. For these, the Server maintains the status of the custom resource, and the Scheduler queries the Server for the resource.

For per-node custom resources, the Scheduler will perform a resource query to each MOM to get the current availability for the resource and use that value for scheduling. If the MOM returns a value it will replace the `resources_available` value reported by the Server. If the MOM returns no value, the value from the Server is kept. If neither specify a value, the Scheduler sets the resource value to 0.

## 9.2 Defining New Custom Resources

To define one or more new resources, the Administrator creates the Server resource definition file, *`PBS_HOME`*`/server_priv/resourcedef`. Each line in the file defines a new resource. The format of each line is:

```
RESOURCE_NAME [type=RTYPE] [flag=FLAGS]
```

`RESOURCE_NAME` is any string made up of alphanumeric characters, starting with an alphabetic character. The underscore character, “_”, and the hyphen, “-”, are also allowed.

`RTYPE` is the type of the resource value, which must be one of the following key words:

long the value is a long integer

float the value is a floating point number

size the value is a integer number following by a suffix denoting magnitude `k`, `m`, `g`, `t` and `b` for bytes or `w` for words.

string a null terminated string

If not specified, the resource will default to type `long`.

`FLAGS` is a set of characters which indicate if and how the Server should accumulate the requested amounts of the resource in the attribute `resources_assigned` when the job is run. The value of `flag` is the concatenation of one or more of the following letters:

q the amount is tracked at the Queue and Server level

n the amount is tracked at the Node level, for all nodes assigned to the job

f the amount is tracked at the Node level for only the first node allocated to the job

If FLAGS is not specified, the resource will not be accumulated in `resources_assigned`.

**Important:** If you are defining a new system-wide resource, the Server will be managing the resource, therefore the `flag` option is required. However, if you are defining a per-node resource, the individual MOMs will be responding to the resource requests, and therefore the `flag` option may be left unspecified.

Once you have defined the new resource(s), you must restart the Server with the following command in order for these changes to take effect.

On UNIX:

```
# qterm -t quick
# /usr/pbs/sbin/pbs_server
```

On Windows:

```
Admin> qterm -t quick
Admin> net start pbs_server
```

Once this is done, the users will be able to submit jobs requesting the new resource, using the normal “`qsub -l RESOURCE=value` ...” syntax to which they are accustomed. See also section 9.5 “Examples” on page 154.

## 9.3 Configuring Per-node Custom Resources

If the new resource you are adding is a per-node resource, then you will also need to configure each MOM such that she will be able to answer the resource query requests from the Scheduler. However, if the new resource is not a per-node resource, you should skip this section, and continue with section 9.4 “Scheduling Custom Resources” on page 153.

Each MOM can be instructed how to respond to a Scheduler resource query by adding shell escapes to the MOM configuration file *`PBS_HOME`*`/server_priv/config`. The format of a shell escape line is:

```
RESOURCE_NAME !/path/to/command [%arg1] [%arg2] [%argN]
```

`RESOURCE_NAME` is any string made up of alphanumeric characters, starting with an alphabetic character. The underscore character, “_”, and the hyphen, “-”, are also allowed. The `RESOURCE_NAME` specified should be the same as the corresponding entry in the Server’s `resourcedef` file.

The rest of the line, following the exclamation mark (“!”), is saved to be executed through the services of the `system`(3) standard library routine. The first line of output from the shell command is returned as the response to the resource query.

**Important:** On Windows, be sure to place double-quote (“ “) marks around the `/path/to/command` if it contains any whitespace characters.

The shell escape provides a means for the resource monitor to yield arbitrary information to the Scheduler and other client commands. Parameter substitution is done such that the value of any qualifier sent with the resource query, as explained below, replaces a token with a percent sign (%) followed by the name of the qualifier. For example, say the following line is in MOM’s configuration file:

```
echotest !/bin/echo %xxx %yyy
```

Then, if a query for resource “`echotest`” is sent with no qualifiers, the command executed would be “`/bin/echo %xxx %yyy`”. If one qualifier is sent, “`echotest[xxx=hi]`”, the command executed would be “`echo hi %yyy`”. If two qualifiers are sent, “`echotest [xxx=hi] [yyy=there]`”, the command executed would be “`echo hi there`". If a qualifier is sent with no matching token in the command line, “`echotest [zzz=snafu]`”, an error is reported.

**Important:** On UNIX, all changes to MOM’s `config` file should be followed by sending a HUP signal to the `pbs_mom` process, so that the configuration file will be re-read.

On Windows, simply restart MOM by doing:

```
Admin> net stop pbs_mom
Admin> net start pbs_mom
```

Typically, what follows the “!” is the full path to the script or program that you wish to be executed, in order to determine the status and/or availability of the new resource you have added.

## 9.4 Scheduling Custom Resources

The last step in creating a new custom resource is configuring the Scheduler to use your new resource. Whether you set up system-wide or per-node resources, the external site-provided script/program is run once per scheduling cycle. Multiple jobs may be started during a cycle. For any job started that requests the resource, the Scheduler maintains an internal count, initialized when the script is run, and decremented for each job started that required the resource.

To direct the Scheduler to use a new system-wide custom resource, add the `server_dyn_res` configuration parameter to the Scheduler `sched_config` file:

```
server_dyn_res: “RESOURCE_NAME !/path/to/command”
```

where `RESOURCE_NAME` should be the same as used in the Server’s `resourcedef` file. (see also section 8.3 “Scheduler Configuration Parameters” on page 128).

Alternatively, to direct the Scheduler to use a new per-node custom resource, add the `mom_resources` configuration parameter to the Scheduler `sched_config` file:

```
mom_resources: “RESOURCE_NAME”
```

where `RESOURCE_NAME` should be the same as used in the Server’s `resourcedef` file and the MOM’s `config` file. (see also section 7.2 “MOM Configuration Parameters” on page 98).

**Important:** On UNIX, all changes to the Scheduler’s `sched_config` file should be followed by sending a HUP signal to the `pbs_sched` process, so that the configuration file will be re-read, and to have the changes take effect.

On Windows, simply restart the Scheduler by doing:

```
Admin> net stop pbs_sched
Admin> net start pbs_sched
```

Because custom resources are external to PBS, they are not completely under PBS’ control. Therefore it is possible for PBS to query and find a resource available, schedule a job to run to use that job, only to have an outside entity take that resource before the job is able to use it.

## 9.5 Examples

This section provides examples which illustrate how to implement custom resources in PBS.

### 9.5.1 Per-node "scratchspace" Example

Say you have jobs that require a large amount of scratch disk space during their execution. To insure that sufficient space is available when starting the job, you first write a script that returns as a single line (with new-line) the amount of space available. This script is placed in `/usr/local/bin/scratchspace` on each node.

Next, edit the Server's resource definition file, (`PBS_HOME/server_priv/resourcedef`) adding a definition for the new resource. (See also "Defining New Resources" on page 79.) For this example, let's call our new resource "scratchspace":

```
scratchspace    type=size
```

**Important:** Note that the optional "flags" is not specified, since this is per-node resource.

Now restart the Server. On UNIX:

```
# qterm -t quick
# /usr/pbs/sbin/pbs_server
```

On Windows:

```
Admin> qterm -t quick
Admin> net start pbs_server
```

Next, configure MOM to use the `scratchspace` script by entering one line into the `PBS_HOME/mom_priv/config` file: .

```
scratchspace !/usr/local/bin/scratchspace
```

Then, under UNIX, issue `kill -HUP` to the MOM.
Under Windows, restart the MOM by doing:

```
Admin> net stop pbs_mom
Admin> net start pbs_mom
```

Finally, edit the Scheduler configuration file (`PBS_HOME/sched_priv/sched_config`), specifying this new resource that you want queried and used for scheduling:

```
mom_resources:  “scratchspace”
resources: “scratchspace”
```

Then, under UNIX, issue `kill -HUP` to the Scheduler

Under Windows, restart the Scheduler by doing:

```
Admin> net stop pbs_sched
Admin> net start pbs_sched
```

Now users will be able to submit jobs which request this new “scratchspace” resource using the normal `qsub -l` syntax to which they are accustomed.

```
% qsub -l scratchspace=100mb ...
```

The Scheduler will see this new resource, and know that it must query the different MOMs when it is searching for the best node on which to run this job.

### 9.5.2 System-wide Resource Example

A common question that arises among PBS Pro customers is regarding how to use the system-wide dynamic resources to coordinate external FlexLM license checking for applications. This example illustrates how to implement such a custom resource.

Before starting you should consider and have answers to the following questions:

How many units of a feature does the application require?
How many features are required to execute the application?
How do I query the FlexLM server to obtain the available licenses of particular features?

With these questions answered you can begin configuring PBS Pro to query the FlexLM servers for the availability of application licenses. Think of a FlexLM feature as a resource. Therefore, you should associate a resource with each feature. Thus, if you need four features to run an application, then you need four custom resources.

To continue with the example, there are four features required to execute an application, thus *PBS_HOME*/server_priv/resourcedef needs to be modified:

```
feature1   type=long
feature3   type=long
feature6   type=long
feature8   type=long
```

Once these resources have been defined, you will need to restart the PBS Server:

```
# qterm -t quick
# /usr/pbs/sbin/pbs_server
```

Now that PBS is aware of the new custom resources we can begin configuring the Scheduler to query the FlexLM server, and schedule based on the availability of the licenses.

Within *PBS_HOME*/sched_priv/sched_config the following options will need to be updated, or introduced depending on your site configuration. The 'resources:' option should already exist with some PBS default resources declared, and therefore you will want to append your new custom resources to this line, as shown below.

```
resources: "ncpus, mem, arch, feature1, feature3, feature6, feature8"
```

You will also need to add the option 'server_dyn_res:' which allows the scheduler to execute a program or script, that will need to be created, to query your FlexLM server for available licenses. Below illustrates the format for this example.

```
server_dyn_res: "feature1 !/path/to/script [arg1 argN]"
server_dyn_res: "feature3 !/path/to/script [arg1 argN]"
server_dyn_res: "feature6 !/path/to/script [arg1 argN]"
server_dyn_res: "feature8 !/path/to/script [arg1 argN]"
```

Once the `PBS_HOME/sched_priv/sched_config` has been updated, you will need to have the `pbs_sched` daemon reread the `config` file by sending a `HUP` to the PID

```
# kill -HUP pbs_sched_pid
```

The '`/path/to/script`' is very important to make this work, and has been deemed a site specific customization. Essentially, the '`/path/to/script`' needs to report the number of available licenses to the Scheduler via an echo to `stdout`. Complexity of the script is entirely site specific due to the nature of how applications are licensed. For instance, an application may require `N+8` units, where `N` is number of CPUs, to run one job. Thus, the script could perform a conversion so that the user will not need to remember how many units are required to execute a `N` CPU application.

# Chapter 10
# Administration

This chapter covers information on the maintenance and administration of PBS, and as such is intended for the PBS Manager. Topics covered include: starting and stopping PBS, security within PBS, prologue/epilogue scripts, accounting, configuration of the PBS GUIs, and using PBS with other products such as Globus.

## 10.1 pbs.conf

During the installation of PBS Pro, the `pbs.conf` file was created as either: `/etc/pbs.conf` (UNIX) or `[PBS Destination Folder]\pbs.conf` (Windows, where [PBS Destination Folder] is the path specified when PBS was installed on the windows platform, e.g., "`C:\Program Files\PBS Pro\pbs.conf`".) The installed copy of `pbs.conf` is similar to the one below.

```
PBS_START_SERVER=1
PBS_START_MOM=1
PBS_START_SCHED=1
PBS_HOME=/usr/spool/PBS
PBS_EXEC=/opt/pbs
PBS_SERVER=hostname.domain
```

This configuration file controls which daemons are to be running on the local system, directory tree location, and various runtime configuration options. Each node in a cluster should have its own pbs.conf file. The following table describes the available parameters as of the current version of PBS Pro:

| Parameters | Meaning |
|---|---|
| PBS_BATCH_SERVICE_PORT | Port Server listens on |
| PBS_BATCH_SERVICE_PORT_DIS | DIS Port Server listens on |
| PBS_CONF_SYSLOG | Controls use of syslog facility |
| PBS_CONF_SYSLOGSEVR | Filters syslog messages by severity |
| PBS_ENVIRONMENT | Location of pbs_environment file |
| PBS_EXEC | Location of PBS bin and sbin directories |
| PBS_HOME | Location of PBS working directories |
| PBS_MANAGER_GLOBUS_SERVICE_PORT | Port Globus Manager listens on |
| PBS_MANAGER_SERVICE_PORT | Port MOM Manager listens on |
| PBS_MOM_GLOBUS_SERVICE_PORT | Port Globus MOM listens on |
| PBS_MOM_HOME | Location of MOM working directories |
| PBS_MOM_SERVICE_PORT | Port MOM listens on |
| PBS_PRIMARY | Hostname of primary Server |
| PBS_RCP | Location of rcp command if rcp is used |
| PBS_SCP | Location of scp command if scp is used; setting this parameter directs PBS to use scp rather than rcp for file transport. |
| PBS_SCHEDULER_SERVICE_PORT | Port Scheduler listens on |
| PBS_SECONDARY | Hostname of secondary Server |
| PBS_SERVER | Hostname of host running the Server |
| PBS_START_SERVER | Set to 1 if Server is to run on this host |
| PBS_START_MOM | Set to 1 if a MOM is to run on this host |
| PBS_START_SCHED | Set to 1 if Scheduler is to run on this node |

## 10.2 Starting and Stopping PBS: UNIX and Linux

The daemon processes, Server, Scheduler, MOM and the optional MOM Globus, must run with the real and effective uid of root. Typically the daemons are started automatically by the system upon reboot. The location of the boot-time start/stop script for PBS varies by OS, as the following table shows.

| OS | Location of PBS Startup Script |
|---|---|
| AIX | `/etc/rc.d/rc2.d/S90pbs` |
| Darwin | `/Library/StartupItems/PBS` |
| HP-UX | `/sbin/init.d/pbs` |
| IRIX | `/etc/init.d/pbs` |
| Linux | `/etc/rc.d/init.d/pbs` |
| OSF1 | `/sbin/init.d/pbs` |
| Solaris | `/etc/init.d/pbs` |
| Tru64 | `/sbin/init.d/pbs` |
| UNICOS | `/etc/config/daemons entry:`<br>`SYS2 pbs YES /opt/pbs/etc/pbs_stop /opt/pbs/etc/pbs_start` |

The PBS startup script reads the `pbs.conf` file to determine which daemons should be started. The startup script can also be run by hand to get status of the PBS daemons, and to start/stop all the PBS daemons on a given host. The command line syntax for the startup script is:

```
STARTUP_SCRIPT [ status | stop | start | restart ]
```

Alternately, you can start the individual PBS daemons manually, as discussed in the following sections. Furthermore, you may wish to change the options specified to various daemons, as discussed below.

**Important:** The method by which the Server and MOMs are shutdown and restarted has different effects on running jobs. Be sure to read section 10.2.6 “Impact of Shutdown / Restart on Running Jobs” on page 170.

### 10.2.1 Manually Starting MOM

MOM should be started at boot time. Typically there are no required options. It is recommended to start MOM before the Server so she will be ready to respond to the Server's "are you there?" ping. Start MOM with the command line:

```
PBS_EXEC/sbin/pbs_mom [options]
```

If MOM is taken down and the host system continues to run, MOM should be restarted with either of the following options:

`-p` This directs MOM to let running jobs continue to run. Because MOM is no longer the parent of the jobs, she will not be notified (**SIGCHLD**) when they die and so must poll to determine when the jobs complete. The resource usage information therefore may not be completely accurate.

`-r` This directs MOM to kill off any jobs which were left running.

Without either the -p or the -r option, MOM will assume the jobs processes no longer exist due to a system restart. This is called a cold start. She will not attempt to kill the processes and will request that any jobs which where running before the system restart be requeued.

Other command line options for MOM include:

`-a alarm` Used to specify the alarm timeout in seconds for computing a resource. Every time a resource request is processed, an alarm is set for the given amount of time. If the request has not completed before the given time, an alarm signal is generated. The default is 5 seconds.

`-C chkdir` Specifies the path of the directory used to hold checkpoint files. The default directory is `PBS_HOME/spool/checkpoint`, see the `-d` option. The directory specified with the `-C` option must be owned by root and accessible (rwx) only by root to protect the security of the checkpoint files. (See also "Site-Specific Job Checkpoint / Restart" on page 109.)

`-c config` Specifies an alternative configuration file, see description below. If this is a relative file name it will be relative to `PBS_HOME/mom_priv`, see the `-d` option. If the specified file cannot be opened, `pbs_mom` will abort. If the `-c` option is not supplied, `pbs_mom` will attempt to open the default configuration file "config" in `PBS_HOME/mom_priv`. If this file is not present, `pbs_mom` will log the fact and continue.

`-d directory` Specifies the path of the directory which is the home of the servers working files, `PBS_HOME`. This option is typically used along with `-M` when debugging MOM. The default directory is given by `PBS_HOME` which is typically `/usr/spool/PBS`.

`-L logfile` Specify an absolute path name for use as the log file. If not specified, MOM will open a file named for the current date in the `PBS_HOME/mom_logs` directory, see the `-d` option.

`-M port` Specifies the port number on which MOM will listen for batch requests. (See also alternate port usage in section 5.3 “Alternate Test Systems” on page 44.)

`-n nice_val` Specifies the priority value of the daemon when it executes.

`-R port` Specifies the port number on which MOM will listen for resource monitor requests, task manager requests and inter-MOM messages. Both a UDP and a TCP port of this number will be used. (See also alternate port usage in section 5.3 “Alternate Test Systems” on page 44.)

`-S port` Specifies the port number on which MOM will communicate with the Server.

`-x` Disables the check for privileged port resource monitor connections. This is used mainly for testing since the privileged port is the only mechanism used to prevent any ordinary user from connecting.

### 10.2.2 Manually Starting the Server

Normally the PBS Server is started from the system boot file via a line such as:

```
PBS_EXEC/sbin/pbs_server [options]
```

The command line options for the Server include:

`-A acctfile` Specifies an absolute path name of the file to use as the accounting file. If not specified, the file is named for the current date in the `PBS_HOME/server_priv/accounting` directory.

`-a active` Specifies if scheduling is active or not. This sets the Server parameter `scheduling`. If the option argument is "true" ("True", "t", "T", or "1"), the server is *active* and the PBS job scheduler will be called. If the argument is "false" ("False", "f", "F", or "0), the server is *idle*, and the scheduler will not be called and no jobs will be run. If this option is not specified, the server will retain the prior value of the `scheduling` parameter.

`-d config` Specifies the path of the directory which is home to the Server's configuration files, `PBS_HOME`. A host may have multiple Servers. Each Server must have a different configuration directory. The default configuration directory is `PBS_HOME` which is typically set to `/usr/spool/PBS`.

`-e mask` Specifies a log event mask to be used when logging. See "log_events" on page 55.

`-F seconds` Specified the delay time (in seconds) from detection of possible Primary Server failure until the Secondary Server takes over.

`-G globus_RPP`

Specifies the port number on which the Server should query the status of PBS MOM Globus daemon.

`-g globus_port`

Specifies the host name and/or port number on which the Server should connect the PBS MOM Globus daemon. The option argument, *globus_port*, is one of the forms: `host_name`, `[:]port_number`, or `host_name:port_number`. If `host_name` not specified, the local host is assumed. If `port_number` is not specified, the default port is assumed.

`-L logfile` Specifies an absolute path name of the file to use as the log file. If not specified, the file is one named for the current date in the

PBS_HOME/server_logs directory, see the -d option.

-M mom_port — Specifies the host name and/or port number on which the server should connect to the MOMs. The option argument, *mom_port*, is one of the forms: host_name, [:]port_number, or host_name:port_number. If host_name not specified, the local host is assumed. If port_number is not specified, the default port is assumed. See the -M option for pbs_mom.

-p port — Specifies the port number on which the Server will listen for batch requests. If multiple Servers are running on a single host, each must have its own unique port number. This option is for use in testing with multiple PBS systems on a single host. (See also alternate port usage in section "Alternate Test Systems" on page 44.)

-R RPPport — Specifies the port number on which the Server should query the status of MOM. See the -R option for pbs_mom.

-S sched_port

Specifies the port number to which the Server should connect when contacting the Scheduler. The option argument, *sched_port*, is of the same syntax as under the -M option.

-t type — Specifies the impact on jobs when the Server restarts. The *type* argument can be one of the following four options:

| Option | Effect Upon Job Running Prior to Server Shutdown |
|---|---|
| cold | All jobs are purged. Positive confirmation is required before this direction is accepted. |
| create | The Server will discard any existing configuration file (including information on nodes, queues and jobs) and re-initialize to the default values. In addition, the server is idled (scheduling set false). Positive confirmation is required before this direction is accepted. |

| Option | Effect Upon Job Running Prior to Server Shutdown |
|---|---|
| `hot` | All jobs in the Running state are retained in that state. Any job that was requeued into the Queued state from the Running state when the server last shut down will be run immediately, assuming the required resources are available. This returns the server to the same state as when it went down. After those jobs are restarted, then normal scheduling takes place for all remaining queued jobs. All other jobs are retained in their current state.<br>If a job cannot be restarted immediately because of a missing resource, such as a node being down, the server will attempt to restart it periodically for up to 5 minutes. After that period, the server will revert to a normal state, as if warm started, and will no longer attempt to restart any remaining jobs which were running prior to the shutdown. |
| `warm`<br>`(default)` | All jobs in the Running state are retained in that state. All other jobs are maintained in their current state. The job scheduler will typically make new selections for which jobs are placed into execution. Warm is the default if `-t` is not specified. |

### 10.2.3 Manually Starting the Scheduler

The Scheduler should also be started at boot time. If starting by hand, use the following command line:

```
PBS_EXEC/sbin/pbs_sched [options]
```

There are no required options for the Standard Scheduler. Available options are listed below.

`-a alarm` This specifies the time in seconds to wait for a schedule run to finish. If a script takes too long to finish, an alarm signal is sent, and the scheduler is restarted. If a core file does not exist in the current directory, `abort`() is called and a core file is generated. The default for *alarm* is 180 seconds.

`-d home` This specifies the PBS home directory, `PBS_HOME`. The current working directory of the scheduler is `PBS_HOME/sched_priv`. If this option is not given, `PBS_HOME` defaults to `PBS_HOME` as defined in the `pbs.conf` file.

`-L logfile` Specifies an absolute path name of the file to use as the log file. If not specified, the scheduler will open a file named for the current date in the `PBS_HOME/sched_logs` directory; see the `-d` option.

`-p file` This specifies the "print" file. Any output from the Scheduler which is written to standard out or standard error will be written to this file. If this option is not given, the file used will be `PBS_HOME/sched_priv/sched_out`. See the `-d` option.

`-R port` This specifies the resource monitor port to use. If this option is not given, the default port for the PBS MOM is used.

`-S port` This specifies the port to use. If this option is not given, the default port for the PBS Scheduler is used. (See also alternate port usage in section 5.3 “Alternate Test Systems” on page 44.)

The options that specify file names may be absolute or relative. If they are relative, their root directory will be `PBS_HOME/sched_priv`.

### 10.2.4 Manually Starting Globus MOM

The optional Globus MOM should be started at boot time if Globus support is desired. Note that the provided startup script does not start the Globus MOM. There are no required options. If starting manually, run it with the line:

```
PBS_EXEC/sbin/pbs_mom_globus [options]
```

If Globus MOM is taken down and the host system continues to run, the Globus MOM should be restarted with the `-r` option. This directs Globus MOM to kill off processes running on behalf of a Globus job. See the **PBS Pro External Reference Specification** (or the `pbs_mom_globus`(1B) manual page) for a more complete explanation.

If the `pbs_mom_globus` daemon is restarted without the `-r` option, the assumption that will be made is that jobs have become disconnected from the Globus gatekeeper due to a system restart (cold start). Consequentially, `pbs_mom_globus` will request that any Globus jobs that were being tracked and which where running be canceled and requeued.

### 10.2.5 Stopping PBS

The `qterm` command is used to shutdown, selectively, or inclusively, the various PBS daemons. The command usage is:

```
qterm [-f | -i | -F] [-m] [-s] [-t type] [server...]
```

The available options, and description of each, follows.

| Option | Description |
|---|---|
| `default`<br>`(no option)` | Used without options, `qterm` will shutdown the Primary Server. |
| `-f` | Specifies that the Secondary Server, in a Server fail-over configuration, should be shut down as well as the Primary Server. If this option is not used in a fail-over configuration, the Secondary Server will become active when the Primary Server exits. The `-f` and `-i` options cannot be used together |
| `-F` | Specifies that the Secondary Server (only) should be shut down. The Primary Server will remain active. The `-F` and `-i` or `-f` options cannot be used together. |
| `-i` | Specifies that the Secondary Server, in a Server fail-over configuration, should return to an idle state and wait for the Primary Server to be restarted. The `-i` and `-f` options cannot be used together. |
| `-m` | Specifies that all known `pbs_mom` daemons should also be told to shut down. This request is relayed by the Server to each MOM. |
| `-s` | Specifies that the Job Scheduler, `pbs_sched`, should also be terminated. |
| `-t type` | Specifies the type of shutdown. |

The `-t type` option takes one of three arguments for the *type* of shutdown you wish to perform. The three types are:

| *type* of shutdown | Description |
|---|---|
| `immediate` | All running jobs are to immediately stop execution. If checkpoint is supported, running jobs that can be checkpointed are checkpointed, terminated, and requeued. If checkpoint is not supported or the job cannot be checkpointed, running jobs are requeued if the rerunable attribute is true. Otherwise, jobs are killed. Normally the Server will not shutdown until there are no jobs in the running state. If the Server is unable to contact the MOM of running job, the job is still listed as running. The Server may be forced down by a second "`qterm -t immediate`" command. |
| `delay` | If checkpoint is supported, running jobs that can be checkpointed are checkpointed, terminated, and requeued. If a job cannot be checkpointed, but can be rerun, the job is terminated and requeued. Otherwise, running jobs are allowed to continue to run. Note, the operator or Administrator may use the `qrerun` and `qdel` commands to remove running jobs. |
| `quick` | This is the default action if the `-t` option is not specified. This option is used when you wish that running jobs be left running when the Server shuts down. The Server will cleanly shutdown and can be restarted when desired. Upon restart of the Server, jobs that continue to run are shown as running; jobs that terminated during the Server's absence will be placed into the exiting state. |

If you are not running in Server Failover mode, then the following command will shut down the entire PBS complex:

```
# qterm -s -m
```

However, if Server Failover is enabled, the above command will result in the Secondary Server becoming active after the Primary has shut down. Therefore, in a Server Failover configuration, the "`-f`" (or the "`-i`") option should be added:

```
# qterm -s -m -f
```

**Important:** Note that `qterm` defaults to `qterm -t quick.` Also, note that the Server does a quick shutdown upon receiving `SIGTERM`.

**Important:** Should you ever have the need to stop a MOM daemon but leave jobs managed by her running, the only way to do this is to kill MOM with a `SIGKILL` (`-9`).

### 10.2.6 Impact of Shutdown / Restart on Running Jobs

The method of how PBS is shut down (and which components are stopped) will effect running jobs differently. The impact of a shutdown (and subsequent restart) on running jobs depends on three things:

1 How the Server (`pbs_server`) is shut down,
2 How MOM (`pbs_mom`) is shut down,
3 How MOM is restarted.

Choose one of the following recommended sequences, based on the desired impact on jobs, to start and restart PBS:

1. To allow running jobs to continue to run:

Shutdown:

```
qterm -t quick -m -s
```

Restart:

```
pbs_server -t warm
pbs_mom -p
pbs_sched
```

2. To requeue the jobs and immediately rerun the ones that were running:

Shutdown:

```
qterm -t immediate -m -s
```

Restart:

```
pbs_mom
pbs_server -t hot
pbs_sched
```

3. To requeue the jobs and ignore what was running before:

Shutdown:

```
qterm -t immediate -m -s
```

Restart:

```
pbs_mom
pbs_server -t warm
pbs_sched
```

### 10.2.7 Stopping / Restarting a Single MOM

If you wish to shutdown and restart a single MOM, be aware of the following effects on jobs.
Methods of manual shutdown of a single MOM:

| | |
|---|---|
| SIGTERM | If a MOM is killed with the signal SIGTERM, jobs are killed before MOM exits. Notification of the terminated jobs is not sent to the Server until the MOM is restarted. Jobs will still appear to be in the “R” (running) state. |
| SIGINT<br>SIGKILL | If a MOM is killed with either of these signals, jobs are not killed before the MOM exits. With SIGINT, MOM exits after cleanly closing network connections. |

A MOM may be restarted with the following options:

| | |
|---|---|
| pbs_mom | Running jobs are returned to the Server to be requeued or deleted (if not rerunnable). Processes associated with the job are not killed (signaled). |
| pbs_mom -r | Processes associated with the job are killed. Running jobs are returned to the Server to be requeued or deleted. This option should not be used if the system has just been rebooted as the process numbers will be incorrect and a process not related to the job would be killed. |
| pbs_mom -p | Jobs which were running when MOM terminated remain running. |

## 10.3 Starting and Stopping PBS: Windows 2000 / XP

When PBS Pro is installed on either Microsoft Windows XP or 2000, the PBS processes are registered as system services. As such, they will be automatically started and stopped when the systems boots and shuts down. However, there may come a time with you need to manually stop or restart the PBS services (such as shutting them down prior to a PBS software upgrade). The following example illustrates how to manually stop and restart the PBS services. (These lines must be typed at a Command Prompt with Administrator prilevege.)

```
net stop pbs_sched
net stop pbs_mom
net stop pbs_server
net stop pbs_rshd
#     and to restart PBS:
net start pbs_server
net start pbs_mom
net start pbs_sched
net start pbs_rshd
```

It is possible to run (Administrator privilege) the PBS services manually, in standalone mode, as follows:

```
Admin> pbs_server -N <options>
Admin> pbs_mom -N <options>
Admin> pbs_sched -N <options>
Admin> pbs_rshd -N <options>
```

Note, however, that `pbs_mom` run in this manner will not have all the needed privileges as when run under the service control manager using a SYSTEM account.

### 10.3.1 Startup Options to PBS Windows Services

The procedure to specify startup options to the PBS Windows Services is as follows:

1. Go to `Start Menu->Control Panel->Administrative Tools->Services` (in Win2000) or `Start Menu->Control Panel->Performance and Maintenance->AdministrativeTools-> Services` (in WinXP).

2. Select the PBS Service you which to alter. For example, if you select "PBS_MOM", the MOM service dialog box will come up.

3. Enter in the "Start parameters" entry line as required. For example, to specify an alternative MOM configuration file, you might specify the following input:

```
-c "\Program Files\PBS Pro\home\mom_priv\config2"
```

4. Lastly, click on "Start" to start the specified Service.

Keep in mind that the windows services dialog does not remember the "Start parameters" value when you close the dialog. For future restarts, you need to always specify the "Start parameters" value.

## 10.4 UNIX and Windows Interoperability Issues

This section discusses some of the issues related to mixing UNIX (or Linux) systems with Windows 2000/XP systems in the same PBS Pro cluster.

### 10.4.1 User Name Length Issues

Windows and UNIX systems have different maximum user login name lengths. In particular, most UNIX systems will not support a login name that is longer than 8 or 10 characters. Therefore, if your Windows usernames are longer than permitted by UNIX, either you cannot combine the systems in the same cluster, or you will need to shorten the Windows names.

## 10.5 Checkpoint / Restart Under PBS

PBS Pro supports two methods of checkpoint / restart: OS-specific and a generic site-specific method. Operating system checkpoint-restart is supported where provided by the system. Currently both SGI IRIX and Cray UNICOS provide OS-level checkpoint packages, which PBS uses. Alternatively, a site may configure the generic checkpointing feature of PBS Pro to use any method of checkpoint / restart. For details see “Site-Specific Job Checkpoint / Restart” on page 109. (In addition, user’s may manage their own checkpointing from within their application. This is discussed further in the **PBS Pro User Guide**.)

The location of the directory into which jobs are checkpointed can now be specified in a number of ways. In order of preference:

1. "`-C path`" command line option to `pbs_mom`
2. **`PBS_CHECKPOINT_PATH`** environment variable
3. "`$checkpoint_path path`" option in MOM's config file
4. default value

### 10.5.1 Manually Checkpointing a Job

On systems which provide OS-level checkpointing, the PBS Administrator may manually force a running job to be checkpointed. This is done by using the `qhold` command. (Discussed in detail in the **PBS Pro Users Guide**).

### 10.5.2 Checkpointing Jobs During PBS Shutdown

As installed, the PBS startup script will not result in PBS checkpointing jobs (on systems which provide OS-level checkpointing). This behavior allows for a faster shut down of the batch system at the expense of rerunning jobs from the beginning. If you prefer jobs to be checkpointed, then append the `-t immediate` option to the `qterm` statement in the script.

### 10.5.3 Suspending/Checkpointing Multi-node Jobs

The PBS suspend/resume and checkpoint/restart capabilities is supported for multi-node jobs. With checkpoint (on systems which provide OS-level checkpointing), the system must be able to save the complete session state in a file. This means any open socket will cause the operation to fail. PBS normally sets up a socket connection to a process (`pbs_demux`) which collects stdio streams from all tasks. If this is not turned off, the checkpoint cannot work. Therefore, a new job attribute has been added: `no_stdio_sockets`. See the `pbs_job_attributes` manual page for more details. If this attribute is true, the `pbs_demux` process will not be started and no open socket will prevent the checkpoint from working. The other place where PBS will use a socket that must be addressed is if the program `pbsdsh` is used to spawn tasks. There is a new option for `pbsdsh` '`-o`' that is used to prevent it from waiting for the spawned tasks to finish. This is done so no socket will be left open to the MOM to receive task manager events. If this is used, the shell must use some other method to wait for the tasks to finish.

### 10.5.4 Checkpointing Jobs Prior to SGI IRIX Upgrade

Under the SGI IRIX operating system, the normal checkpoint procedure does not save shared libraries in the restart image in order to reduce the image size and time required to write it. This type of image cannot be restarted following a IRIX operating system upgrade. In order to produce an image which can be restarted following an upgrade, a special flag is required when calling checkpoint. MOM has a `config` file option `$checkpoint_upgrade` which if present causes PBS to use the special upgrade checkpoint flag. It is recommended that this flag be set (and `pbs_mom` be restarted via `SIGHUP`) only when shutting down PBS just prior to upgrading your system.

## 10.6 Security

There are three parts to security in the PBS system:

| | |
|---|---|
| **Internal security** | Can the daemons be trusted? |
| **Authentication** | How do we believe a client about who it is. |
| **Authorization** | Is the client entitled to have the requested action performed. |

### 10.6.1 Internal Security

A significant effort has been made to insure the various PBS daemon themselves cannot be a target of opportunity in an attack on the system. The two major parts of this effort is the security of files used by the daemons and the security of the daemons environment. Any file used by PBS, especially files that specify configuration or other programs to be run, must be secure. The files must be owned by root and in general cannot be writable by anyone other than root.

A corrupted environment is another source of attack on a system. To prevent this type of attack, each daemon resets its environment when it starts. If it does not already exists, this file is created during the install process. As built by the install process, it will contain a very basic path and, if found in root's environment, the following variables: **TZ**, **LANG**, **LC_ALL**, **LC_COLLATE**, **LC_CTYPE**, **LC_MONETARY**, **LC_NUMERIC**, and **LC_TIME**. The `environment` file may be edited to include the other variables required on your system.

**Important:** Please note that **PATH** must be included. This value of **PATH** will be passed on to batch jobs. To maintain security, it is important that **PATH** be restricted to known, safe directories. Do NOT include "." in **PATH**. Another variable which can be dangerous and should not be set is **IFS**.

The entries in the **PBS_ENVIRONMENT** file can take two possible forms:

```
variable_name=value
variable_name
```

In the later case, the value for the variable is obtained before the environment is reset.

### 10.6.2 Host Authentication

PBS uses a combination of information to authenticate a host. If a request is made from a client whose socket is bound to a privileged port (less than 1024, which requires root privilege), PBS believes the IP (Internet Protocol) network layer as to whom the host is. If the client request is from a non-privileged port, the name of the host which is making a client request must be included in the credential sent with the request and it must match the IP network layer opinion as to the host's identity.

### 10.6.3 Host Authorization

Access to the Server from another system may be controlled by an access control list (ACL). Access to pbs_mom is controlled through a list of hosts specified in the pbs_mom's configuration file. By default, only "localhost" and the name returned by gethostname(2) are allowed. See the man pages pbs_mom(8B) for more information on the configuration file. Access to pbs_sched is not limited other than it must be from a privileged port.

### 10.6.4 User Authentication

The PBS Server authenticates the user name included in a request using the supplied PBS credential. This credential is supplied by pbs_iff.

### 10.6.5 User Authorization

PBS as shipped does not assume a consistent user name space within the set of systems which make up a PBS cluster. However, the Administrator can enable this assumption, if desired. By default, the routine site_map_user() is called twice, once to map the name of the requester and again to map the job owner to a name on the Server's (local) system. If the two mappings agree, the requester is considered the job owner.

If running PBS in an environment that *does* have a flat user namespace, the Administrator can disable these checks by setting the flatuid Server parameter to True via qmgr:

```
# qmgr
Qmgr: set server flatuid=True
```

If flatuid is set to true, a UserA on HostX who submits a job to the PBS Server on HostY will *not* require an entry in the /etc/passwd file, nor a .rhosts entry on HostY for HostX, nor must HostX appear in HostY's /etc/hosts.equiv file. In either case, if a job is submitted by *UserA@HostA* PBS will allow the job to be deleted or altered by *UserA@HostB*

**Important:** When flatuid is true, a password entry for the submitting user is not required on the server system. But, without the password entry, the Server does not know the user's group (it is not passed implicitly from qsub). Without knowing the group, the acl_group parameter on a queue cannot be checked. The error of "not allowed" will be returned. This applies to acl_group on the Server or Queue level.

If flatuid is NOT set to true, a user may supply a name under which the job is to be executed on a certain system. If one is not supplied, the name of the job owner is chosen to be the execution name see the -u user_list option of the qsub(1B) command. Authorization to execute the job under the chosen name is granted under the following conditions:

1. The job was submitted on the Server’s (local) host and the submitter’s name is the same as the selected execution name.

2. The host from which the job was submitted is declared trusted by the execution host in the system hosts.equiv file or the submitting host and submitting user’s name are listed in the execution users’ .rhosts file. The system supplied library function, ruserok(), is used to make these checks.

   The hosts.equiv file is located in /etc under UNIX, and in %WINDIR%\system32\drivers\etc\ under Windows).

In addition to the above checks, access to a PBS Server and queues within that Server may be controlled by access control lists. (For details see “Server Configuration Parameters” on page 52 and “Queue Configuration Parameters” on page 62.)

### 10.6.6 Group Authorization

PBS allows a user to submit jobs and specify under which group the job should be executed. The user specifies a `group_list` attribute for the job which contains a list of group@host similar to the user list. See the `group_list` attribute under the -W option of `qsub`(1B). The PBS Server will ensure the user is a member of the specified group by:

1. Checking if the specified group is the user's primary group in the password entry on the execution host. In this case the user's name does not have to appear in the group entry for his primary group.

2. Checking on the execution host for the user's name in the specified group entry in `/etc/group` (under UNIX) or in the group membership field of the user's account profile (under Windows).

The job will be aborted if both checks fail. The checks are skipped if the user does not supply a group list attribute. In this case the user's primary group from the password file will be used.

When staging files in or out, PBS also uses the selected execution group for the copy operation. This provides normal UNIX access security to the files. Since all group information is passed as a string of characters, PBS cannot determine if a numeric string is intended to be a group name or GID. Therefore when a group list is specified by the user, PBS places one requirement on the groups within a system: each and every group in which a user might execute a job MUST have a group name and an entry in `/etc/group`. If no `group_list` are ever used, PBS will use the login group and will accept it even if the group is not listed in `/etc/group`. Note, in this latter case, the `egroup` attribute value is a numeric string representing the GID rather than the group "name".

## 10.7 External Security

In addition to the security measures discussed above, PBS provides three levels of privilege: user, Operator, and Manager. Users have user privilege which allows them to manipulate their own jobs. Manager or Operator privilege is required to set or unset attributes of the Server, queues, nodes, and to act on other peoples jobs. For specific limitations on "user" privilege, and additional parameters available to Managers and Operators, review the following: "The qmgr Command" on page 47; the introduction to Chapter 11 "Administrator Commands" on page 195; and the discussion of user commands in the **PBS Pro User Guide**.

## 10.8 Root Owned Jobs

The Server will reject any job which would execute under the UID of zero unless the owner of the job, typically root on this or some other system, is listed in the Server parameter `acl_roots`.

Under Windows, a root account is any account that is a member of the “Administrators” group. Also, the special LOCAL SYSTEM account is considered to be a root account.

## 10.9 Managing PBS and Multi-node Parallel Jobs

Many customers use PBS Pro in cluster configurations for the purpose of managing multi-node parallel applications. This sections provides the PBS Administrator with information specific to this situation.

### 10.9.1 Interfacing MPICH with PBS Pro

If you are running an open source version of MPI, such as MPICH, then the `mpirun` command can be modified to check for the PBS environment and use the PBS supplied host file. In the case of MPICH, this is easily done by editing the `.../mpich/bin/mpirun.args` file and adding the following near line 40 (depending on the version being used):

```
if [ "$PBS_NODEFILE" != "" ]
then
     machineFile=$PBS_NODEFILE
fi
```

**Important:** Additional information regarding checkpointing of parallel jobs is given in “Suspending/Checkpointing Multi-node Jobs” on page 174.

## 10.10 SGI Job Container / Limits Support

PBS Pro support the SGI Job Container/Limit feature. Each PBS job is placed in its own

SGI Job container. Limits on the job are set as the `MIN(ULDB limit, PBS resource_list limit)`. The ULDB domains are set in the following order:

> PBS_{queue name}
> PBS
> batch

Limits are set for the following resources: "cput" and "vmem". A job limit is NOT set for "mem" because the kernel does not factor in shared memory segments among `sproc`() processes, thus the system reported usage is too high.

## 10.11 Job Prologue / Epilogue Programs

PBS provides the ability for the Administrator run a site supplied script (or program) before (`prologue`) and/or after (`epilogue`) each job runs. This provides the capability to perform initialization or cleanup of resources, such as temporary directories or scratch files. The scripts may also be used to write "banners" on the job's output files. When multiple nodes are allocated to a job, these scripts are run only by the "Mother Superior", the `pbs_mom` on the first node allocated. This is also where the job shell script is run. Note that both the prologue and epilogue are run with root privilege (not as the user), and neither are included in the job session, thus the `prologue` cannot be used to modify the job environment or change limits on the job.

If a `prologue` or `epilogue` script is not present, MOM continues in a normal manner. If present, the script is run with root privilege. In order to be run, the script must adhere to the following rules:

- The script must be in the *`PBS_HOME`*`/mom_priv` directory with the name `prologue` (under UNIX) or `prologue.bat` (under Windows) for the script to be run before the job and the name `epilogue` (under UNIX) or `epilogue.bat` (under Windows) for the script to be run after the job.
- The script must be owned by root (Administrator).
- The script must be readable and executable by root (Administrator).
- The script cannot be writable by anyone but root (Administrator).

The "script" may be either a shell script or an executable object file. Typically, a shell script should start with a line of the form:

```
#!/path/interpreter
```

For more information, see the system documentation for `execve`(2) or `exec`(2).

The `prologue` will be run immediately prior to executing the job. When that execution completes for any reason, (normal termination, job deleted while running, error exit, or even if `pbs_mom` detects an error and cannot completely start the job), the `epilogue` script will be run. If the job is deleted while it is "queued", then neither the `prologue` nor the `epilogue` is run.

If a job is rerun or requeued as the result of being checkpointed, the exit status passed to the `epilogue` (and recorded in the accounting record) will have a one of the following special values:

-11 - Job was rerun
-12 - Job was checkpointed and aborted

### 10.11.1 Prologue and Epilogue Arguments

When invoked, the `prologue` is called with the following arguments:

| | |
|---|---|
| argv[1] | the job id. |
| argv[2] | the user name under which the job executes. |
| argv[3] | the group name under which the job executes. |

The `epilogue` is called with the above, plus:

| | |
|---|---|
| argv[4] | the job name. |
| argv[5] | the session id. |
| argv[6] | the requested resource limits (list). |
| argv[7] | the list of resources used |
| argv[8] | the name of the queue in which the job resides. |
| argv[9] | the account string, if one exists. |
| argv[10] | the exit status of the job. |

For both the `prologue` and `epilogue`:

| | |
|---|---|
| envp | The environment passed to the script is null. |

cwd The current working directory is `PBS_HOME/mom_priv` (prologue) or the user's home directory (epilogue).

input When invoked, both scripts have standard input connected to a system dependent file. Currently, this file is `/dev/null`.

output With one exception, the standard output and standard error of the scripts are connected to the files which contain the standard output and error of the job. If a job is an interactive PBS job, the standard output and error of the epilogue is pointed to `/dev/null` because the pseudo terminal connection used was released by the system when the job terminated.

### 10.11.2 Prologue and Epilogue Time Out

To prevent an error condition within the `prologue` or `epilogue` from delaying PBS, MOM places an alarm around the script's/program's execution. This is currently set to 30 seconds. If the alarm sounds before the scripts has terminated, MOM will kill the script. The alarm value can be changed via `$prologalarm` MOM configuration parameter.

### 10.11.3 Prologue and Epilogue Error Processing

Normally, the `prologue` and `epilogue` programs should exit with a zero exit status. MOM will record in her log any case of a non-zero exit codes. Exit status values and their impact on the job are:

| Exit Code | Meaning | Prologue | Epilogue |
|---|---|---|---|
| -4 | The script timed out (took too long). | The job will be requeued. | Ignored |
| -3 | The `wait`(2) call waiting for the script to exit returned with an error. | The job will be requeued | Ignored |
| -2 | The input file to be passed to the script could not be opened. | The job will be requeued. | Ignored |
| -1 | The script has a permission error, is not owned by root, and/or is writable by others than root. | The job will be requeued. | Ignored |
| 0 | The script was successful. | The job will run. | Ignored |

| Exit Code | Meaning | Prologue | Epilogue |
|---|---|---|---|
| 1 | The script returned an exit value of 1. | The job will be aborted. | Ignored |
| >1 | The script returned a value greater than one, | The job will be requeued. | Ignored |

The above apply to normal batch jobs. Interactive-batch jobs (`-I` option) cannot be requeued on a non-zero status. The network connection back to `qsub` is lost and cannot be re-established. Interactive jobs will be aborted on any non-zero `prologue` exit.

**Important:** The Administrator must exercise great caution in setting up the `prologue` to prevent jobs from being flushed from the system.

`Epilogue` script exit values which are non-zero are logged, but have no impact on the state of the job.

## 10.12 Use and Maintenance of Logfiles

The PBS system tends to produce a large number of logfile entries. There are two types of logfiles: the event logs which record events from each PBS daemon (`pbs_server`, `pbs_mom`, and `pbs_sched`) and the PBS accounting log.

### 10.12.1 PBS Events

The amount of output in the PBS event logfiles depends on the specified log filters for each daemon. All three PBS daemons can be directed to record only messages pertaining to certain event types. The specified events are logically "or-ed" to produce a mask representing the events the local site wishes to have logged. The available events, and corresponding decimal and hexidecimal values are show below.

| Value | Hex | Event Description |
|---|---|---|
| 1 | 0x1 | Internal PBS Errors. |
| 2 | 0x2 | System (OS) Errors, such as malloc failure. |
| 4 | 0x4 | Administrator related events, such as changing queue parameters. |

| Value | Hex | Event Description |
|---|---|---|
| 8 | 0x8 | Job related events: submitted, ran, deleted, ... |
| 16 | 0x10 | Job Resource Usage. |
| 32 | 0x20 | Security related events, such as attempts to connect from an unknown host. |
| 64 | 0x40 | When the scheduler was called and why. |
| 128 | 0x80 | First level, common, debug messages. |
| 256 | 0x100 | Second level, more rare, debug messages. |

Everything turned on is of course 511. 127 is a good value to use. The event logging mask is controlled differently for the different daemons. The following table shows the log event parameter for each daemon, and page reference for details.

| Daemon | Attribute and Reference | Notes |
|---|---|---|
| Server | See “log_events” on page 55. | Takes effect immediately with `qmgr` |
| MOM | See “$logevent” on page 101. | Requires `SIGHUP` to MOM |
| Scheduler | See “log_filter” on page 131. | Requires `SIGHUP` to Scheduler |

### 10.12.2 The Event Logfiles

Each PBS daemon maintains separate event logfiles. The logfiles default to a file with the current date as the name in the `PBS_HOME/(daemon)_logs` directory. This location can be overridden with the "`-L pathname`" option where pathname must be an absolute path.

If the default logfile name is used (no `-L` option), the log will be closed and reopened with the current date daily. This happens on the first message after midnight. If a path is given with the `-L` option, the automatic close/reopen does not take place. All daemons will close and reopen the same named log file on receipt of **SIGHUP**. The process identifier (PID) of the daemon is available in its lock file in its home directory. Thus it is possible to move the current log file to a new name and send **SIGHUP** to restart the file thusly:

```
# cd /usr/spool/PBS/DAEMON_logs
# mv current archive
# kill -HUP ‘cat ../DAEMON_priv/daemon.lock‘
#
```

### 10.12.3 Event Logfile Format

The daemon event logfile is a text file with each entry terminated by a new line. The format of an entry is:

```
date-time;event_code;server_name;object_type;object_name;message
```

The `date-time` field is a date and time stamp in the format:

`mm/dd/yyyy hh:mm:ss`.

The `event_code` is the type of event which triggered the event logging. It corresponding to the bit position, 0 to n, in the event mask (discussed above) of the daemon writing the event record.

The `server_name` is the name of the Server which logged the message. This is recorded in case a site wishes to merge and sort the various logs in a single file.

The `object_type` is the type of object which the message is about:

| | |
|---|---|
| `Svr` | for server |
| `Que` | for queue |
| `Job` | for job |
| `Req` | for request |
| `Fil` | for file. |

The `object_name` is the name of the specific object. `message_text` field is the text of the log message.

When reading the PBS event logfiles, you may see messages of the form “Type 19 request received from PBS_Server...”. These “type codes” correspond to different PBS batch requests. Appendix B contains a listing of all “types” and each corresponding batch request.

### 10.12.4 The Accounting Log

The PBS Server daemon maintains an accounting log. The log name defaults to *PBS_HOME/*server_priv/accounting/*ccyymmdd* where *ccyymmdd* is the date. The accounting log files may be placed elsewhere by specifying the -A option on the pbs_server command line. The option argument is the full (absolute) path name of the file to be used. If a null string is given, then the accounting log will not be opened and no accounting records will be recorded. For example

```
# pbs_server -A ""
#
```

The accounting file is changed according to the same rules as the log files. If the default file is used, named for the date, the file will be closed and a new one opened every day on the first event (write to the file) after midnight. With either the default file or a file named with the -A option, the Server will close the accounting log and reopen it upon the receipt of a **SIGHUP** signal. This allows you to rename the old log and start recording again on an empty file. For example, if the current date is February 9, 2001 the Server will be writing in the file 20010209. The following actions will cause the current accounting file to be renamed feb9 and the Server to close the file and starting writing a new 20010209.

```
# mv 20010209 feb9
# kill -HUP 1234     (the Server’s pid)
#
```

### 10.12.5 Accounting Log Format

The PBS accounting file is a text file with each entry terminated by a new line. The format of an entry is:

```
date time;record_type;id_string;message_text
```

The date time field is a date and time stamp in the format:

```
mm/dd/yyyy hh:mm:ss
```

The id_string is the job, reservation, or reservation-job identifier. The message_text is ascii text. The content depends on the record type. The message text format is blank separated keyword=value fields. The record_type is a single character indicating the type of record. The types are:

A Job was aborted by the server.

B Beginning of reservation period. The message_text field contains details about the specified advance reservation. Possible attributes include:

| Attribute | Explanation |
|---|---|
| owner=ownername | Name of party who submitted the resources reservation request. |
| name=reservation_name | If submitter supplied a name string for the reservation. |
| account=account_string | If submitter supplied a to be recorded in accounting. |
| queue=queue_name | The name of the instantiated reservation queue if this is a general resources reservation. If the resources reservation is for a reservation job, this is the name of the queue to which the reservation-job belongs. |
| ctime=creation_time | Time at which the resources reservation got created, seconds since the epoch. |
| start=period_start | Time at which the reservation period is to start, in seconds since the epoch. |
| end=period_end | Time at which the reservation period is to end, seconds since the epoch. |
| duration=reservation_duration | The duration specified or computed for the resources reservation, in seconds. |
| nodes=reserved_nodes | If nodes with specified properties are required, this string is the allocated set. |
| authorized_users=users_list | The list of acl_users on the queue that is instantiated to service the reservation. |

| Attribute | Explanation |
|---|---|
| `authorized_groups=`groups_list | If specified, the list of acl_groups on the queue that is instantiated to service the reservation. |
| `authorized_hosts=`hosts_list | If specified, the list of acl_hosts on the queue that is instantiated to service the reservation. |
| `resource_list=`resources_list | List of resources requested by the reservation. Resources are listed individually as, for example: resource_list.ncpus=16 resource_list.mem=1048676. |

`C` Job was checkpointed and held.

`D` Job was deleted by request. The `message_text` will contain `requestor=user@host` to identify who deleted the job.

`E` Job ended (terminated execution). The `message_text` field contains detailed information about the job. Possible attributes include:

| Attribute | Explanation |
|---|---|
| `user=`username | The user name under which the job executed. |
| `group=`groupname | The group name under which the job executed. |
| `account`=account_string | If job has an "account name" string. |
| `jobname=`job_name | The name of the job. |
| `queue=`queue_name | The name of the queue in which the job executed. |
| `resvname`=reservation_name | The name of the resource reservation, if applicable. |
| `resvID`=reservation_ID_string | The ID of the resource reservation, if applicable. |

| Attribute | Explanation |
|---|---|
| `resvjobID=`reservation_ID_string | The ID of the resource reservation, if applicable. |
| `ctime=`time | Time in seconds when job was created (first submitted). |
| `qtime=`time | Time in seconds when job was queued into current queue. |
| `etime=`time | Time in seconds when job became eligible to run. |
| `start=`time | Time in seconds when job execution started. |
| `exec_host=`host | Name of host on which the job is being executed. |
| `Resource_List.`resource=limit | List of the specified resource limits. |
| `session=`sessionID | Session number of job. |
| `alt_id=`id | Optional alternate job identifier. Included only for certain systems: IRIX 6.x with Array Services - The alternate id is the Array Session Handle (ASH) assigned to the job. For irix6cpuset mom, the alternate id holds the name of the cpuset assigned to the job as well as resources assigned to the job. For example, alt_id=cpuset=357.sgi3:1024kb/1p |
| `end=`time | Time in seconds when job ended execution. |
| `Exit_status=`value | The exit status of the job. If the value is less than 10000 (decimal) it is the exit value of the top level process of the job, typically the shell. If the value is greater than 10000, the top process exited on a signal whose number is given by subtracting 10000 from the exit value. |
| `Resources_used.resource=`<br>usage_amount | Amount of specified resource used over the duration of the job. |

F Resources reservation period finished.

K Scheduler or server requested removal of the reservation. The message_text field contains: requestor=user@host to identify who deleted the resources reservation.

k Resources reservation terminated by ordinary client - e.g. an owner issuing a pbs_rdel command. The message_text field contains: requestor=user@host to identify who deleted the resources reservation.

Q Job entered a queue. The message_text contains queue=name identifying the queue into which the job was placed. There will be a new Q record each time the job is routed or moved to a new (or the same) queue.

R Job was rerun.

S Job execution started. The message_text field contains:

| Attribute | Explanation |
|---|---|
| user=username | The user name under which the job will execute. |
| group=groupname | The group name under which the job will execute. |
| jobname=job_name | The name of the job. |
| queue=queue_name | The name of the queue in which the job resides. |
| ctime=time | Time in seconds when job was created (first submitted). |
| qtime=time | Time in seconds when job was queued into current queue. |
| etime=time | Time in seconds when job became eligible to run; no holds, etc. |
| start=time | Time in seconds when job execution started. |
| exec_host=host | Name of host on which the job is being executed. |
| Resource_List.resource=limit | List of the specified resource limits. |

| Attribute | Explanation |
|---|---|
| session=sessionID | Session number of job. |

T Job was restarted from a checkpoint file.

U Created unconfirmed resources reservation on Server. The message_text field contains requestor=user@host to identify who requested the resources reservation.

Y Resources reservation confirmed by the Scheduler. The message_text field contains the same item (items) as in a U record type.

For Resource_List and Resources_used, there is one entry per resource.

## 10.13 Using the syslog facility with PBS Pro

Traditionally, each PBS daemon logged various levels of information about events in its own log file. While having the advantage of a concise location for the information from a daemon, the disadvantage is that in a cluster, the logged information is scattered across each execution host.

If your site uses the syslog subsystem, PBS may be configured to make full use of it. The following entries in pbs.conf control the use of syslog by the PBS daemons:

| | |
|---|---|
| PBS_CONF_SYSLOG=x | This variable controls the use of syslog and syslog "facility" under which the entries are logged. If x is:<br>0 - no syslogging<br>1 - logged via LOG_DAEMON facility<br>2 - logged via LOG_LOCAL0 facility<br>3 - logged via LOG_LOCAL1 facility<br>...<br>9 - logged via LOG_LOCAL7 facility |

| | |
|---|---|
| PBS_CONF_SYSLOGSEVR=y | This variable controls the severity level of messages that are logged, See /usr/include/sys/syslog.h. If y is:<br>0 - only LOG_EMERG messages are logged<br>1 - messages up to LOG_ALERT are logged<br>...<br>7 - messages up to LOG_DEBUG are logged |

**Important:** PBS_CONF_SYSLOGSEVR is used in addition to PBS's log_event mask which controls the class of events (job, node, ...) that are logged.

## 10.14 Interpreting PBS Exit Codes

The PBS Server logs and accounting logs record an “exit status” of jobs. Zero or positive exit status is the status of the top level shell. Certain negative exit status are used internally and will never be reported to the user. The positive exit status values indicate which signal killed the job. Depending on the system, values greater than 128 (or on some systems 256, see wait(2) or waitpid(2) for more information) are the value of the signal that killed the job. To interpret (or “decode”) the signal contained in the exit status value, subtract the base value from the exit status. For example, if a job had an exit status of 143, that indicates the jobs was killed via a SIGTERM (e.g. 143 - 128 = 15, signal 15 is SIGTERM). See the kill(1) manual page for a mapping of signal numbers to signal name on your operating system.

## 10.15 Shell Invocation

When PBS starts a job, it invokes the user’s login shell (unless the user submitted the job with the -S option). PBS passes the job script which is a shell script to the login process in one of two ways depending on how PBS was installed.

**1.** Name of Script on Standard Input
The default method is to pass the name of the job script to the shell program. This is equivalent to typing the script name as a command to an interactive shell. Since this is the only line passed to the script, standard input will be empty to any commands. This approach offers both advantages and disadvantages:

+ Any command which reads from standard input without redirection will get an EOF.

+ The shell syntax can vary from script to script, it does not have to match the syntax for the user's login shell. The first line of the script, even before any #PBS directives, should be

`#!/`*`shell`* where *`shell`* is the full path to the shell of choice, `/bin/sh`, `/bin/csh`, ...

The login shell will interpret the `#!` line and invoke that shell to process the script.

- An extra shell process is run to process the job script.

- If the script does not include a `#!` line as the first line, the wrong shell may attempt to interpret the script producing syntax errors.

- If a non-standard shell is used via the -S option, it will not receive the script, but its name, on its standard input.

**2.** Script as Standard Input
The alternative method for PBS, is to open the script file as standard input for the shell. This is equivalent to typing:

```
% /path/shell < script
```

This also offers advantages and disadvantages:

+ The user's script will always be directly processed by the user's login shell.

+ If the user specifies a non-standard shell (any old program) with the -S option, the script can be read by that program as its input.

- If a command within the job script reads from standard input, it may read lines from the script depending on how far ahead the shell has buffered its input. Any command line so read will not be executed by the shell. A command that reads from standard input with out explicit redirection is generally unwise in a batch job.

The choice of shell invocation methods is left to the site. It is recommended that all PBS execution servers (pbs_mom) within that site be built to use the same shell invocation method.

## 10.16 TCL/tk Interface to PBS API

The PBS Pro software includes a published and documented Application Programming Interface (API). Its use is discussed in detail in the **PBS Pro External Reference Specification** (**ERS**). There is also a TCL/tk interface to PBS. Wrapped versions of the external API calls are compiled into a special version of the TCL shell, called `pbs_tclsh`. (A special version of the tk window shell is also provided, called `pbs_wish`.). These tools were originally developed as part of a TCL scheduler interface, and as such are documented in the `pbs_tclapi(3B)` manual page.

# Chapter 11
# Administrator Commands

There are two types of commands in PBS: those that users use to manipulate their own jobs, and those that the PBS Administrator uses to manage the PBS system. This chapter covers the various PBS administrator commands.

The table below lists all the PBS commands; the left column identities all the user commands, and the right column identifies all the administrator commands. (The user commands are described in detail in the **PBS Pro User Guide**.)

**Important:** Individuals with PBS Operator or Manager privilege can use the user commands to act on any user job. For example, a PBS Operator can delete or move any user job. (Detailed discussion of privilege within PBS is discussed under the heading of section 10.7 “External Security” on page 178.)

Some of the PBS commands are intended to be used by the PBS Operator or Manager. These are the administrator commands, which are described in detail in this chapter. Some administrator commands can be executed by normal users but with limited results. The `qmgr` command can be run by a normal user, who can view but cannot alter any Server configuration information. If you want normal users to be able to run the `tracejob` and `pbs-report` commands, you can add read access to the `server_priv/accounting` directory, enabling the commands to report job-specific information. But note that all job information will then be available to all users.

**Table 1: PBS Pro User and Administrator Commands**

| User Commands | | Administrator Commands | |
|---|---|---|---|
| **Command** | **Purpose** | **Command** | **Purpose** |
| `nqs2pbs` | Convert from NQS | `pbs-report` | Report job statistics |
| `pbs_rdel` | Delete Adv. Reservation | `pbs_hostid` | Report host identifier |
| `pbs_rstat` | Status Adv. Reservation | `pbs_hostn` | Report host name(s) |
| `pbs_rsub` | Submit Adv.Reservation | `pbs_probe` | Report PBS diagnostics |
| `pbsdsh` | PBS distributed shell | `pbs_rcp` | File transfer tool |
| `pbspoe` | Job launcher: IBM POE | `pbs_tclsh` | TCL shell with PBS API |
| `qalter` | Alter job | `pbsfs` | Report fairshare usage |
| `qdel` | Delete job | `pbskill` | Kill MS/Windows tasks |
| `qhold` | Hold a job | `pbsnodes` | Node manipulation tool |
| `qmove` | Move job | `printjob` | Report job details |
| `qmsg` | Send message to job | `qdisable` | Disable a queue |
| `qorder` | Reorder jobs | `qenable` | Enable a queue |
| `qrls` | Release hold on job | `qmgr` | PBS manager interface |
| `qselect` | Select jobs by criteria | `qrerun` | Requeue a running job |
| `qsig` | Send signal to job | `qrun` | Manually start a job |
| `qstat` | Status job, queue, server | `qstart` | Start a queue |
| `qsub` | Submit a job | `qstop` | Stop a queue |
| `xpbs` | Graphical User Interface | `qterm` | Shutdown PBS Server |
| | | `tracejob` | Report job history |
| | | `xpbsmon` | GUI monitoring tool |
| | | `xpbs -admin` | Graphical User Interface |

## 11.1 The pbs_hostid Command

The **pbs_hostid** command reports the host identifier (hostID) of the current host. This hostID is used by the PBS license manager to enforce node-locked licenses. The pbs_hostid command may be needed if you change your Server hardware and need to generate a new license key. The command usage is:

```
pbs_hostid [ -i | -n | -v]
```

The available options, and description of each, follows.

| Option | Description |
|---|---|
| -i | Prints the host identifier of the system |
| -n | Prints the primary hostname of the system |
| -v | Prints the version number of the PBS Pro Server |

## 11.2 The pbs_hostn Command

The **pbs_hostn** command takes a hostname, and reports the results of both gethostbyname(3) and gethostbyaddr(3) system calls. Both forward and reverse lookup of hostname and network addresses need to succeed in order for PBS to authenticate a host. Running this command can assist in troubleshooting problems related to incorrect or non-standard network configuration, especially within clusters. The command usage is:

```
pbs_hostn [ -v] hostname
```

The available options, and description of each, follows.

| Option | Description |
|---|---|
| -v | Turns on verbose mode |

## 11.3 The pbs_probe Command

The **pbs_probe** command reports post-installation information that is useful for PBS diagnostics. Aside from the direct information that is supplied on the command line, pbs_probe uses as the source for basic information the file pbs.conf and the values of any of the following environment variable that may be set in the environment in which pbs_probe is run: PBS_CONF, PBS_HOME, PBS_EXEC, PBS_START_SERVER, PBS_START_MOM, and PBS_START_SCHED. The command usage is:

```
    pbs_probe [ -f | -v ]
```

The available options, and description of each, follows.

| Option | Description |
|---|---|
| (default) | Run in "report" mode. In this mode pbs_probe reports on any errors in the PBS infrastructure files that it detects. The problems are categorized, and a list of the problem messages placed in each category are output. Those categories which are empty do not show in the output. |
| -f | Run in "fix" mode. In this mode pbs_probe will examine each of the relevant infrastructure files and, where possible, fix any errors that it detects, and print a message of what got changed. If it is unable to fix a problem, it will simply print a message regarding what was detected. |
| -v | Run in “verbose” mode. If the verbose option is turned on, pbs_probe will also output a complete list of the infrastructure files that it checked. |

## 11.4 The pbs_rcp vs. pbs_scp Command

The **pbs_rcp** command is used internally by PBS as the default file delivery mechanism. This default behavior can be changed to use SCP (**pbs_scp**) by so specifying in the PBS global configuration file. Specifically, to enable scp, set the PBS_SCP parameter as described in the discussion of “pbs.conf” on page 159.)

## 11.5 The pbsfs (PBS Fairshare) Command

The **`pbsfs`** command allows the Administrator to display or manipulate PBS fairshare usage data. The command usage is:

```
pbsfs [ -d | -e | -p | -t ]
pbsfs [ -c ent1 ent2 ] [ -g ent ] [ -s ent usage ]
```

The available options, and description of each, follows.

| Option | Description |
|---|---|
| `-c` *`ent1 ent2`* | Compare two *entities* and print the most deserving entity. |
| `-d` | Decay the fairshare tree (divide all values in half) |
| `-e` | Trim the fairshare tree to include only the entries in the resource_group file |
| `-g` *`ent`* | Print all data for specified *entity* and print the path from the root of the tree to the node. |
| `-p` | Print the fairshare tree in a flat format (default format). |
| `-s` *`ent usage`* | Set *entity's* usage value to *usage_value*. Please note that editing a non-leaf node is ignored. All non-leaf node usage values are calculated each time the scheduler is run or HUPed. |
| `-t` | Print the fairshare tree in a hierarchical format. |

**Important:** The `pbsfs` command can only be run as root (Unix) or any user with Administrator privilege (Windows). This command should only be used when the scheduler is not running. It is recommend that you first send a SIGHUP to the Scheduler (to force an update to be written to the usage file), then terminate the Scheduler. Be sure to restart the Scheduler after using the `pbsfs` command.

There are multiple parts to a fairshare node and you can print these data in different formats.The data displayed is:

| Data | Description |
|---|---|
| fairshare entity | the name of the entity to use in the fairshare tree |
| group | the group ID the entity is in (i.e. the entity’s parent) |
| cgroup | the group ID of this entity |
| shares | the number of shares the entity has |
| usage | the amount of usage |
| percentage | the percentage the entity has of the tree. Note that only the leaf nodes sum to 100%. If all of the nodes are summed, the result will be greater then 100%. Only the leaf nodes of the tree are fairshare entities. |
| usage / perc | The value the scheduler will use to the pick which entity has priority over another. The smaller the number the higher the priority. |
| path from root | The path from the root of the tree to the leaf node. This is useful because the scheduler will compare two entitiies by starting at the root, and working toward the leaf nodes, to determine which has the higher priority. |
| resource | The resource for which the scheduler accumulates usage for its fairshare calculations. This defaults to cput (cpu seconds) but can be set in the scheduler's config file. |

Whenever the fairshare usage database is changed, the original database is saved with the name “`usage.bak`”. Only one backup will be made.

## 11.6 The pbs_tclsh Command

The **`pbs_tclsh`** command is a version of the TCL shell (tclsh) linked with special TCL-wrapped versions of the PBS Pro external API library calls. This enables the user to write TCL scripts which utilize the PBS Pro API to query information. For usage see the `pbs_tclapi(3B)` manual page, and the **PBS Pro External Reference Specification.**

## 11.7 The pbs_wish Command

The **`pbs_wish`** command is a version of TK Window Shell linked with a wrapped versions of the PBS Pro external API library. For usage see the `pbs_tclapi(3B)` manual page, and the **PBS Pro External Reference Specification.**

## 11.8 The pbsnodes Command

The **`pbsnodes`** command is used to query the status of nodes, or mark nodes down, free or off-line. Node information is obtained by sending a request to the PBS Server. The command usage is:

```
pbsnodes [-a|-l|-s][-c node][-d node][-o node][-r node]
```

The available options, and description of each, follows.

| Option | Description |
|---|---|
| `(no option)` | Prints the command usage syntax. |
| *`node1 node2`* | Prints the status of nodes `node1` and `node2.` |
| `-a` | Lists all nodes and all their attributes. |
| `-c` *`nodes`* | Clear OFFLINE or DOWN from listed nodes. The listed nodes are "free" to be allocated to jobs. |
| `-d` *`nodes`* | The nodes specified as operands are marked DOWN and unavailable to run jobs. It is important that all the nodes known to be down are given as arguments on the command line. This is because nodes which are not listed are assumed to be UP and will be indicated as such if they were previously marked DOWN. I.e., *`"pbsnodes -d"`* will mark all nodes as free. |
| `-l` | List all nodes marked in any way. |

| Option | Description |
|---|---|
| `-o` *`nodes`* | Mark listed nodes as OFFLINE even if currently in use. An automated script that checks nodes being up or down and calls `pbsnodes` with a list of down nodes will not change the status of nodes marked OFFLINE. This gives the Administrator a tool to hold a node out of service without changing the automatic script. |
| `-r` *`nodes`* | Clear OFFLINE from listed nodes. |
| `-s` | Specify the PBS Server to which to connect. |

**Important:** Only the `-d` option will change the marking for nodes which are not given on the command line.

## 11.9 The printjob Command

The **`printjob`** command is used to print the contents of the binary file representing a PBS batch job saved within the PBS system. By default all the job data including job attributes are printed. This command is useful for troubleshooting, as during normal operation, the `qstat` command is the preferred method for displaying job-specific data and attributes. The command usage is:

```
printjob [ -a] file [file...]
```

The available options, and description of each, follows.

| Option | Description |
|---|---|
| `-a` | Suppresses the printing of job attributes. |

## 11.10 The qdisable Command

The `qdisable` command directs that a queue destination should no longer accept batch jobs. If the command is accepted, the destination will no longer accept Queue Job requests which specified the disabled queue. Jobs which already reside in the queue will continue to be processed. This allows a queue to be "drained." The command usage is:

```
qdisable destination ...
```

## 11.11 The qenable Command

The `qenable` command directs that a queue destination should accept batch jobs. This command sends a Manage request to the batch Server specified on the command line. If the command is accepted, the destination will accept Queue Job requests which specify the queue. The command usage is:

```
qenable destination ...
```

## 11.12 The qstart Command

The `qstart` command directs that a queue destination should process batch jobs. If the destination is an execution queue, the Server will begin to schedule jobs that reside in the queue for execution. If the destination is a routing queue, the Server will begin to route jobs from that queue. The command usage is:

```
qstart destination ...
```

## 11.13 The qstop Command

The `qstop` command directs that a queue destination should stop processing batch jobs. If the destination is an execution queue, the Server will cease scheduling jobs that reside in the queue for execution. If the destination is a routing queue, the Server will cease routing jobs from that queue. The command usage is:

```
qstop destination ...
```

## 11.14 The qrerun Command

The **`qrerun`** command directs that the specified jobs are to be rerun if possible. To rerun a job is to terminate the session leader of the job and return the job to the queued state in the execution queue in which the job currently resides. If a job is marked as not rerunable then the rerun request will fail for that job. (See also the discussion of the `-r` option to `qsub` in the **PBS Pro User Guide**.) The command usage is:

```
qrerun [ -W force ] jobID [ jobID ...]
```

The available options, and description of each, follows.

| Option | Description |
|---|---|
| `-W force` | This option, where `force` is the literal character string “force” directs that the job is to be requeued even if the node on which the job is executing is unreachable. |

## 11.15 The qrun Command

The **`qrun`** command is used to force a Server to initiate the execution of a batch job. The job is run regardless of scheduling position, resource requirements and availability, or state. The command usage is:

```
qrun [ -a ] [ -H host-spec ] jobID [ jobID ...]
```

The available options, and description of each, follows.

| Option | Description |
|---|---|
| `-a` | Specifies that the job is to be run "asynchronously". |
| `-H host-spec` | Specifies the host(s) within the cluster on which the job(s) are to be run. The *host-spec* argument can be any valid host specification that can be entered by the user in the `-l nodes=X` resource requirement, where the nodes must be managed by the local Server. If the option is not specified, the Server will select the "worst possible" host on which to execute the job. |

## 11.16 The qmgr Command

The **`qmgr`** command is the Administrator interface to PBS, and is discussed in detail earlier in this book, in the section entitled “The qmgr Command” on page 47.

## 11.17 The qterm Command

The **qterm** command is used to shut down PBS, and is discussed in detail earlier in this book, in section 10.2.5 “Stopping PBS” on page 168.

## 11.18 The tracejob Command

PBS includes the **tracejob** utility to extract log messages for a particular job (from all log files available on the local host) and print them sorted into chronological order. Usage for the tracejob command is:

```
tracejob [-a|s|l|m|v][-w size][-p path][-n days][-f filter] jobid
```

The available options, and description of each, follows.

| Option | Description |
|---|---|
| -a | Don't use accounting log files |
| -c | What message count is considered excessive |
| -f | Filter out types of log entries, multiple -f's can be specified to filter: error, system, admin, job, job_usage, security, sched, debug, debug2, or absolute numeric equivalent |
| -l | Don't use scheduler log files |
| -m | Don't use mom log files |
| -n | Number of days in the past to look for job(s) [default 1] |
| -p | Path to PBS_SERVER_HOME |
| -s | Don't use server log files |
| -w | Number of columns of your terminal |
| -v | Verbose mode - show more error messages |
| -z | Toggle filtering excessive messages |

The following example requests all log message for a particular job from today's (the default date) log file. Note that third column of the display contains a single letter (S, M, A, or L) indicating the source of the log message (**S**erver, **M**OM, **A**ccounting, or scheduLer log files).

```
% tracejob 475
Job: 475.riverside.pbspro.com
03/10/2002 14:29:15 S enqueuing into workq, state 1 hop 1
03/10/2002 14:29:15 S Job Queued at request of jjones, owner=
         jjones@mars.pbspro.com, job name = STDIN
03/10/2002 15:06:30 S Job Modified at request of Scheduler
03/10/2002 15:06:30 L Considering job to run
03/10/2002 15:06:30 S Job Run at request of Scheduler
03/10/2002 15:06:32 L Job run on node mars
03/10/2002 15:06:32 M Started, pid = 25282
03/10/2002 15:06:32 M Terminated
03/10/2002 15:06:32 M task 1 terminated
03/10/2002 15:06:32 M kill_job
03/10/2002 15:06:32 S Obit received
03/10/2002 15:06:32 S dequeuing from workq, state 5
03/10/2002 15:06:32 A user=jwang group=mrj jobname=subrun
         queue=workq ctime=1026928565 qtime=1026928565
         etime=1026928565 start=1026928848 exec_host=south/0
         Resource_List.arch=linux Resource_List.ncpus=1
         Resource_List.walltime=00:10:00 session=6022
         end=1026929149 Exit_status=0 resources_used.ncpus=1
         resources_used.cpupercent=0 resources_used.vmem=498kb
         resources_used.cput=00:00:00 resources_used.mem=224kb
         resources_used.walltime=00:05:01
```

## 11.19 The qalter Command and Job Comments

Users tend to want to know what is happening to their job. PBS provides a special job attribute, `comment` which is available to the operator, manager, or the Scheduler program. This attribute can be set to a string to pass information to the job owner. It might be used to display information about why the job is not being run or why a hold was placed on the job. Users are able to see this attribute, when set, by using the `-f and -s` option of the `qstat` command. (For details see "Displaying Job Comments" in the **PBS Pro User Guide**.) Operators and managers may use the `-W` option of the `qalter` command, for example

```
qalter -W comment="some text" job_id
```

## 11.20 The pbs-report Command

The **`pbs-report`** command allows the PBS Administrator to generate a report of job statistics from the PBS accounting logfiles. Options are provided to filter the data reported based on start and end times for the report, as well as indicating specific data that should be reported. The available options are shown below, followed by sample output of the `pbs-report` command.

**Important:** Before first using `pbs-report`, the PBS Administrator is advised to tune the `pbs-report` configuration file to match the local site. This can be done by editing the file `PBS_EXEC/lib/pm/PBS.pm`.

### 11.20.1 pbs-report Options

`--age -a seconds[:offset]`

Report age in seconds. If an offset is specified, the age range is taken from that offset backward in time, otherwise a zero offset is assumed. The time span is from (now - age - offset) to (now - offset). This option silently supersedes `--begin`, `--end`, and `--range`.

`--account account`

Limit results to those jobs with the specified account string. Multiple values may be concatenated with colons or specified with multiple instances of `--account`.

`--begin -b yyyymmdd[:hhmm[ss]]`

Report begin date and optional time (default: most recent log data).

`--count -c`

Display a numeric count of matching jobs. Currently only valid with `--cpumax` for use in monitoring rapidly-exiting jobs.

`--cpumax seconds`

Filter out any jobs which have more than the specified number of CPU seconds.

`--cpumin seconds`

Filter out any jobs which have less than the specified number of CPU seconds.

`--dept -d department`

Limit results to those jobs whose owners are in the indicated department (default: any). This option only works in conjunction with an LDAP server which supplies department codes. See also the `--group` option. Multiple values may be concatenated with colons or specified with multiple instances of `--dept`.

`--end -e yyyymmdd[:hhmm[ss]]`

Report end date and optional time (default: most recent log data).

`--exit -x integer`

Limit results to jobs with the specified exit status (default: any).

`--group -g group`

Limit results to the specified group name. Multiple values may be concatenated with colons or specified with multiple instances of `--group`.

`--help -h`

Prints a brief help message and exits.

`--host -m execution host`

Limit results to the specified execution host. Multiple values may be concatenated with colons or specified with multiple instances of `--host`.

`--inclusive key`

Limit results to jobs which had *both* start and end times in the range.

`--index -i key`

Field on which to index the summary report (default: user). Valid values include: date, dept, host, package, queue, user.

`--man`

Prints the manual page and exits.

`--negate -n option name`

Logically negate the selected options; print all records *except* those that match the values for the selected criteria (default: unset; valid values: account, dept, exit, group, host, package, queue, user). Defaults cannot be negated; only options explicitly specified are negated. Multiple values may be concatenated with colons or specified with multiple instances of `--negate`.

`--package -p package`

Limit results to the specified software package. Multiple values may be concatenated with colons or specified with multiple instances of `--package`. Valid values are can be seen by running a report with the `--index package` option. This option keys on custom resources requested at job submission time. Sites not using such custom resources will have all jobs reported under the catch-all *None* package with this option.

`--point yyyymmdd[:hhmm[ss]]`

Print a report of all jobs which were actively running at the point in time specified. This option cannot be used with any other date or age option.

`--queue -q queue`

Limit results to the specified queue. Multiple values may be concatenated with colons or specified with multiple instances of `--queue`. The special name *all* may be used to select all queues when a default queue has been configured.

`--range -r date range`

Provides a shorthand notation for current date ranges (default: all). Valid values are today, week, month, quarter, and year. This option silently supersedes `--begin` and `--end`, and is superseded by `--age`.

`--reslist`

Include resource requests for all matching jobs. This option is mutually exclusive with `--verbose`.

`--sched -t`

Generate a brief statistical analysis of scheduler cycle times. No other data on jobs is reported.

`--sort -s field`

Field by which to sort reports (default: user). Valid values are cpu, date, dept, host, jobs, package, queue, suspend (aka muda), wait, and wall.

`--time option`

Used to indicate how time should be accounted. The default of `full` is to count the entire job's CPU and wall time in the report if the job ended during the report's date range. Optionally the `partial` option is used to cause only CPU and wall time during the report's date range to be counted.

`--user -u username`

Limit results to the specified user name. Multiple values may be concatenated with colons or specified with multiple instances of `--user`.

`--verbose -v`

Include attributes for all matching individual jobs (default: summary only).

`--vsort field`

Field by which to sort the verbose output section reports (default: jobid). Valid values are cpu, date, exit, host, jobid, jobname, mem, name, package, queue, scratch, suspend, user, vmem, wall, wait. If neither `--verbose` nor `--reslist` is specified, `--vsort` is silently ignored. The scratch sort option is available only for resource reports (`--reslist`).

`--waitmax seconds`

Filter out any jobs which have more than the specified wait time in seconds.

`--waitmin seconds`

Filter out any jobs which have less than the specified wait time in seconds.

`--wallmax seconds`

Filter out any jobs which have more than the specified wall time in seconds.

`--wallmin seconds`

Filter out any jobs which have less than the specified wall time in seconds.

`--wall -w`

Use the walltime resource attribute rather than wall time calculated by subtracting the job start time from end time. The walltime resource attribute does *not* accumulate when a job is suspended for any reason, and thus may not accurately reflect the local interpretation of wall time.

Several options allow for filtering of which jobs to include. These options are as follows.

`--begin, --end, --range, --age, --point`

Each of these options allows the user to filter jobs by some range of dates or times. `--begin` and `--end` work from hard date limits. Omitting either will cause the report to contain all data to either the beginning or the end of the accounting data. Unbounded date reports may take several minutes to run, depending on the volume of work logged. `--range` is a shorthand way of selecting a prior date range and will supersede `--begin` and `--end`. `--age` allows the user to select an arbitrary period going back a specified number of seconds from the time the report is run. `--age` will silently supersede all other date options. `--point` displays all jobs which were running at the specified point in time, and is incompatible with the other options. `--point` will produce an error if specified with any other date-related option.

`--cpumax, --cpumin`
`--waitmax, --waitmin`
`--wallmax, --wallmin`

Each of these six options sets a filter which bounds the jobs on one of their three time attributes (CPU time, queue wait time, or wall time). A maximum value will cause any jobs with more than the specified amount to be ignored. A minimum value will cause any jobs with less than the specified amount to be ignored. All six options may be combined, though doing so will often restrict the filter such that no jobs can meet the requested criteria. Combine time filters for different time with caution.

`--dept, --group, --user`

Each of these user-based filters allow the user to filter jobs based on who submitted them. `--dept` allows for integration with an LDAP server and will generate reports based on department codes as queried from that server. If no LDAP server is available, department-based filtering and sorting will not function. `--group` allows for filtering of jobs by primary group ownership of the submitting user, as defined by the operating system on which the PBS server runs. `--user` allows for explicit naming of users to be included. It is possible to specify a list of values for these filters, by providing a single colon-concatenated argument or using the option multiple times, each with a single value.

`--account`

This option allows the user to filter jobs based on an arbitrary, user-specified job account string. The content and format of these strings is site-defined and unrestricted. Most often a custom job front-end at the customer site will enforce restrictions on permissible account strings, which are passed to qsub with qsub's -A option.

`--host, --exit, --package, --queue`

Each of these job-based filters allow the user to filter jobs based on some property of the job itself. `--host` allows for filtering of jobs based on the host on which the job was executed. `--exit` allows for filtering of jobs based on the job exit code. `--package` allows for filtering of jobs based on the software package used in the job. This option will only function when a package-specific custom resource is defined for the PBS server and requested by the jobs as they are submitted. `--queue` allows for filtering of jobs based on the queue in which the job finally executed. With the exception of `--exit`, it is possible to specify a list of values for these filters, by providing a single colon-concatenated argument or using the option multiple times, each with a single value.

`--negate`

The `--negate` option bears special mentioning. It allows for logical negation of one or more specified filters. Only the account, dept, exit, group, host, package, queue, and user filters may be negated. If a user is specified with `--user`, and the '`--negate user`' option is used, only jobs *not* belonging to that user will be included in the report. Multiple report filters may be negated by providing a single colon-concatenated argument or using `--negate` multiple times, each with a single value.

Several report types can be generated, each indexed and sorted according to the user's needs.

`--verbose`

This option generates a wide tabular output with detail for every job matching the filtering criteria. It can be used to generate output for import to a spreadsheet which can manipulate the data beyond what `pbs_report` currently provides. Verbose reports may be sorted on any field using the `--vsort` option. The default with no option specified is to produce a summary report only.

`--reslist`

This option generates a tabular output with detail on resources requested (*not* resources used) for every job matching the filtering criteria. Resource list reports may be sorted on any field using the `--vsort` option. The default with no option specified is to produce a summary report only.

`--inclusive`

Normal convention is to credit a job's entire run to the time at which it ends. So all date selections are bounds around the end time. This option allows a user to require that the job's start time also falls within the date range.

`--index`

This option allows the user to select a field on which data in the summary should be grouped. The fields listed in the option description are mutually exclusive. Only one can be chosen, and will represent the left-most column of the summary report output. One value may be selected as an index while another is selected for sorting. However, since index values are mutually exclusive, the only sort options which may be used (other than the index itself) are account, cpu, jobs, suspend, wait, and wall. If no sort order is selected, the index is used as the sort key for the summary.

`--sort`

This option allows the user to specify a field on which to sort the summary report. It operates independently of the sort field for verbose reports (see `--vsort`). See the description for `--index` for notes on how the two options interact.

`--vsort`

This option allows the user to specify a field on which to sort the verbose report. It operates independently of the sort field for summary reports (see `--sort`).

`--time`

This option allows the user to modify how time associated with a job is accounted. With *full*, all time is accounted for the job, and credited at the point when the job ended. For a job which ended a few seconds after the report range begins, this can cause significant overlap, which may boost results. During a sufficiently large time frame, this overlap effect is negligible and may be ignored. This value for `--time` should be used when generating monthly usage reports.

With *partial*, any CPU or wall time accumulated prior to the beginning of the report is ignored. *partial* is intended to allow for more accurate calculation of overall cluster efficiency during short time spans during which a significant 'overlap' effect can skew results.

### 11.20.2 pbs-report Examples

This section explains several complex report queries to serve as examples for further experimentation. Note that some of options to `pbs-report` produce summary information of the resources requested by jobs (such as mem, vmem, ncpus, etc.). These resources are explained in Chapter 4 of the **PBS Pro User Guide**.

Consider the following question: "This month, how much resources did every job which waited more than 10 minutes request?"

```
pbs-report --range month --waitmin 600 --reslist
```

This information might be valuable to determine if some simple resource additions (e.g. more memory or more disk) might increase overall throughput of the cluster. At the bottom of the summary statistics, prior to the job set summary, is a statistical breakdown of the values in each column. For example:

```
             # of       Total       Total           Average
Date         jobs    CPU Time   Wall Time Efcy.   Wait Time
----------  -----   ----------  ----------  -----  --------
--
TOTAL        1900    10482613    17636290 0.594        1270
... individual rows indexed by date ...
Minimum         4        4715       13276 0.054         221
Maximum       162     1399894     2370006 1.782       49284
Mean           76      419304      705451 0.645        2943
Deviation      41      369271      616196 0.408        9606
Median         80      242685      436724 0.556         465
```

This summary should be read in column format. The values each represent a statistical data point in the column. For instance, while the minimum number of jobs run in one day was 4 and the maximum 162, these values do ***not*** correlate to the 4715 and 1399894 CPU seconds listed as minimums and maximums.

In the Job Set Summary section, the values should be read in rows, as shown here:

```
                                                   Standard
           Minimum     Maximum         Mean   Deviation      Median
          ----------  ----------  ----------  ----------  ----------
CPU time           0       18730         343         812           0
Wall time          0      208190        8496       19711          93
Wait time          0      266822        4129        9018           3
```

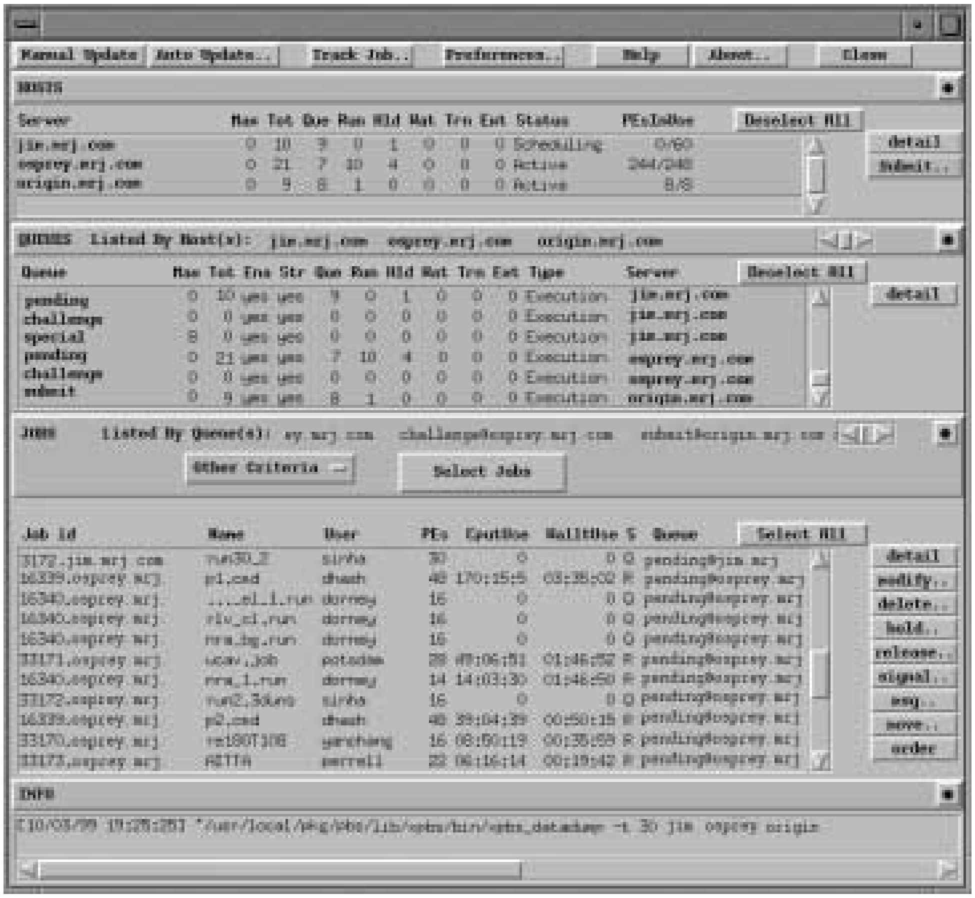
These values represent aggregate statistical analysis for the entire set of jobs included in the report. The values in the prior summary represent values over the set of totals based on the summary index (e.g. Maximum and Minimum are the maximum and minimum totals for a given day/user/department, rather than an individual job. The job set summary represents an analysis of all individual jobs.

### 11.20.3 pbs-report Cluster Monitoring

The pbs-report options --count and --cpumax are intended to allow an Administrator to periodically run this report to monitor for jobs which are exiting rapidly, representing a potential global error condition causing all jobs to fail. It is most useful in conjunction with --age, which allows a report to span an arbitrary number of seconds backward in time from the current moment. A typical set of options would be "--count --cpumax 30 --age 21600", which would show a total number of jobs which consumed less than 30 seconds of CPU time within the last six hours.

## 11.21 The xpbs Command (GUI) Admin Features

PBS currently provides two Graphical User Interfaces (GUIs): xpbs (intended primarily for users) and xpbsmon (intended for PBS operators and managers). Both are built using the Tool Control Language Toolkit (TCL/tk). The first section below discusses the user GUI, xpbs. The following section discusses xpbsmon.



```
Server                Max Tot Que Run Hld Wat Trn Ext Status      PEsInUse
jim.mrj.com             0  10   9   0   1   0   0   0 Scheduling      0/60
osprey.mrj.com          0  21   7  10   4   0   0   0 Active       244/248
origin.mrj.com          0   9   8   1   0   0   0   0 Active           8/8
```

```
QUEUES  Listed By Host(s):  jim.mrj.com  osprey.mrj.com  origin.mrj.com

Queue             Max Tot Ena Str Que Run Hld Wat Trn Ext Type       Server
pending             0  10 yes yes   9   0   1   0   0   0 Execution  jim.mrj.com
challenge           0   0 yes yes   0   0   0   0   0   0 Execution  jim.mrj.com
special             8   0 yes yes   0   0   0   0   0   0 Execution  jim.mrj.com
pending             0  21 yes yes   7  10   4   0   0   0 Execution  osprey.mrj.com
challenge           0   0 yes yes   0   0   0   0   0   0 Execution  osprey.mrj.com
submit              0   9 yes yes   8   1   0   0   0   0 Execution  origin.mrj.com
```

```
JOBS    Listed By Queue(s): ey.mrj.com  challenge@osprey.mrj.com  submit@origin.mrj.com

Job id             Name          User      PEs  CputUse   WalltUse S  Queue
3172.jim.mrj.com   run30_2       sinha      30         0         0 Q  pending@jim.mrj
16339.osprey.mrj   p1.cmd        dhash      48 170:15:5  03:35:02 R  pending@osprey.mrj
16340.osprey.mrj   ....el_1.run  dorney     16         0         0 Q  pending@osprey.mrj
16340.osprey.mrj   rlv_el.run    dorney     16         0         0 Q  pending@osprey.mrj
16340.osprey.mrj   nra_bg.run    dorney     16         0         0 Q  pending@osprey.mrj
33171.osprey.mrj   ucav.job      potadma    28 49:06:51  01:46:52 R  pending@osprey.mrj
16340.osprey.mrj   nra_1.run     dorney     14 14:03:30  01:46:50 R  pending@osprey.mrj
33172.osprey.mrj   run2.3dvns    sinha      16         0         0 Q  pending@osprey.mrj
16339.osprey.mrj   p2.cmd        dhash      48 39:04:39  00:50:15 R  pending@osprey.mrj
33170.osprey.mrj   re180T108     yanchang   16 08:50:119 00:35:59 R  pending@osprey.mrj
33173.osprey.mrj   AETTA         perrell    22 06:16:14  00:19:42 R  pending@osprey.mrj
```

```
[10/03/99 19:28:25] "/usr/local/pkg/pbs/lib/xpbs/bin/xpbs_datadump -t 30 jim osprey origin
```

### 11.21.1 xpbs GUI Configuration

**xpbs** provides a user-friendly point-and-click interface to the PBS commands. To run `xpbs` as a regular, non-privileged user, type:

```
% setenv DISPLAY your_workstation_name:0
% xpbs
```

To run `xpbs` with the additional purpose of terminating PBS Servers, stopping and starting queues, running/rerunning jobs (as well as then run:

```
% xpbs -admin
```

**Important:** Table 7 in the **PBS Pro User Guide** lists all functionality of `xpbs`, and identifies which are available only via the `-admin` option.

Running `xpbs` will initialize the X resource database from various sources in the following order:

1. The RESOURCE_MANAGER property on the root window (updated via `xrdb`) with settings usually defined in the `.Xdefaults` file

2. Preference settings defined by the system Administrator in the global `xpbsrc` file

3. User's `~/.xpbsrc` file-- this file defines various X resources like fonts, colors, list of PBS hosts to query, criteria for listing queues and jobs, and various view states.

The system Administrator can specify a global resources file to be read by the GUI if a personal `.xpbsrc` file is missing: *`/usr/pbs/`*`lib/xpbs/xpbsrc`. Keep in mind that within an Xresources file (Tk only), later entries take precedence. For example, suppose in your `.xpbsrc` file, the following entries appear in order:

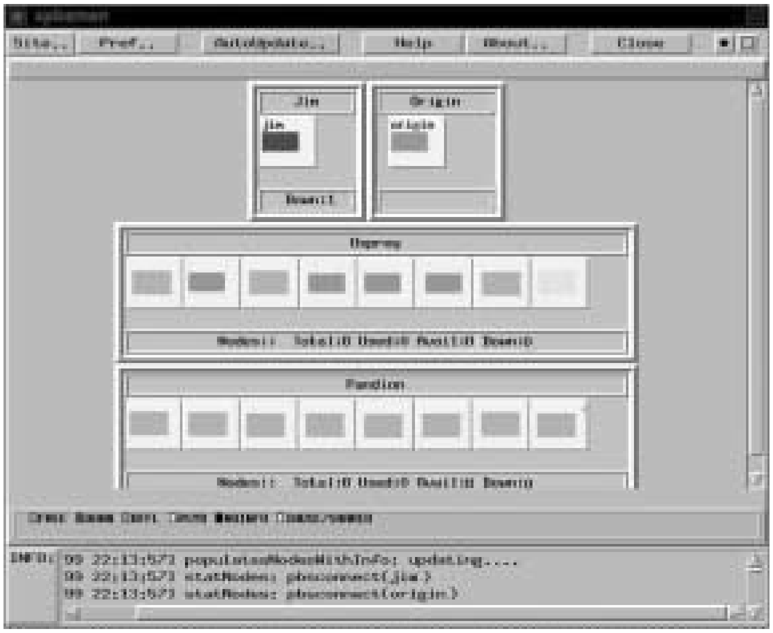
```
xpbsrc*backgroundColor: blue
*backgroundColor: green
```

The later entry "green" will take precedence even though the first one is more precise and longer matching. The things that can be set in the personal preferences file are fonts, colors, and favorite Server host(s) to query.

`xpbs` usage, command correlation, and further customization information is provided in the **PBS Pro User Guide**, Chapter 5, “Using the xpbs GUI”.

## 11.22 The xpbsmon GUI Command

**xpbsmon** is the node monitoring GUI for PBS. It is used for graphically displaying information about execution hosts in a PBS environment. Its view of a PBS environment consists of a list of sites where each site runs one or more Servers, and each Server runs jobs on one or more execution hosts (nodes).



```
INFO: 09 22:13:57] populateNodesWithInfo: updating....
      09 22:13:57] statNodes: pbsconnect(jim)
      09 22:13:57] statNodes: pbsconnect(origin)
```

The system Administrator needs to define the site’s information in a global X resources file, *`/usr/pbs/lib/`*`xpbsmon/xpbsmonrc` which is read by the GUI if a personal `.xpbsmonrc` file is missing. A default `xpbsmonrc` file usually would have been created already during installation, defining (under *sitesInfo resource) a default site name, list of Servers that run on a site, set of nodes (or execution hosts) where jobs on a particular Server run, and the list of queries that are communicated to each node’s `pbs_mom.` If node queries have been specified, the host where `xpbsmon` is running must have been given explicit permission by the `pbs_mom` daemon to post queries to it. This is done by including a `$restricted` entry in the MOM’s config file.

It is not recommended to manually update the *sitesInfo value in the xpbsmonrc file as its syntax is quite cumbersome. The recommended procedure is to bring up xpbsmon, click on "Pref.." button, manipulate the widgets in the Sites, Server, and Query Table dialog boxes, then click "Close" button and save the settings to a .xpbsmonrc file. Then copy this file over to the */usr/pbs/*lib/xpbsmon/ directory.

## 11.23 The pbskill Command

Under Microsoft Windows XP and Windows 2000, PBS includes the **pbskill** utility to terminate any job related tasks or processes. DOS/windows prompt usage for the pbskill utility is:

```
pbskill processID1 [[processID2] [processID3] ... ]
```

# Chapter 12
# Example Configurations

Up to this point in this manual, we have seen many examples of how to configure the individual PBS daemons, set limits, and otherwise tune a PBS installation. Those examples were used to illustrate specific points or configuration options. This chapter pulls these various examples together into configuration-specific scenarios which will hopefully clarify any remaining configuration questions. Four configuration models are discussed, followed by several complex example of specific features.

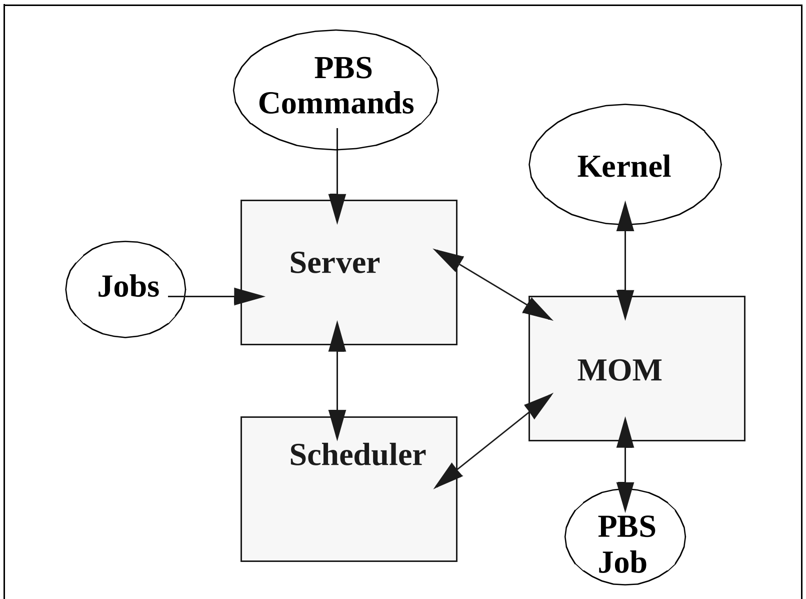
Single Node Time-sharing System
Single Node System with Separate PBS Server
Multi-node Time-sharing Cluster
Multi-node Space-sharing Cluster
Complex Multi-level Route Queues (including group ACLs)
Multiple User ACLs

For each of these possible configuration models, the following information is provided:

General description for the configuration model
Type of system the model for which the model is well suite
Contents of Server nodes file
Any required Server configuration
Any required MOM configuration
Any required Scheduler configuration

## 12.1 Single Node Time-sharing System

Running PBS on a single node/host as a standalone time-sharing system is the least complex configuration. This model is most applicable to sites who have a single large Server system, a single SMP system (e.g. an SGI Origin server), or even a vector supercomputer. In this model, all three PBS daemons run on the same host, which is the same host on which jobs will be executed, as shown in the figure below.



All daemons on a single host.

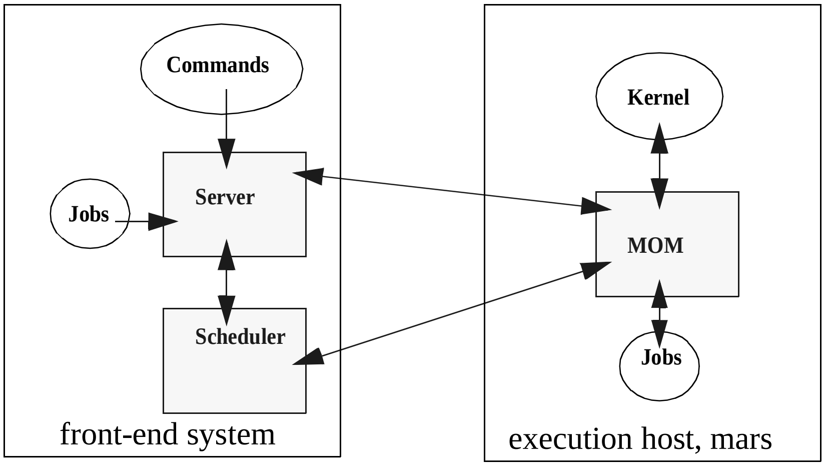
For this example, lets assume we have a 32-CPU server machine named “mars”. We want users to log into “mars” and jobs will be run via PBS on mars.

In this configuration, the default PBS `nodes` file (which should contain the name of the host on which the Server was installed) is sufficient. Our example `nodes` file would contain only one entry: `mars:ts`

The default MOM and Scheduler `config` files, as well as the default queue/Server limits are also sufficient in order to run jobs. No changes are required from the default configuration, however you may wish to customize PBS to your site.

## 12.2 Single Timesharing Node with Separate Server

A variation on the model presented above would be to provide a “front-end” system that ran the PBS Server and Scheduler, and from which users submitted their jobs. Only the MOM daemon would run on our execution server, mars. This model is recommended when the user load would otherwise interfere with the computational load on the Server.



In this case, the PBS `server_priv/nodes` file would contain the name of our execution server mars, but this may not be what was written to the file during installation, depending on which options were selected. It is possible the hostname of the machine on which the Server was installed was added to the file, in which case you would need to either manually edit the `nodes` file, or use `qmgr(1B)` to manipulate the contents to contain one node: `mars:ts`. If the default scheduling policy, based on available CPUs and memory, meets your requirements, then no changes are required in either the MOM or Scheduler configuration files.

However, if you wish the execution node (mars) to be scheduled based on load average, the following changes are needed. Edit MOM’s `mom_priv/config` file so that it contains the target and maximum load averages, e.g.:

```
$ideal_load 30
$max_load 32
```

In the scheduler `sched_priv/config` file, the following options would need to be set:

```
load_balancing: true all
```

## 12.3 Multi-node Timesharing Cluster

The multi-node time-sharing cluster model is a very common configuration for PBS. In this model, there is typically a “front-end” system as we saw in the previous example, with a number of ‘back-end” compute nodes. The PBS Server and Scheduler are typically run on the front-end system, and a MOM daemon is run on each of the execution nodes, as shown in the diagram to the right.

In this model, the PBS `nodes` file will need to contain the list of all the nodes in the cluster, with the timesharing attribute `:ts` appended:

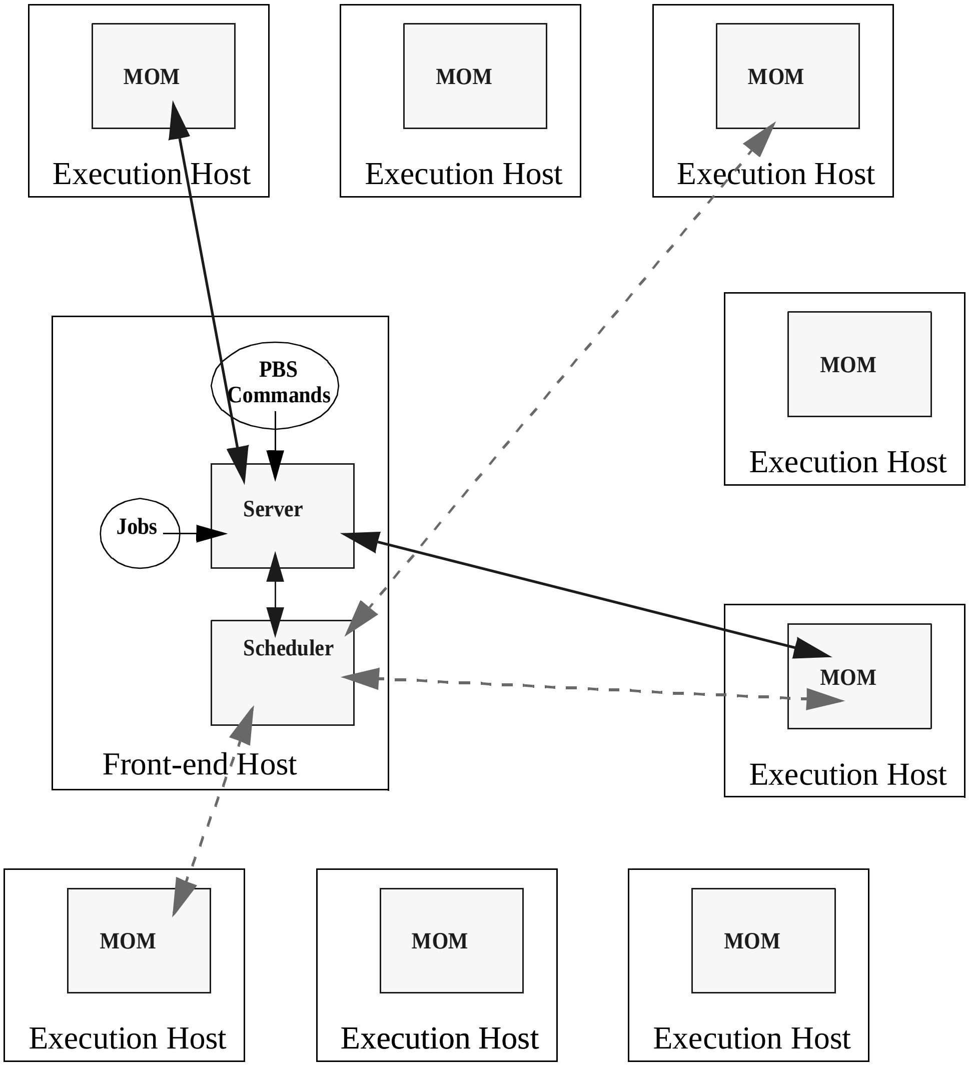
```
mercury:ts
venus:ts
earth:ts
mars:ts
jupiter:ts
saturn:ts
uranus:ts
neptune:ts
```

The MOM `config` file on each node will need two static resources added, to specify the target load for each node. If we assume each of the nodes in our planets cluster is a 32-processor system, then the following example shows what might be desirable values to add to the MOM `config` files:

```
$ideal_load 30
$max_load 32
```

Furthermore, suppose we want the Scheduler to load balance the workload across the available nodes, making sure not to run two job in a row on the same node (round robin node scheduling). We accomplish this by editing the Scheduler configuration file and enabling load_balancing:

```
load_balancing: true all
smp_cluster_dist: round_robin
```



This diagram illustrates a multi-node cluster configuration wherein the Scheduler and Server communicate with the MOMs on the execution nodes. Jobs are submitted to the Server, scheduled for execution by the Scheduler, and then transferred to a MOM when it's time to be run. MOM periodically sends status information back to the Server, and answers resource requests from the Scheduler.

## 12.4 Multi-node Space-sharing Cluster

A variation on the time-sharing cluster is the “space-shared” cluster. In this context, space-sharing refers to assigning an entire node (or set of nodes) to a single job at a time, for the duration of the job. This is usually done in order to achieve high-throughput and predictability of runtimes for a specific application (such as message-passing parallel jobs).

In this model, the PBS `nodes` file would **not** have the `:ts` appended to the node names, e.g.:

```
mercury
venus
earth
mars
jupiter
saturn
uranus
neptune
```

There would be no edits required to either the Scheduler or the MOM `config` files.

Lastly, since in this model users specify their job resource requirements using the “`-l nodes=...`” syntax of `qsub`, we need to set node-specific limits in the Server:

```
# qmgr
Qmgr: set server resources_default.nodes = 1
Qmgr: set server resources_default.nodect = 1
Qmgr: set server resources_default.neednodes = 1
```

## 12.5 Complex Multi-level Route Queues

There are times when a site may wish to create a series of route queues in order to filter jobs, based on specific resources, or possibly to different destinations. For this example, consider a site that has two large Server systems, and a Linux cluster. The Administrator wants to configure route queues such that everyone submits jobs to a single queue, but the jobs get routed based on (1) requested architecture and (2) individual groupIDs. In other words, users request the architecture they want, and PBS find the right queue for them. Only groups “math”, “chemistry”, and “physics” are permitted to use either server systems; while anyone can use the cluster. Lastly, the jobs coming into the cluster should be divided into three separate queues for long, short, and normal jobs. But the “long” queue was created for the astrology department, so only members of that group should be permitted into that queue. Given these requirements, lets look at how we would set up such collection of route queues. (Note that this is only one way to accomplish this task. There are various other ways too.)

First we create a queue to which everyone will submit their jobs. Let’s call it “submit”. It will need to be a route queue with three destinations, as shown:

```
# qmgr
Qmgr: create queue submit
Qmgr: set queue submit queue_type = Route
Qmgr: set queue submit route_destinations = server_1
Qmgr: set queue submit route_destinations += server_2
Qmgr: set queue submit route_destinations += cluster
Qmgr: set queue submit enabled = True
Qmgr: set queue submit started = True
```

Now we need to create the destination queues. (Notice in the above example, we have already decided what to call the three destinations: `server_1`, `server_2`, `cluster`.) First we create the `server_1` queue, complete with a group ACL, and a specific architecture limit.

```
Qmgr: create queue server_1
Qmgr: set queue server_1 queue_type = Execution
Qmgr: set queue server_1 from_route_only = True
Qmgr: set queue server_1 resources_max.arch = irix6
Qmgr: set queue server_1 resources_min.arch = irix6
Qmgr: set queue server_1 acl_group_enable = True
Qmgr: set queue server_1 acl_groups = math
Qmgr: set queue server_1 acl_groups += chemistry
Qmgr: set queue server_1 acl_groups += physics
Qmgr: set queue server_1 enabled = True
Qmgr: set queue server_1 started = True
```

Next we create the queues for `server_2` and `cluster`. Note that the `server_2` queue is very similar to the `server_1` queue, only the architecture differs. Also notice that the `cluster` queue is another route queue, with multiple destinations.

```
Qmgr: create queue server_2
Qmgr: set queue server_2 queue_type = Execution
Qmgr: set queue server_2 from_route_only = True
Qmgr: set queue server_2 resources_max.arch = sv2
Qmgr: set queue server_2 resources_min.arch = sv2
Qmgr: set queue server_2 acl_group_enable = True
Qmgr: set queue server_2 acl_groups = math
Qmgr: set queue server_2 acl_groups += chemistry
Qmgr: set queue server_2 acl_groups += physics
Qmgr: set queue server_2 enabled = True
Qmgr: set queue server_2 started = True
Qmgr: create queue cluster
Qmgr: set queue cluster queue_type = Route
Qmgr: set queue cluster from_route_only = True
Qmgr: set queue cluster resources_max.arch = linux
Qmgr: set queue cluster resources_min.arch = linux
Qmgr: set queue cluster route_destinations = long
Qmgr: set queue cluster route_destinations += short
Qmgr: set queue cluster route_destinations += medium
Qmgr: set queue cluster enabled = True
Qmgr: set queue cluster started = True
```

In the cluster queue above, you will notice the particular order of the three destination queues (`long`, `short`, `medium`). PBS will attempt to route a job into the destination queues in the order specified. Thus, we want PBS to first try the `long` queue (with will have an ACL on it), then the `short` queue (with its short time limits). Thus any jobs that had not been routed into any other queues (server or cluster) will end up in the `medium` cluster queue. Now to create the remaining queues.

```
Qmgr: create queue long
Qmgr: set queue long queue_type = Execution
Qmgr: set queue long from_route_only = True
Qmgr: set queue long resources_max.cput = 20:00:00
Qmgr: set queue long resources_max.walltime = 20:00:00
Qmgr: set queue long resources_min.cput = 02:00:00
Qmgr: set queue long resources_min.walltime = 03:00:00
Qmgr: set queue long acl_group_enable = True
Qmgr: set queue long acl_groups = astrology
Qmgr: set queue long enabled = True
Qmgr: set queue long started = True
```

```
Qmgr: create queue short
Qmgr: set queue short queue_type = Execution
Qmgr: set queue short from_route_only = True
Qmgr: set queue short resources_max.cput = 01:00:00
Qmgr: set queue short resources_max.walltime = 01:00:00
Qmgr: set queue short enabled = True
Qmgr: set queue short started = True
Qmgr: create queue medium
Qmgr: set queue medium queue_type = Execution
Qmgr: set queue medium from_route_only = True
Qmgr: set queue medium enabled = True
Qmgr: set queue medium started = True

Qmgr: set server default_queue = submit
```

Notice that the `long` and `short` queues have time limits specified. This will ensure that jobs of certain sizes will enter (or be prevented from entering) these queues. The last queue, `medium`, has no limits, thus it will be able to accept any job that is not routed into any other queue.

Lastly, note the last line in the example above, which specified that the default queue is the new `submit` queue. This way users will simply submit their jobs with the resource and architecture requests, without specifying a queue, and PBS will route the job into the correct location.

For example, if a user submitted a job with the following syntax, the job would be routed into the `server_2` queue:

```
qsub -l arch=sv2,ncpus=4 testjob
```

## 12.6 Multiple User ACL Example

A site may have a need to restrict individual users to particular queues. In the previous example we set up queues with group-based ACLs, in this example we show user-based ACLs. Say a site has two different groups of users, and want to limit them to two separate queues (perhaps with different resource limits). The following example illustrates this.

```
Qmgr: create queue structure
Qmgr: set queue structure queue_type = Execution
Qmgr: set queue structure acl_user_enable = True
Qmgr: set queue structure acl_users = curly
Qmgr: set queue structure acl_users += jerry
Qmgr: set queue structure acl_users += larry
Qmgr: set queue structure acl_users += moe
Qmgr: set queue structure acl_users += tom
Qmgr: set queue structure resources_max.nodes = 48
Qmgr: set queue structure enabled = True
Qmgr: set queue structure started = True
Qmgr:
Qmgr: create queue engine
Qmgr: set queue engine queue_type = Execution
Qmgr: set queue engine acl_user_enable = True
Qmgr: set queue engine acl_users = bill
Qmgr: set queue engine acl_users += bobby
Qmgr: set queue engine acl_users += chris
Qmgr: set queue engine acl_users += jim
Qmgr: set queue engine acl_users += mike
Qmgr: set queue engine acl_users += rob
Qmgr: set queue engine acl_users += scott
Qmgr: set queue engine resources_max.nodes = 12
Qmgr: set queue engine resources_max.walltime=04:00:00
Qmgr: set queue engine enabled = True
Qmgr: set queue engine started = True
Qmgr:
```

## 12.7 External Software License Management

PBS Pro can be configured to manage (and schedule) externally-controlled licensed software (such as applications licensed via FlexLM). A detailed example is provided in section 9.5.2 “System-wide Resource Example” on page 155.

# Chapter 13
# Problem Solving

The following is a list of common problems and recommended solutions. Additional information is always available online at the PBS website, www.pbspro.com. The last section in this chapter gives important information on how to get additional assistance from the PBS Support staff.

## 13.1 Directory Permission Problems

If for some reason the access permissions on the PBS file tree are changed from their default settings, a component of the PBS system may detect this as a security violation, and refuse to execute. If this is the case, an error message to this effect will be written to the corresponding log file. You can run the `pbs_probe` command to check (and optionally correct) any directory permission (or ownership) problems. For details on usage of the `pbs_probe` command see section 11.3 “The pbs_probe Command” on page 198.

## 13.2 Resource Temporarily Unavailable - 15044

The "Resources temporarily unavailable" message (error 15044) is returned from the Server in the situation where a job has requested nodes with certain properties or attributes, and such nodes do exist within the cluster, but at least one of the required nodes is not currently available. This can happen if the required node is currently assigned to another job, or if the node went down between the time the Scheduler decided to run the job and when the Server attempted to start the job.

## 13.3 Clients Unable to Contact Server

If a client command (such as `qstat` or `qmgr`) is unable to connect to a Server there are several possibilities to check. If the error return is `15034`, “`No server to connect to`”, check (1) that there is indeed a Server running and (2) that the default Server information is set correctly. The client commands will attempt to connect to the Server specified on the command line if given, or if not given, the Server specified by **`SERVER_NAME`** in `pbs.conf`.

If the error return is `15007`, “`No permission`”, check for (2) as above. Also check that the executable `pbs_iff` is located in the search path for the client and that it is setuid root. Additionally, try running `pbs_iff` by typing:

```
pbs_iff -t server_host 15001
```

Where *`server_host`* is the name of the host on which the Server is running and `15001` is the port to which the Server is listening (if started with a different port number, use that number instead of `15001`). Check for an error message and/or a non-zero exit status. If `pbs_iff` exits with no error and a non-zero status, either the Server is not running or was installed with a different encryption system than was `pbs_iff`.

## 13.4 Nodes Down

The PBS Server determines the state of nodes (`up` or `down`), by communicating with MOM on the node. The state of nodes may be listed by two commands: `qmgr` and `pbsnodes`.

```
% qmgr
Qmgr: list node @active

% pbsnodes -a
Node jupiter
        state = down, state-unknown
        properties = sparc, mine
        ntype = cluster
```

A node in PBS may be marked “`down`” in one of two substates. For example, the state above of node “jupiter” shows that the Server has not had contact with MOM since the Server came up. Check to see if a MOM is running on the node. If there is a MOM and if the MOM was just started, the Server may have attempted to poll her before she was up. The Server should see her during the next polling cycle in 10 minutes. If the node is still marked “`down, state-unknown`” after 10+ minutes, either the node name specified in the Server’s node file does not map to the real network hostname or there is a network problem between the Server’s host and the node.

If the node is listed as

```
% pbsnodes -a
Node jupiter
        state = down
        properties = sparc, mine
        ntype = cluster
```

then the Server has been able to ping MOM on the node in the past, but she has not responded recently. The Server will send a “ping” PBS message to every free node each ping cycle, 10 minutes.  If a node does not acknowledge the ping before the next cycle, the Server will mark the node down.

On an IBM SP, a node may also be marked down if MOM on the node believes that the node is not connected to the high speed switch. When the Server receives an acknowledgment from MOM on the node, the node will again be marked up (free).

## 13.5 Requeueing a Job “Stuck” on a Down Node

PBS Pro will detect if a node fails when a job is running on it, and will automatically requeue and schedule the job to run elsewhere. If the user marked the job as “not rerunable” (i.e. via the `qsub -r n` option), the then job will be deleted rather than requeued. If the affected node is node 0 (Mother Superior), the requeue will occur quickly. If it is another node in the set assigned to the job, it could take a few minutes before PBS takes action to requeue or delete the job. However, if the auto-requeue feature is not enabled (see “node_fail_requeue” on page 57), or if you wish to act immediately, you can manually force the requeueing and/or rerunning of the job.

If you wish to have PBS simply remove the job from the system, use the “`-Wforce`” option to qdel:

```
% qdel -Wforce jobID
```

If instead you want PBS to requeue the job, and have it immediately eligible to run again, use the "`-Wforce`" option to qrerun:

```
% qrerun -Wforce jobID
```

## 13.6 Non Delivery of Output

If the output of a job cannot be delivered to the user, it is saved in a special directory: `PBS_HOME/undelivered` and mail is sent to the user. The typical causes of non-delivery are:

1. The destination host is not trusted and the user does not have a `.rhosts` file.
2. An improper path was specified.
3. A directory in the specified destination path is not writable.
4. The user's `.cshrc` on the destination host generates output when executed.
5. The path specified by **`PBS_SCP`** in `pbs.conf` is incorrect.
6. The `PBS_HOME/spool` directory on the execution host does not have the correct permissions. This directory must have mode 1777 (`drwxrwxrwxt`).

See also the "Delivery of Output Files" section of the **PBS Pro User Guide**.

## 13.7 Job Cannot be Executed

If a user receives a mail message containing a job id and the line "`Job cannot be executed`", the job was aborted by MOM when she tried to place it into execution. The complete reason can be found in one of two places, MOM's log file or the standard error file of the user's job.

If the second line of the message is "`See Administrator for help`", then MOM aborted the job before the job's files were set up. The reason will be noted in MOM's log. Typical reasons are a bad user/group account, checkpoint/restart file (Cray or SGI), or a system error.

If the second line of the message is "`See job standard error file`", then MOM had created the job's file and additional messages were written to standard error. This is typically the result of a bad resource request.

## 13.8 Running Jobs with No Active Processes

On very rare occasions, PBS may be in a situation where a job is in the Running state but has no active processes. This should never happen as the death of the job’s shell should trigger MOM to notify the Server that the job exited and end of job processing should begin. If this situation is noted, PBS offers a way out. Use the `qsig` command to send `SIGNULL`, signal 0, to the job. (Usage of the `qsig` command is provided in the **PBS Pro User Guide**.) If MOM finds there are no processes then she will force the job into the exiting state.

## 13.9 Windows: Qstat errors

If the `qstat` command produces an error such as:

```
     illegally formed job identifier.
```

This means that the DNS lookup is not working properly, or reverse lookup is failing.

If however, `qstat` reports "No Permission", then check `pbs.conf`, and look for the entry "PBS_EXEC". `qstat` (in fact all the PBS commands) will execute the command “ *`PBS_EXEC`*`\sbin\pbs_iff`” to do its authentication. Ensure that the path specified in `pbs.conf` is correct.

## 13.10 Windows: Qsub errors

If, when attempting to submit a job to a remote server, qsub reports:

```
     BAD uid for job execution
```

Then you need to add an entry in the remote system's `.rhosts` or `hosts.equiv` pointing to your windows 2000 machine. Be sure to put all hostnames that resolve to your machine. See also section 10.6.5 “User Authorization” on page 176.

If remote account maps to an Administrator-type account, then you need to set up a `.rhosts` entry, and the remote server must carry the account on its `acl_roots` list.

## 13.11 Windows: Server Reports Error 10035

If Server is not able to contact the Scheduler running on the same local host, it may print to its log file the error message,

```
10035 (Resources Temporarily Unavailable)
```

This is often caused by the local hostname is resolving to a bad IP address. Perhaps, in `%WINDIR%\system32\drivers\etc\hosts`, localhost and *hostname* were mapped to 127.0.0.1.

## 13.12 Windows: PBS Permission Errors

If the Server, MOM, or Scheduler fails to start up because of permission problems on some of its configuration files like `pbs_environment`, `server_priv/nodes`, `mom_priv/config`, then correct the permission by running:

```
pbs_mkdirs server
pbs_mkdirs mom
pbs_mkdirs sched
```

## 13.13 Getting Help

If the material in the PBS manuals is unable to help you solve a particular problem, you may need to contact the PBS Support Team for assistance. First, be sure to check the Customer Login area of the PBS Pro website, which has a number of ways to assist you in resolving problems with PBS, such as the Tips & Advice page.

The PBS Pro support team can also be reached directly via email and phone (contact information on the inside front cover of this manual).

**Important:** When contacting PBS Pro Support, please provide as much of the following information as possible:

**PBS SiteID**
Output of the following commands:

```
qstat -Bf
qstat -Qf
pbsnodes -a
```

If the question pertains to a certain type of job, include:

```
    qstat -f job_id
```

If the question is about scheduling, also send your:

`(PBS_HOME)/sched_priv/sched_config` file.

To expand, renew, or change your PBS support contract, contact our Sales Department. (See contact information on the inside front cover of this manual.)

# Appendix A: Error Codes

The following table lists all the PBS error codes, their textual names, and a description of each.

| Error Name | Error Code | Description |
|---|---|---|
| PBSE_NONE | 0 | No error |
| PBSE_UNKJOBID | 15001 | Unknown Job Identifier |
| PBSE_NOATTR | 15002 | Undefined Attribute |
| PBSE_ATTRRO | 15003 | Attempt to set READ ONLY attribute |
| PBSE_IVALREQ | 15004 | Invalid request |
| PBSE_UNKREQ | 15005 | Unknown batch request |
| PBSE_TOOMANY | 15006 | Too many submit retries |
| PBSE_PERM | 15007 | No permission |
| PBSE_BADHOST | 15008 | Access from host not allowed |
| PBSE_JOBEXIST | 15009 | Job already exists |
| PBSE_SYSTEM | 15010 | System error occurred |
| PBSE_INTERNAL | 15011 | Internal Server error occurred |

| Error Name | Error Code | Description |
|---|---|---|
| PBSE_REGROUTE | 15012 | Parent job of dependent in route queue |
| PBSE_UNKSIG | 15013 | Unknown signal name |
| PBSE_BADATVAL | 15014 | Bad attribute value |
| PBSE_MODATRRUN | 15015 | Cannot modify attrib in run state |
| PBSE_BADSTATE | 15016 | Request invalid for job state |
| PBSE_UNKQUE | 15018 | Unknown queue name |
| PBSE_BADCRED | 15019 | Invalid Credential in request |
| PBSE_EXPIRED | 15020 | Expired Credential in request |
| PBSE_QUNOENB | 15021 | Queue not enabled |
| PBSE_QACESS | 15022 | No access permission for queue |
| PBSE_BADUSER | 15023 | Missing userID, username, or GID. |
| PBSE_HOPCOUNT | 15024 | Max hop count exceeded |
| PBSE_QUEEXIST | 15025 | Queue already exists |
| PBSE_ATTRTYPE | 15026 | Incompatible queue attribute type |
| PBSE_OBJBUSY | 15027 | Object Busy |
| PBSE_QUENBIG | 15028 | Queue name too long |
| PBSE_NOSUP | 15029 | Feature/function not supported |
| PBSE_QUENOEN | 15030 | Cannot enable queue, needs add def |
| PBSE_PROTOCOL | 15031 | Protocol (ASN.1) error |
| PBSE_BADATLST | 15032 | Bad attribute list structure |
| PBSE_NOCONNECTS | 15033 | No free connections |
| PBSE_NOSERVER | 15034 | No Server to connect to |
| PBSE_UNKRESC | 15035 | Unknown resource |
| PBSE_EXCQRESC | 15036 | Job exceeds Queue resource limits |
| PBSE_QUENODFLT | 15037 | No Default Queue Defined |

| Error Name | Error Code | Description |
|---|---|---|
| PBSE_NORERUN | 15038 | Job Not Rerunnable |
| PBSE_ROUTEREJ | 15039 | Route rejected by all destinations |
| PBSE_ROUTEEXPD | 15040 | Time in Route Queue Expired |
| PBSE_MOMREJECT | 15041 | Request to MOM failed |
| PBSE_BADSCRIPT | 15042 | (qsub) Cannot access script file |
| PBSE_STAGEIN | 15043 | Stage In of files failed |
| PBSE_RESCUNAV | 15044 | Resources temporarily unavailable |
| PBSE_BADGRP | 15045 | Bad Group specified |
| PBSE_MAXQUED | 15046 | Max number of jobs in queue |
| PBSE_CKPBSY | 15047 | Checkpoint Busy, may be retries |
| PBSE_EXLIMIT | 15048 | Limit exceeds allowable |
| PBSE_BADACCT | 15049 | Bad Account attribute value |
| PBSE_ALRDYEXIT | 15050 | Job already in exit state |
| PBSE_NOCOPYFILE | 15051 | Job files not copied |
| PBSE_CLEANEDOUT | 15052 | Unknown job id after clean init |
| PBSE_NOSYNCMSTR | 15053 | No Master in Sync Set |
| PBSE_BADDEPEND | 15054 | Invalid dependency |
| PBSE_DUPLIST | 15055 | Duplicate entry in List |
| PBSE_DISPROTO | 15056 | Bad DIS based Request Protocol |
| PBSE_EXECTHERE | 15057 | Cannot execute there |
| PBSE_SISREJECT | 15058 | Sister rejected |
| PBSE_SISCOMM | 15059 | Sister could not communicate |
| PBSE_SVRDOWN | 15060 | Request rejected -server shutting down |
| PBSE_CKPSHORT | 15061 | Not all tasks could checkpoint |

| Error Name | Error Code | Description |
|---|---|---|
| PBSE_UNKNODE | 15062 | Named node is not in the list |
| PBSE_UNKNODEATR | 15063 | Node-attribute not recognized |
| PBSE_NONODES | 15064 | Server has no node list |
| PBSE_NODENBIG | 15065 | Node name is too big |
| PBSE_NODEEXIST | 15066 | Node name already exists |
| PBSE_BADNDATVAL | 15067 | Bad node-attribute value |
| PBSE_MUTUALEX | 15068 | State values are mutually exclusive |
| PBSE_GMODERR | 15069 | Error(s) during global mod of nodes |
| PBSE_NORELYMOM | 15070 | Could not contact MOM |
| PBSE_NOTSNODE | 15071 | No time-shared nodes |
| PBSE_RESV_NO_ WALLTIME | 15075 | Job reserv lacking walltime |
| PBSE_JOBNOTRESV | 15076 | Not a reservation job |
| PBSE_TOOLATE | 15077 | Too late for job reservation |
| PBSE_IRESVE | 15078 | Internal reservation-system error |
| PBSE_UNKRESVTYPE | 15079 | Unknown reservation type |
| PBSE_RESVEXIST | 15080 | Reservation already exists |
| PBSE_resvFail | 15081 | Reservation failed |
| PBSE_genBatchReq | 15082 | Batch request generation failed |
| PBSE_mgrBatchReq | 15083 | qmgr batch request failed |
| PBSE_UNKRESVID | 15084 | Unknown reservation ID |
| PBSE_delProgress | 15085 | Delete already in progress |
| PBSE_BADTSPEC | 15086 | Bad time specification(s) |
| PBSE_RESVMSG | 15087 | So reply_text can return a msg |
| PBSE_NOTRESV | 15088 | Not a reservation |

| Error Name | Error Code | Description |
|---|---|---|
| PBSE_BADNODESPEC | 15089 | Node(s) specification error |
| PBSE_LICENSECPU | 15090 | Licensed CPUs exceeded |
| PBSE_LICENSEINV | 15091 | License is invalid |
| PBSE_RESVAUTH_H | 15092 | Host not authorized to make AR |
| PBSE_RESVAUTH_G | 15093 | Group not authorized to make AR |
| PBSE_RESVAUTH_U | 15094 | User not authorized to make AR |
| PBSE_R_UID | 15095 | Bad effective UID for reservation |
| PBSE_R_GID | 15096 | Bad effective GID for reservation |
| PBSE_IBMSPSWITCH | 15097 | IBM SP Switch error |
| PBSE_LICENSEUNAV | 15098 | Floating License unavailable |
|  | 15099 | UNUSED |
| PBSE_RESCNOTSTR | 15100 | Resource is not of type string |
| Resource monitor specific error codes |  |  |
| PBSE_RMUNKNOWN | 15201 | Resource unknown |
| PBSE_RMBADPARAM | 15202 | Parameter could not be used |
| PBSE_RMNOPARAM | 15203 | A needed parameter did not exist |
| PBSE_RMEXIST | 15204 | Something specified didn't exist |
| PBSE_RMSYSTEM | 15205 | A system error occurred |
| PBSE_RMPART | 15206 | Only part of reservation made |
| RM_ERR_UNKNOWN | PBSE_RMUNKNOWN |  |
| RM_ERR_BADPARAM | PBSE_RMBADPARAM |  |
| RM_ERR_NOPARAM | PBSE_RMNOPARAM |  |
| RM_ERR_EXIST | PBSE_RMEXIST |  |
| RM_ERR_SYSTEM | PBSE_RMSYSTEM |  |

# Appendix B: Request Codes

When reading the PBS event logfiles, you may see messages of the form “Type 19 request received from PBS_Server...”. These “type codes” correspond to different PBS batch requests. The following table lists all the PBS type codes and the corresponding request of each.

| | |
|---|---|
| 0 | PBS_BATCH_Connect |
| 1 | PBS_BATCH_QueueJob |
| 2 | UNUSED |
| 3 | PBS_BATCH_jobscript |
| 4 | PBS_BATCH_RdytoCommit |
| 5 | PBS_BATCH_Commit |
| 6 | PBS_BATCH_DeleteJob |
| 7 | PBS_BATCH_HoldJob |
| 8 | PBS_BATCH_LocateJob |
| 9 | PBS_BATCH_Manager |
| 10 | PBS_BATCH_MessJob |
| 11 | PBS_BATCH_ModifyJob |

| | |
|---|---|
| 12 | PBS_BATCH_MoveJob |
| 13 | PBS_BATCH_ReleaseJob |
| 14 | PBS_BATCH_Rerun |
| 15 | PBS_BATCH_RunJob |
| 16 | PBS_BATCH_SelectJobs |
| 17 | PBS_BATCH_Shutdown |
| 18 | PBS_BATCH_SignalJob |
| 19 | PBS_BATCH_StatusJob |
| 20 | PBS_BATCH_StatusQue |
| 21 | PBS_BATCH_StatusSvr |
| 22 | PBS_BATCH_TrackJob |
| 23 | PBS_BATCH_AsyrunJob |
| 24 | PBS_BATCH_Rescq |
| 25 | PBS_BATCH_ReserveResc |
| 26 | PBS_BATCH_ReleaseResc |
| 27 | PBS_BATCH_FailOver |
| 48 | PBS_BATCH_StageIn |
| 49 | PBS_BATCH_AuthenUser |
| 50 | PBS_BATCH_OrderJob |
| 51 | PBS_BATCH_SelStat |
| 52 | PBS_BATCH_RegistDep |
| 54 | PBS_BATCH_CopyFiles |
| 55 | PBS_BATCH_DelFiles |
| 56 | PBS_BATCH_JobObit |
| 57 | PBS_BATCH_MvJobFile |
| 58 | PBS_BATCH_StatusNode |

| | |
|---|---|
| 59 | PBS_BATCH_Disconnect |
| 60 | UNUSED |
| 61 | UNUSED |
| 62 | PBS_BATCH_JobCred |
| 63 | PBS_BATCH_CopyFiles_Cred |
| 64 | PBS_BATCH_DelFiles_Cred |
| 65 | PBS_BATCH_GSS_Context |
| 66 | UNUSED |
| 67 | UNUSED |
| 68 | UNUSED |
| 69 | UNUSED |
| 70 | PBS_BATCH_SubmitResv |
| 71 | PBS_BATCH_StatusResv |
| 72 | PBS_BATCH_DeleteResv |

# Appendix C: File Listing

The following table lists all the PBS files and directories.

| Directory / File | Owner | Permission | Average Size |
|---|---|---|---|
| *PBS_HOME* | root | drwxr-xr-x | 4096 |
| *PBS_HOME*/pbs_environment | root | -rw-r--r-- | 0 |
| *PBS_HOME*/server_logs | root | drwxr-xr-x | 4096 |
| *PBS_HOME*/spool | root | drwxrwxrwt | 4096 |
| *PBS_HOME*/server_priv | root | drwxr-xr-x | 4096 |
| *PBS_HOME*/server_priv/accounting | root | drwxr-xr-x | 4096 |
| *PBS_HOME*/server_priv/acl_groups | root | drwxr-x--- | 4096 |
| *PBS_HOME*/server_priv/acl_hosts | root | drwxr-x--- | 4096 |
| *PBS_HOME*/server_priv/acl_svr | root | drwxr-x--- | 4096 |
| *PBS_HOME*/server_priv/acl_svr/managers | root | -rw------- | 13 |
| *PBS_HOME*/server_priv/acl_users | root | drwxr-x--- | 4096 |
| *PBS_HOME*/server_priv/jobs | root | drwxr-x--- | 4096 |

| Directory / File | Owner | Permission | Average Size |
|---|---|---|---|
| *PBS_HOME*/server_priv/queues | root | drwxr-x--- | 4096 |
| *PBS_HOME*/server_priv/queues/workq | root | -rw------- | 303 |
| *PBS_HOME*/server_priv/queues/newqueue | root | -rw------- | 303 |
| *PBS_HOME*/server_priv/resvs | root | drwxr-x--- | 4096 |
| *PBS_HOME*/server_priv/nodes | root | -rw-r--r-- | 59 |
| *PBS_HOME*/server_priv/server.lock | root | -rw------- | 4 |
| *PBS_HOME*/server_priv/tracking | root | -rw------- | 0 |
| *PBS_HOME*/server_priv/serverdb | root | -rw------- | 876 |
| *PBS_HOME*/server_priv/license_file | root | -rw-r--r-- | 34 |
| *PBS_HOME*/aux | root | drwxr-xr-x | 4096 |
| *PBS_HOME*/checkpoint | root | drwx------ | 4096 |
| *PBS_HOME*/mom_logs | root | drwxr-xr-x | 4096 |
| *PBS_HOME*/mom_priv | root | drwxr-x--x | 4096 |
| *PBS_HOME*/mom_priv/jobs | root | drwxr-x--x | 4096 |
| *PBS_HOME*/mom_priv/config | root | -rw-r--r-- | 18 |
| *PBS_HOME*/mom_priv/mom.lock | root | -rw-r--r-- | 4 |
| *PBS_HOME*/undelivered | root | drwxrwxrwt | 4096 |
| *PBS_HOME*/sched_logs | root | drwxr-xr-x | 4096 |
| *PBS_HOME*/sched_priv | root | drwxr-x--- | 4096 |
| *PBS_HOME*/sched_priv/dedicated_time | root | -rw-r--r-- | 557 |
| *PBS_HOME*/sched_priv/holidays | root | -rw-r--r-- | 1228 |
| *PBS_HOME*/sched_priv/sched_config | root | -rw-r--r-- | 6370 |
| *PBS_HOME*/sched_priv/resource_group | root | -rw-r--r-- | 0 |
| *PBS_HOME*/sched_priv/sched.lock | root | -rw-r--r-- | 4 |

| **Directory / File** | **Owner** | **Permission** | **Average Size** |
|---|---|---|---|
| *PBS_HOME*/sched_priv/sched_out | root | -rw-r--r-- | 0 |
| *PBS_EXEC*/ | root | drwxr-xr-x | 4096 |
| *PBS_EXEC*/bin | root | drwxr-xr-x | 4096 |
| *PBS_EXEC*/bin/nqs2pbs | root | -rwxr-xr-x | 16062 |
| *PBS_EXEC*/bin/pbs_hostid | root | -rwxr-xr-x | 35604 |
| *PBS_EXEC*/bin/pbs_hostn | root | -rwxr-xr-x | 35493 |
| *PBS_EXEC*/bin/pbs_rdel | root | -rwxr-xr-x | 151973 |
| *PBS_EXEC*/bin/pbs_rstat | root | -rwxr-xr-x | 156884 |
| *PBS_EXEC*/bin/pbs_rsub | root | -rwxr-xr-x | 167446 |
| *PBS_EXEC*/bin/pbs_tclsh | root | -rwxr-xr-x | 857552 |
| *PBS_EXEC*/bin/pbs_wish | root | -rwxr-xr-x | 1592236 |
| *PBS_EXEC*/bin/pbsdsh | root | -rwxr-xr-x | 111837 |
| *PBS_EXEC*/bin/pbsnodes | root | -rwxr-xr-x | 153004 |
| *PBS_EXEC*/bin/printjob | root | -rwxr-xr-x | 42667 |
| *PBS_EXEC*/bin/qalter | root | -rwxr-xr-x | 210723 |
| *PBS_EXEC*/bin/qdel | root | -rwxr-xr-x | 164949 |
| *PBS_EXEC*/bin/qdisable | root | -rwxr-xr-x | 139559 |
| *PBS_EXEC*/bin/qenable | root | -rwxr-xr-x | 139558 |
| *PBS_EXEC*/bin/qhold | root | -rwxr-xr-x | 165368 |
| *PBS_EXEC*/bin/qmgr | root | -rwxr-xr-x | 202526 |
| *PBS_EXEC*/bin/qmove | root | -rwxr-xr-x | 160932 |
| *PBS_EXEC*/bin/qmsg | root | -rwxr-xr-x | 160408 |
| *PBS_EXEC*/bin/qorder | root | -rwxr-xr-x | 146393 |

| Directory / File | Owner | Permission | Average Size |
|---|---|---|---|
| *PBS_EXEC*/bin/qrerun | root | -rwxr-xr-x | 157228 |
| *PBS_EXEC*/bin/qrls | root | -rwxr-xr-x | 165361 |
| *PBS_EXEC*/bin/qrun | root | -rwxr-xr-x | 160978 |
| *PBS_EXEC*/bin/qselect | root | -rwxr-xr-x | 163266 |
| *PBS_EXEC*/bin/qsig | root | -rwxr-xr-x | 160083 |
| *PBS_EXEC*/bin/qstart | root | -rwxr-xr-x | 139589 |
| *PBS_EXEC*/bin/qstat | root | -rwxr-xr-x | 207532 |
| *PBS_EXEC*/bin/qstop | root | -rwxr-xr-x | 139584 |
| *PBS_EXEC*/bin/qsub | root | -rwxr-xr-x | 275460 |
| *PBS_EXEC*/bin/qterm | root | -rwxr-xr-x | 132188 |
| *PBS_EXEC*/bin/tracejob | root | -rwxr-xr-x | 64730 |
| *PBS_EXEC*/bin/xpbs | root | -rwxr-xr-x | 817 |
| *PBS_EXEC*/bin/xpbsmon | root | -rwxr-xr-x | 817 |
| *PBS_EXEC*/etc | root | drwxr-xr-x | 4096 |
| *PBS_EXEC*/etc/pbs_dedicated | root | -rw-r--r-- | 557 |
| *PBS_EXEC*/etc/pbs_holidays | root | -rw-r--r-- | 1173 |
| *PBS_EXEC*/etc/pbs_init.d | root | -rwx------ | 5382 |
| *PBS_EXEC*/etc/pbs_postinstall | root | -rwx------ | 10059 |
| *PBS_EXEC*/etc/pbs_resource_group | root | -rw-r--r-- | 657 |
| *PBS_EXEC*/etc/pbs_sched_config | root | -rw-r--r-- | 9791 |
| *PBS_EXEC*/etc/pbs_setlicense | root | -rwx------ | 2118 |
| *PBS_EXEC*/include | root | drwxr-xr-x | 4096 |
| *PBS_EXEC*/include/pbs_error.h | root | -r--r--r-- | 7543 |
| *PBS_EXEC*/include/pbs_ifl.h | root | -r--r--r-- | 17424 |

| **Directory / File** | **Owner** | **Permission** | **Average Size** |
|---|---|---|---|
| *PBS_EXEC*/include/rm.h | root | -r--r--r-- | 740 |
| *PBS_EXEC*/include/tm.h | root | -r--r--r-- | 2518 |
| *PBS_EXEC*/include/tm_.h | root | -r--r--r-- | 2236 |
| *PBS_EXEC*/lib | root | drwxr-xr-x | 4096 |
| *PBS_EXEC*/lib/libattr.a | root | -rw-r--r-- | 390274 |
| *PBS_EXEC*/lib/libcmds.a | root | -rw-r--r-- | 328234 |
| *PBS_EXEC*/lib/liblog.a | root | -rw-r--r-- | 101230 |
| *PBS_EXEC*/lib/libnet.a | root | -rw-r--r-- | 145968 |
| *PBS_EXEC*/lib/libpbs.a | root | -rw-r--r-- | 1815486 |
| *PBS_EXEC*/lib/libsite.a | root | -rw-r--r-- | 132906 |
| *PBS_EXEC*/lib/pbs_sched.a | root | -rw-r--r-- | 822026 |
| *PBS_EXEC*/lib/pm | root | drwxr--r-- | 4096 |
| *PBS_EXEC*/lib/pm/PBS.pm | root | -rw-r--r-- | 3908 |
| *PBS_EXEC*/lib/xpbs | root | drwxr-xr-x | 4096 |
| *PBS_EXEC*/lib/xpbs/pbs_acctname.tk | root | -rw-r--r-- | 3484 |
| *PBS_EXEC*/lib/xpbs/pbs_after_depend.tk | root | -rw-r--r-- | 8637 |
| *PBS_EXEC*/lib/xpbs/pbs_auto_upd.tk | root | -rw-r--r-- | 3384 |
| *PBS_EXEC*/lib/xpbs/pbs_before_depend.tk | root | -rw-r--r-- | 8034 |
| *PBS_EXEC*/lib/xpbs/pbs_bin | root | drwxr-xr-x | 4096 |
| *PBS_EXEC*/lib/xpbs/pbs_bin/xpbs_datadump | root | -rwxr-xr-x | 190477 |
| *PBS_EXEC*/lib/xpbs/pbs_bin/xpbs_scriptload | root | -rwxr-xr-x | 173176 |
| *PBS_EXEC*/lib/xpbs/pbs_bindings.tk | root | -rw-r--r-- | 26029 |
| *PBS_EXEC*/lib/xpbs/pbs_bitmaps | root | drwxr-xr-x | 4096 |

| Directory / File | Owner | Permission | Average Size |
|---|---|---|---|
| *PBS_EXEC*/lib/xpbs/pbs_bitmaps/Downarrow.bmp | root | -rw-r--r-- | 299 |
| *PBS_EXEC*/lib/xpbs/pbs_bitmaps/Uparrow.bmp | root | -rw-r--r-- | 293 |
| *PBS_EXEC*/lib/xpbs/pbs_bitmaps/curve_down_arrow.bmp | root | -rw-r--r-- | 320 |
| *PBS_EXEC*/lib/xpbs/pbs_bitmaps/curve_up_arrow.bmp | root | -rw-r--r-- | 314 |
| *PBS_EXEC*/lib/xpbs/pbs_bitmaps/cyclist-only.xbm | root | -rw-r--r-- | 2485 |
| *PBS_EXEC*/lib/xpbs/pbs_bitmaps/hourglass.bmp | root | -rw-r--r-- | 557 |
| *PBS_EXEC*/lib/xpbs/pbs_bitmaps/iconize.bmp | root | -rw-r--r-- | 287 |
| *PBS_EXEC*/lib/xpbs/pbs_bitmaps/logo.bmp | root | -rw-r--r-- | 67243 |
| *PBS_EXEC*/lib/xpbs/pbs_bitmaps/maximize.bmp | root | -rw-r--r-- | 287 |
| *PBS_EXEC*/lib/xpbs/pbs_bitmaps/sm_down_arrow.bmp | root | -rw-r--r-- | 311 |
| *PBS_EXEC*/lib/xpbs/pbs_bitmaps/sm_up_arrow.bmp | root | -rw-r--r-- | 305 |
| *PBS_EXEC*/lib/xpbs/pbs_box.tk | root | -rw-r--r-- | 25912 |
| *PBS_EXEC*/lib/xpbs/pbs_button.tk | root | -rw-r--r-- | 18795 |
| *PBS_EXEC*/lib/xpbs/pbs_checkpoint.tk | root | -rw-r--r-- | 6892 |
| *PBS_EXEC*/lib/xpbs/pbs_common.tk | root | -rw-r--r-- | 25940 |
| *PBS_EXEC*/lib/xpbs/pbs_concur.tk | root | -rw-r--r-- | 8445 |
| *PBS_EXEC*/lib/xpbs/pbs_datetime.tk | root | -rw-r--r-- | 4533 |
| *PBS_EXEC*/lib/xpbs/pbs_email_list.tk | root | -rw-r--r-- | 3094 |
| *PBS_EXEC*/lib/xpbs/pbs_entry.tk | root | -rw-r--r-- | 12389 |
| *PBS_EXEC*/lib/xpbs/pbs_fileselect.tk | root | -rw-r--r-- | 7975 |
| *PBS_EXEC*/lib/xpbs/pbs_help | root | drwxr-xr-x | 4096 |
| *PBS_EXEC*/lib/xpbs/pbs_help/after_depend.hlp | root | -rw-r--r-- | 1746 |

| **Directory / File** | **Owner** | **Permission** | **Average Size** |
|---|---|---|---|
| *PBS_EXEC*/lib/xpbs/pbs_help/auto_update.hlp | root | -rw-r--r-- | 776 |
| *PBS_EXEC*/lib/xpbs/pbs_help/before_depend.hlp | root | -rw-r--r-- | 1413 |
| *PBS_EXEC*/lib/xpbs/pbs_help/concur.hlp | root | -rw-r--r-- | 1383 |
| *PBS_EXEC*/lib/xpbs/pbs_help/datetime.hlp | root | -rw-r--r-- | 698 |
| *PBS_EXEC*/lib/xpbs/pbs_help/delete.hlp | root | -rw-r--r-- | 632 |
| *PBS_EXEC*/lib/xpbs/pbs_help/email.hlp | root | -rw-r--r-- | 986 |
| *PBS_EXEC*/lib/xpbs/pbs_help/fileselect.hlp | root | -rw-r--r-- | 1655 |
| *PBS_EXEC*/lib/xpbs/pbs_help/hold.hlp | root | -rw-r--r-- | 538 |
| *PBS_EXEC*/lib/xpbs/pbs_help/main.hlp | root | -rw-r--r-- | 15220 |
| *PBS_EXEC*/lib/xpbs/pbs_help/message.hlp | root | -rw-r--r-- | 677 |
| *PBS_EXEC*/lib/xpbs/pbs_help/misc.hlp | root | -rw-r--r-- | 4194 |
| *PBS_EXEC*/lib/xpbs/pbs_help/modify.hlp | root | -rw-r--r-- | 6034 |
| *PBS_EXEC*/lib/xpbs/pbs_help/move.hlp | root | -rw-r--r-- | 705 |
| *PBS_EXEC*/lib/xpbs/pbs_help/notes.hlp | root | -rw-r--r-- | 3724 |
| *PBS_EXEC*/lib/xpbs/pbs_help/preferences.hlp | root | -rw-r--r-- | 1645 |
| *PBS_EXEC*/lib/xpbs/pbs_help/release.hlp | root | -rw-r--r-- | 573 |
| *PBS_EXEC*/lib/xpbs/pbs_help/select.acctname.hlp | root | -rw-r--r-- | 609 |
| *PBS_EXEC*/lib/xpbs/pbs_help/select.checkpoint.hlp | root | -rw-r--r-- | 1133 |
| *PBS_EXEC*/lib/xpbs/pbs_help/select.hold.hlp | root | -rw-r--r-- | 544 |
| *PBS_EXEC*/lib/xpbs/pbs_help/select.jobname.hlp | root | -rw-r--r-- | 600 |
| *PBS_EXEC*/lib/xpbs/pbs_help/select.owners.hlp | root | -rw-r--r-- | 1197 |
| *PBS_EXEC*/lib/xpbs/pbs_help/select.priority.hlp | root | -rw-r--r-- | 748 |
| *PBS_EXEC*/lib/xpbs/pbs_help/select.qtime.hlp | root | -rw-r--r-- | 966 |

| Directory / File | Owner | Permission | Average Size |
|---|---|---|---|
| *PBS_EXEC*/lib/xpbs/pbs_help/select.rerun.hlp | root | -rw-r--r-- | 541 |
| *PBS_EXEC*/lib/xpbs/pbs_help/select.resources.hlp | root | -rw-r--r-- | 1490 |
| *PBS_EXEC*/lib/xpbs/pbs_help/select.states.hlp | root | -rw-r--r-- | 562 |
| *PBS_EXEC*/lib/xpbs/pbs_help/signal.hlp | root | -rw-r--r-- | 675 |
| *PBS_EXEC*/lib/xpbs/pbs_help/staging.hlp | root | -rw-r--r-- | 3702 |
| *PBS_EXEC*/lib/xpbs/pbs_help/submit.hlp | root | -rw-r--r-- | 9721 |
| *PBS_EXEC*/lib/xpbs/pbs_help/terminate.hlp | root | -rw-r--r-- | 635 |
| *PBS_EXEC*/lib/xpbs/pbs_help/trackjob.hlp | root | -rw-r--r-- | 2978 |
| *PBS_EXEC*/lib/xpbs/pbs_hold.tk | root | -rw-r--r-- | 3539 |
| *PBS_EXEC*/lib/xpbs/pbs_jobname.tk | root | -rw-r--r-- | 3375 |
| *PBS_EXEC*/lib/xpbs/pbs_listbox.tk | root | -rw-r--r-- | 10544 |
| *PBS_EXEC*/lib/xpbs/pbs_main.tk | root | -rw-r--r-- | 24147 |
| *PBS_EXEC*/lib/xpbs/pbs_misc.tk | root | -rw-r--r-- | 14526 |
| *PBS_EXEC*/lib/xpbs/pbs_owners.tk | root | -rw-r--r-- | 4509 |
| *PBS_EXEC*/lib/xpbs/pbs_pbs.tcl | root | -rw-r--r-- | 52524 |
| *PBS_EXEC*/lib/xpbs/pbs_pref.tk | root | -rw-r--r-- | 3445 |
| *PBS_EXEC*/lib/xpbs/pbs_preferences.tcl | root | -rw-r--r-- | 4323 |
| *PBS_EXEC*/lib/xpbs/pbs_prefsave.tk | root | -rw-r--r-- | 1378 |
| *PBS_EXEC*/lib/xpbs/pbs_priority.tk | root | -rw-r--r-- | 4434 |
| *PBS_EXEC*/lib/xpbs/pbs_qalter.tk | root | -rw-r--r-- | 35003 |
| *PBS_EXEC*/lib/xpbs/pbs_qdel.tk | root | -rw-r--r-- | 3175 |
| *PBS_EXEC*/lib/xpbs/pbs_qhold.tk | root | -rw-r--r-- | 3676 |
| *PBS_EXEC*/lib/xpbs/pbs_qmove.tk | root | -rw-r--r-- | 3326 |
| *PBS_EXEC*/lib/xpbs/pbs_qmsg.tk | root | -rw-r--r-- | 4032 |

| **Directory / File** | **Owner** | **Permission** | **Average Size** |
|---|---|---|---|
| *PBS_EXEC*/lib/xpbs/pbs_qrls.tk | root | -rw-r--r-- | 3674 |
| *PBS_EXEC*/lib/xpbs/pbs_qsig.tk | root | -rw-r--r-- | 5171 |
| *PBS_EXEC*/lib/xpbs/pbs_qsub.tk | root | -rw-r--r-- | 37466 |
| *PBS_EXEC*/lib/xpbs/pbs_qterm.tk | root | -rw-r--r-- | 3204 |
| *PBS_EXEC*/lib/xpbs/pbs_qtime.tk | root | -rw-r--r-- | 5790 |
| *PBS_EXEC*/lib/xpbs/pbs_rerun.tk | root | -rw-r--r-- | 2802 |
| *PBS_EXEC*/lib/xpbs/pbs_res.tk | root | -rw-r--r-- | 4807 |
| *PBS_EXEC*/lib/xpbs/pbs_spinbox.tk | root | -rw-r--r-- | 7144 |
| *PBS_EXEC*/lib/xpbs/pbs_staging.tk | root | -rw-r--r-- | 12183 |
| *PBS_EXEC*/lib/xpbs/pbs_state.tk | root | -rw-r--r-- | 3657 |
| *PBS_EXEC*/lib/xpbs/pbs_text.tk | root | -rw-r--r-- | 2738 |
| *PBS_EXEC*/lib/xpbs/pbs_trackjob.tk | root | -rw-r--r-- | 13605 |
| *PBS_EXEC*/lib/xpbs/pbs_wmgr.tk | root | -rw-r--r-- | 1428 |
| *PBS_EXEC*/lib/xpbs/tclIndex | root | -rw-r--r-- | 19621 |
| *PBS_EXEC*/lib/xpbs/xpbs.src.tk | root | -rwxr-xr-x | 9666 |
| *PBS_EXEC*/lib/xpbs/xpbsrc | root | -rw-r--r-- | 2986 |
| *PBS_EXEC*/lib/xpbsmon | root | drwxr-xr-x | 4096 |
| *PBS_EXEC*/lib/xpbsmon/pbs_auto_upd.tk | root | -rw-r--r-- | 3281 |
| *PBS_EXEC*/lib/xpbsmon/pbs_bindings.tk | root | -rw-r--r-- | 9288 |
| *PBS_EXEC*/lib/xpbsmon/pbs_bitmaps | root | drwxr-xr-x | 4096 |
| *PBS_EXEC*/lib/xpbsmon/pbs_bitmaps/cyclist-only.xbm | root | -rw-r--r-- | 2485 |
| *PBS_EXEC*/lib/xpbsmon/pbs_bitmaps/hour-glass.bmp | root | -rw-r--r-- | 557 |

| Directory / File | Owner | Permission | Average Size |
|---|---|---|---|
| *PBS_EXEC*/lib/xpbsmon/pbs_bitmaps/iconize.bmp | root | -rw-r--r-- | 287 |
| *PBS_EXEC*/lib/xpbsmon/pbs_bitmaps/logo.bmp | root | -rw-r--r-- | 67243 |
| *PBS_EXEC*/lib/xpbsmon/pbs_bitmaps/maximize.bmp | root | -rw-r--r-- | 287 |
| *PBS_EXEC*/lib/xpbsmon/pbs_box.tk | root | -rw-r--r-- | 15607 |
| *PBS_EXEC*/lib/xpbsmon/pbs_button.tk | root | -rw-r--r-- | 7543 |
| *PBS_EXEC*/lib/xpbsmon/pbs_cluster.tk | root | -rw-r--r-- | 44406 |
| *PBS_EXEC*/lib/xpbsmon/pbs_color.tk | root | -rw-r--r-- | 5634 |
| *PBS_EXEC*/lib/xpbsmon/pbs_common.tk | root | -rw-r--r-- | 5716 |
| *PBS_EXEC*/lib/xpbsmon/pbs_dialog.tk | root | -rw-r--r-- | 8398 |
| *PBS_EXEC*/lib/xpbsmon/pbs_entry.tk | root | -rw-r--r-- | 10697 |
| *PBS_EXEC*/lib/xpbsmon/pbs_expr.tk | root | -rw-r--r-- | 6163 |
| *PBS_EXEC*/lib/xpbsmon/pbs_help | root | drwxr-xr-x | 4096 |
| *PBS_EXEC*/lib/xpbsmon/pbs_help/auto_update.hlp | root | -rw-r--r-- | 624 |
| *PBS_EXEC*/lib/xpbsmon/pbs_help/main.hlp | root | -rw-r--r-- | 15718 |
| *PBS_EXEC*/lib/xpbsmon/pbs_help/notes.hlp | root | -rw-r--r-- | 296 |
| *PBS_EXEC*/lib/xpbsmon/pbs_help/pref.hlp | root | -rw-r--r-- | 1712 |
| *PBS_EXEC*/lib/xpbsmon/pbs_help/prefQuery.hlp | root | -rw-r--r-- | 4621 |
| *PBS_EXEC*/lib/xpbsmon/pbs_help/prefServer.hlp | root | -rw-r--r-- | 1409 |
| *PBS_EXEC*/lib/xpbsmon/pbs_listbox.tk | root | -rw-r--r-- | 10640 |
| *PBS_EXEC*/lib/xpbsmon/pbs_main.tk | root | -rw-r--r-- | 6760 |
| *PBS_EXEC*/lib/xpbsmon/pbs_node.tk | root | -rw-r--r-- | 60640 |
| *PBS_EXEC*/lib/xpbsmon/pbs_pbs.tk | root | -rw-r--r-- | 7090 |
| *PBS_EXEC*/lib/xpbsmon/pbs_pref.tk | root | -rw-r--r-- | 22117 |
| *PBS_EXEC*/lib/xpbsmon/pbs_preferences.tcl | root | -rw-r--r-- | 10212 |

| **Directory / File** | **Owner** | **Permission** | **Average Size** |
|---|---|---|---|
| *PBS_EXEC*/lib/xpbsmon/pbs_prefsave.tk | root | -rw-r--r-- | 1482 |
| *PBS_EXEC*/lib/xpbsmon/pbs_spinbox.tk | root | -rw-r--r-- | 7162 |
| *PBS_EXEC*/lib/xpbsmon/pbs_system.tk | root | -rw-r--r-- | 47760 |
| *PBS_EXEC*/lib/xpbsmon/pbs_wmgr.tk | root | -rw-r--r-- | 1140 |
| *PBS_EXEC*/lib/xpbsmon/tclIndex | root | -rw-r--r-- | 30510 |
| *PBS_EXEC*/lib/xpbsmon/xpbsmon.src.tk | root | -rwxr-xr-x | 13999 |
| *PBS_EXEC*/lib/xpbsmon/xpbsmonrc | root | -rw-r--r-- | 3166 |
| *PBS_EXEC*/man | root | drwxr-xr-x | 4096 |
| *PBS_EXEC*/man/man1 | root | drwxr-xr-x | 4096 |
| *PBS_EXEC*/man/man1/nqs2pbs.1B | root | -rw-r--r-- | 3276 |
| *PBS_EXEC*/man/man1/pbs.1B | root | -rw-r--r-- | 5376 |
| *PBS_EXEC*/man/man1/pbs_rdel.1B | root | -rw-r--r-- | 2342 |
| *PBS_EXEC*/man/man1/pbs_rstat.1B | root | -rw-r--r-- | 2682 |
| *PBS_EXEC*/man/man1/pbs_rsub.1B | root | -rw-r--r-- | 9143 |
| *PBS_EXEC*/man/man1/pbsdsh.1B | root | -rw-r--r-- | 2978 |
| *PBS_EXEC*/man/man1/qalter.1B | root | -rw-r--r-- | 21569 |
| *PBS_EXEC*/man/man1/qdel.1B | root | -rw-r--r-- | 3363 |
| *PBS_EXEC*/man/man1/qhold.1B | root | -rw-r--r-- | 4323 |
| *PBS_EXEC*/man/man1/qmove.1B | root | -rw-r--r-- | 3343 |
| *PBS_EXEC*/man/man1/qmsg.1B | root | -rw-r--r-- | 3244 |
| *PBS_EXEC*/man/man1/qorder.1B | root | -rw-r--r-- | 3028 |
| *PBS_EXEC*/man/man1/qrerun.1B | root | -rw-r--r-- | 2965 |
| *PBS_EXEC*/man/man1/qrls.1B | root | -rw-r--r-- | 3927 |

| Directory / File | Owner | Permission | Average Size |
|---|---|---|---|
| *PBS_EXEC*/man/man1/qselect.1B | root | -rw-r--r-- | 12690 |
| *PBS_EXEC*/man/man1/qsig.1B | root | -rw-r--r-- | 3817 |
| *PBS_EXEC*/man/man1/qstat.1B | root | -rw-r--r-- | 15274 |
| *PBS_EXEC*/man/man1/qsub.1B | root | -rw-r--r-- | 36435 |
| *PBS_EXEC*/man/man1/xpbs.1B | root | -rw-r--r-- | 26956 |
| *PBS_EXEC*/man/man1/xpbsmon.1B | root | -rw-r--r-- | 26365 |
| *PBS_EXEC*/man/man3 | root | drwxr-xr-x | 4096 |
| *PBS_EXEC*/man/man3/pbs_alterjob.3B | root | -rw-r--r-- | 5475 |
| *PBS_EXEC*/man/man3/pbs_connect.3B | root | -rw-r--r-- | 3493 |
| *PBS_EXEC*/man/man3/pbs_default.3B | root | -rw-r--r-- | 2150 |
| *PBS_EXEC*/man/man3/pbs_deljob.3B | root | -rw-r--r-- | 3081 |
| *PBS_EXEC*/man/man3/pbs_disconnect.3B | root | -rw-r--r-- | 1985 |
| *PBS_EXEC*/man/man3/pbs_geterrmsg.3B | root | -rw-r--r-- | 2473 |
| *PBS_EXEC*/man/man3/pbs_holdjob.3B | root | -rw-r--r-- | 3006 |
| *PBS_EXEC*/man/man3/pbs_manager.3B | root | -rw-r--r-- | 4337 |
| *PBS_EXEC*/man/man3/pbs_movejob.3B | root | -rw-r--r-- | 3220 |
| *PBS_EXEC*/man/man3/pbs_msgjob.3B | root | -rw-r--r-- | 2912 |
| *PBS_EXEC*/man/man3/pbs_orderjob.3B | root | -rw-r--r-- | 2526 |
| *PBS_EXEC*/man/man3/pbs_rerunjob.3B | root | -rw-r--r-- | 2531 |
| *PBS_EXEC*/man/man3/pbs_rescquery.3B | root | -rw-r--r-- | 5804 |
| *PBS_EXEC*/man/man3/pbs_rescreserve.3B | root | -rw-r--r-- | 4125 |
| *PBS_EXEC*/man/man3/pbs_rlsjob.3B | root | -rw-r--r-- | 3043 |
| *PBS_EXEC*/man/man3/pbs_runjob.3B | root | -rw-r--r-- | 3484 |
| *PBS_EXEC*/man/man3/pbs_selectjob.3B | root | -rw-r--r-- | 7717 |

| **Directory / File** | **Owner** | **Permission** | **Average Size** |
|---|---|---|---|
| *PBS_EXEC*/man/man3/pbs_sigjob.3B | root | -rw-r--r-- | 3108 |
| *PBS_EXEC*/man/man3/pbs_stagein.3B | root | -rw-r--r-- | 3198 |
| *PBS_EXEC*/man/man3/pbs_statjob.3B | root | -rw-r--r-- | 4618 |
| *PBS_EXEC*/man/man3/pbs_statnode.3B | root | -rw-r--r-- | 3925 |
| *PBS_EXEC*/man/man3/pbs_statque.3B | root | -rw-r--r-- | 4009 |
| *PBS_EXEC*/man/man3/pbs_statserver.3B | root | -rw-r--r-- | 3674 |
| *PBS_EXEC*/man/man3/pbs_submit.3B | root | -rw-r--r-- | 6320 |
| *PBS_EXEC*/man/man3/pbs_submitresv.3B | root | -rw-r--r-- | 3878 |
| *PBS_EXEC*/man/man3/pbs_terminate.3B | root | -rw-r--r-- | 3322 |
| *PBS_EXEC*/man/man3/rpp.3B | root | -rw-r--r-- | 6476 |
| *PBS_EXEC*/man/man3/tm.3B | root | -rw-r--r-- | 11062 |
| *PBS_EXEC*/man/man7 | root | drwxr-xr-x | 4096 |
| *PBS_EXEC*/man/man7/pbs_job_attributes.7B | root | -rw-r--r-- | 15920 |
| *PBS_EXEC*/man/man7/pbs_node_attributes.7B | root | -rw-r--r-- | 7973 |
| *PBS_EXEC*/man/man7/pbs_queue_attributes.7B | root | -rw-r--r-- | 11062 |
| *PBS_EXEC*/man/man7/pbs_resources_linux.7B | root | -rw-r--r-- | 8452 |
| *PBS_EXEC*/man/man7/pbs_resv_attributes.7B | root | -rw-r--r-- | 11662 |
| *PBS_EXEC*/man/man7/pbs_server_attributes.7B | root | -rw-r--r-- | 14327 |
| *PBS_EXEC*/man/man8 | root | drwxr-xr-x | 4096 |
| *PBS_EXEC*/man/man8/pbs_idled.8B | root | -rw-r--r-- | 2628 |
| *PBS_EXEC*/man/man8/pbs_mom.8B | root | -rw-r--r-- | 23496 |
| *PBS_EXEC*/man/man8/pbs_mom_globus.8B | root | -rw-r--r-- | 11054 |
| *PBS_EXEC*/man/man8/pbs_sched_cc.8B | root | -rw-r--r-- | 6731 |

| Directory / File | Owner | Permission | Average Size |
|---|---|---|---|
| *PBS_EXEC*/man/man8/pbs_server.8B | root | -rw-r--r-- | 7914 |
| *PBS_EXEC*/man/man8/pbsfs.8B | root | -rw-r--r-- | 3703 |
| *PBS_EXEC*/man/man8/pbsnodes.8B | root | -rw-r--r-- | 3441 |
| *PBS_EXEC*/man/man8/qdisable.8B | root | -rw-r--r-- | 3104 |
| *PBS_EXEC*/man/man8/qenable.8B | root | -rw-r--r-- | 2937 |
| *PBS_EXEC*/man/man8/qmgr.8B | root | -rw-r--r-- | 7282 |
| *PBS_EXEC*/man/man8/qrun.8B | root | -rw-r--r-- | 2850 |
| *PBS_EXEC*/man/man8/qstart.8B | root | -rw-r--r-- | 2966 |
| *PBS_EXEC*/man/man8/qstop.8B | root | -rw-r--r-- | 2963 |
| *PBS_EXEC*/man/man8/qterm.8B | root | -rw-r--r-- | 4839 |
| *PBS_EXEC*/sbin | root | drwxr-xr-x | 4096 |
| *PBS_EXEC*/sbin/pbs-report | root | -rwxr-xr-x | 68296 |
| *PBS_EXEC*/sbin/pbs_demux | root | -rwxr-xr-x | 38688 |
| *PBS_EXEC*/sbin/pbs_idled | root | -rwxr-xr-x | 99373 |
| *PBS_EXEC*/sbin/pbs_iff | root | -rwsr-xr-x | 133142 |
| *PBS_EXEC*/sbin/pbs_mom | root | -rwx------ | 839326 |
| *PBS_EXEC*/sbin/pbs_mom.cpuset | root | -rwx------ | 0 |
| *PBS_EXEC*/sbin/pbs_mom.standard | root | -rwx------ | 0 |
| *PBS_EXEC*/sbin/pbs_rcp | root | -rwsr-xr-x | 75274 |
| *PBS_EXEC*/sbin/pbs_sched | root | -rwx------ | 705478 |
| *PBS_EXEC*/sbin/pbs_server | root | -rwx------ | 1133650 |
| *PBS_EXEC*/sbin/pbsfs | root | -rwxr-xr-x | 663707 |
| *PBS_EXEC*/tcltk | root | drwxr-xr-x | 4096 |
| *PBS_EXEC*/tcltk/bin | root | drwxr-xr-x | 4096 |

| **Directory / File** | **Owner** | **Permission** | **Average Size** |
|---|---|---|---|
| *PBS_EXEC*/tcltk/bin/tclsh8.3 | root | -rw-r--r-- | 552763 |
| *PBS_EXEC*/tcltk/bin/wish8.3 | root | -rw-r--r-- | 1262257 |
| *PBS_EXEC*/tcltk/include | root | drwxr-xr-x | 4096 |
| *PBS_EXEC*/tcltk/include/tcl.h | root | -rw-r--r-- | 57222 |
| *PBS_EXEC*/tcltk/include/tclDecls.h | root | -rw-r--r-- | 123947 |
| *PBS_EXEC*/tcltk/include/tk.h | root | -rw-r--r-- | 47420 |
| *PBS_EXEC*/tcltk/include/tkDecls.h | root | -rw-r--r-- | 80181 |
| *PBS_EXEC*/tcltk/lib | root | drwxr-xr-x | 4096 |
| *PBS_EXEC*/tcltk/lib/libtcl8.3.a | root | -rw-r--r-- | 777558 |
| *PBS_EXEC*/tcltk/lib/libtclstub8.3.a | root | -rw-r--r-- | 1832 |
| *PBS_EXEC*/tcltk/lib/libtk8.3.a | root | -rw-r--r-- | 1021024 |
| *PBS_EXEC*/tcltk/lib/libtkstub8.3.a | root | -rw-r--r-- | 3302 |
| *PBS_EXEC*/tcltk/lib/tcl8.3 | root | drwxr-xr-x | 4096 |
| *PBS_EXEC*/tcltk/lib/tclConfig.sh | root | -rw-r--r-- | 7076 |
| *PBS_EXEC*/tcltk/lib/tk8.3 | root | drwxr-xr-x | 4096 |
| *PBS_EXEC*/tcltk/lib/tkConfig.sh | root | -rw-r--r-- | 3822 |
| *PBS_EXEC*/tcltk/license.terms | root | -rw-r--r-- | 2233 |

# Index