Altair®

PBS Professional™ 9.1

Administrator's Guide

UNIX®, LINUX® and Windows®

PBS Professional™ Administrator's Guide
Altair® PBS Professional™ 9.1, Updated: October 24, 2007
Edited by: Anne Urban

For more information, copies of these books, and for product sales, contact Altair at:

Web: www.altair.com www.pbspro.com
Email: sales@pbspro.com

Technical Support

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	hwsupport@altair.de
India	+91 80 658 8540 +91 80 658 8542	pbs-support@india.altair.com
Italy	+39 0832 315573 +39 800 905595	support@altairtorino.it
Japan	+81 3 5396 1341	pbs@altairjp.co.jp
Korea	+82 31 728 8600	support@altair.co.kr
Scandinavia	+46 (0)46 286 2050	support@altair.se
UK	+44 (0) 2476 323 600	support@uk.altair.com

Table of Contents

Acknowledgements

PBS Professional 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 was 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 continues to provide excellent feedback on the product.

Preface

Intended Audience

This document provides the system administrator with the information required to install, configure, and manage PBS Professional (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 Professional Quick Start Guide: Provides a quick overview of PBS Professional installation and license file generation.

PBS Professional User's Guide: Provides an overview of PBS Professional 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 Professional External Reference Specification: Discusses in detail the PBS application programming interface (API), security within PBS, and intra-component communication.

Ordering Software and Publications

To order additional copies of this manual and other PBS publications, or to purchase additional software licenses, contact your Altair sales representative. 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's Guide** to PBS Professional is intended as your knowledgeable companion to the PBS Professional software. This edition pertains to PBS Professional in general, with specific information for version 9.1.

1.1 Book Organization

This book is organized into 14 chapters, plus 5 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 Professional software and licenses.
Chapter 5	**Licensing**: Describes the Altair FLEXlm licensing scheme, as well as the old trial licenses.
Chapter 6	**Upgrading PBS Professional**: Provides important information for sites that are upgrading from a previous version of PBS.

Chapter 7	**Configuring the Server**: Describes how to configure the PBS Server, and set up queues and vnodes.
Chapter 8	**Configuring MOM**: Describes how to configure the PBS MOM processes.
Chapter 9	**Configuring the Scheduler**: Describes how to configure the PBS Scheduler.
Chapter 10	**Customizing PBS Resources**: Describes how to configure custom resources and dynamic consumable resources.
Chapter 11	**Integration & Administration**: Discusses PBS day-to-day administration and and related activities.
Chapter 12	**Administrator Commands**: Describes all PBS commands intended to be used by the Administrator.
Chapter 13	**Example Configurations**: Provides examples and sample configurations.
Chapter 14	**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 Professional, with owner, permissions, and average size.
Appendix D	**Log Messages**: Explains some PBS log messages.
Appendix C	**License Agreement**: Contains the Altair license agreement.

1.2 Supported Platforms

For a list of supported platforms, see the Release Notes.

1.3 What is PBS Professional?

PBS Professional 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 Professional addresses these problems for computing-intensive enterprises such as science, engineering, finance, and entertainment.
Now you can use the power of PBS Professional 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 Professional can also help you to efficiently manage growth by tracking real usage levels across your systems and by enhancing effective utilization of future purchases.

1.3.1 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, in the year 2000, the Portable Batch System Professional Edition (PBS Professional), 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 Professional and related Grid software. This unit then became part of Altair Engineering, Inc.

1.4 About the PBS Team

The PBS Professional product is being developed by the same team that originally designed PBS for NASA. In addition to the core engineering team, Altair Engineering 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-management 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.

1.5 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

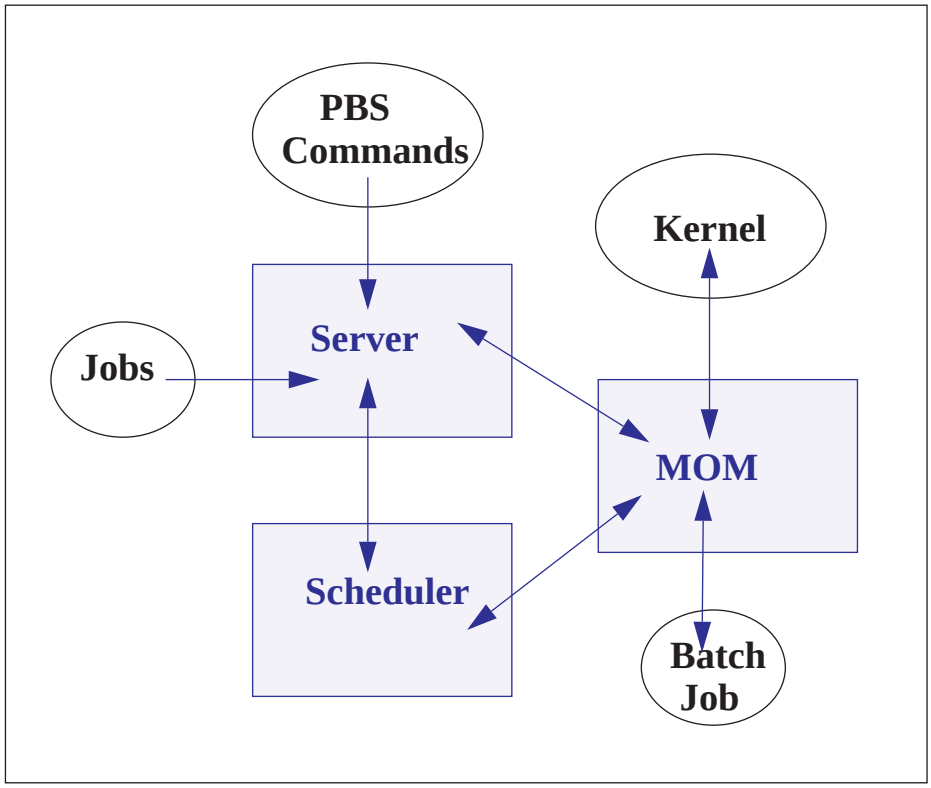
PBS is a distributed workload management system. As such, PBS handles the management and monitoring of the computational workload on a set of one or more computers. Modern workload/resource management solutions like PBS include the features of traditional batch queueing but offer greater flexibility 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 running jobs. Tools are provided to aid human monitoring of the PBS system as well.

2.1 PBS Components

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



Server The *Server* 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 communication with the Server are 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 component that actually places the job into execution. This process, *pbs_mom*, is informally called *MOM* as it is the mother of all executing jobs. 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 runs on each computer which will execute PBS jobs.

Scheduler The *Scheduler*, *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 11.7.7 "External Security" on page 427.

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 No longer used. See vnode. 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 Linux clusters, which contain separate computational units each with their own OS, are collections of nodes. Note that this is usually used to mean *host*.

Vnode A virtual node, or *vnode*, is an abstract object representing a set of resources which form a usable part of a machine. This could be an entire host, or a nodeboard or a blade. A single host can be made up of multiple vnodes. Each vnode can be managed and scheduled independently. Each vnode in a complex must have a unique name. Vnodes can share resources, such as node-locked licenses.

Host A machine with its own operating system, made up of one or more vnodes. Also, all vnodes with the same value for *resources_available.host*. A single host can be made up of multiple vnodes.

Chunk A set of resources allocated as a unit to a job. Specified inside a selection directive. All parts of a chunk come from the same host. In a typical MPI (Message-Passing Interface) job, there is one chunk per MPI process.

Cluster Generally, a very homogeneous set of systems that are viewed as one unit. Typically, the word "cluster" means "Linux cluster", although it is also being used to mean "Windows cluster".

Complex A PBS complex consists of the machines running one primary Server+Scheduler (plus, optionally, a secondary backup Server+Scheduler) and all the machines on which the MOMs (attached to this Server+Scheduler) are running. In general, it can be a very heterogeneous mix of system architectures, operating systems, and can include several clusters.

Load Balance A policy wherein jobs are distributed across multiple 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 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. 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 (characteristics) associated with them that provide control information. Such attributes include: `state`, the list of jobs to which the vnode is allocated, `boolean resources`, `max_running`, `max_user_run`, `max_group_run`, and both assigned and available resources ("`resources_assigned`" and "`resources_available`").

PBS Professional PBS consists of one server (`pbs_server`), one 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 system, multiple systems, a cluster of vnodes, or any combination of these.

Virtual Processor (VP) A vnode 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 vnode. The default number of virtual processors on a vnode is the number of currently functioning physical processors; the PBS Manager can change the number of VPs as required by local policy.

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

Account An *account* is an 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 Professional External Reference Specification**. A site may make use of the API to implement new commands if so desired.

Attribute An *attribute* is a data item whose value affects the behavior of or provides information about the object and can be set by its owner. For example, the user can supply values for attributes of a job, or

the administrator can set attributes of queues and vnodes.

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 vnodes that are allocated to only one job at a time or of vnodes that have many jobs executing at once on each vnode or a combination of these two scenarios.

Core — If multiple processors are implemented in a single package, connected to the computer by a single socket for all of them, then each individual processor is called a core. Note that this “socket” is a hardware socket, different from a network , or TCP, socket.

CPU — Has 2 meanings:
1) The package or silicon chip holding one or more processors ("single-core CPU", "dual-core CPU", or "quad-core CPU").
2) Resource provided by PBS execution hosts and requested by PBS jobs. Resource name is "ncpus". The amount offered by PBS execution hosts defaults to the number of installed cores, but may be changed by PBS managers (when they are called "virtual CPUs"). The amount requested by a job owner for a job denotes the number of processors intended to be used for this job.

Destination — This is the location within PBS where a job is sent. 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 — *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 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.

Group ID (GID) Numeric identifier uniquely assigned to each group (see Group).

Hold A restriction which prevents a job from being selected for processing. There are four types of holds. One is applied by the job owner ("user"), another is applied by a PBS *Operator*, a third is applied by the system itself or the PBS *Manager*; the fourth is set if the job fails due to an invalid password.

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.

Job Array A job array is a container for a collection of similar jobs. It can be submitted, queried, modified and displayed as a unit. For more on job arrays, see Job Arrays in the **PBS Professional User's Guide**.

Job State A job exists in one of the possible states throughout its existence within the PBS system. Possible states are: Queued, Running, Waiting, Transiting, Exiting, Suspended, Held, and Checkpointed. See also section 11.19.4 "Job States" on page 487.

Manager A *manager* is authorized to use all restricted capabilities of PBS. A 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 user who submitted a specific job to PBS.

PBS_HOME Refers to the path under which PBS was installed on the local system. Your local system administrator can provide the specific location.

Parameter A *parameter* provides control information for a component of PBS. Typically this is done by editing various configuration files.

Placement Set A set of vnodes. Placement sets are used to improve task placement (optimizing to provide a "good fit") by exposing

information on system configuration and topology. See “Placement Sets and Task Placement” on page 326.

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.

Processor Computer component which interprets instructions thereby processing data. Connected to the computer by a socket. Note that this “socket” is a hardware socket, different from a network , or TCP, socket.

Requeue The process of stopping a running (executing) job and putting it back into the queued (“Q”) state. This includes placing the job as close as possible to its former position in that queue.

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

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.

State See Job State.

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

Task Placement The process of choosing a set of vnodes to allocate to a job that will both satisfy the job's resource request (select and place specifications) and satisfy the configured Scheduling policy. See section 9.6 “Placement Sets and Task Placement” on page 326.

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

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).

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 Professional is provided. Next is the information necessary to make certain planning decisions.

3.1 New Features in PBS Professional 9.1

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

Administrator's Guide	Tunable formula for computing job priorites. See section 9.7.2 "Tunable Formula for Computing Job Priorities" on page 342.
Release Notes	Support for SLES 10 on x86, x86_64, and IA64

3.1.1 New Server Attribute for Tunable Formula

The new server attribute "job_sort_formula" is used for sorting jobs according to a site-defined formula. See section 9.7.2 "Tunable Formula for Computing Job Priorities" on page 342.

3.1.2 Change to sched_config

The default job_sort_key of cput is commented out in the default sched_config file. It is left in as a usage example.

3.1.3 Deprecations

The sort_priority option to job_sort_key is deprecated, and is replaced with job_priority.

3.2 Changes in Previous Release

3.2.1 Change to Licensing

PBS now depends on a FLEX/Altair license server that will hand out licenses to be assigned to PBS jobs. See section 5.1 “FLEX Licensing Feature” on page 81. A site can still use a trial license. See section 5.2 “Trial Licenses” on page 82. PBS Professional versions 8.0 and below will continue to be licensed using the proprietary licensing scheme.

3.2.2 Installing With FLEX Licensing

You must install and configure the FLEXlm license server before installing and configuring PBS. See section 4.1 “Overview of Installing PBS” on page 35.

3.2.3 Unset Resources Have Zero Value

An unset numerical resource at the host level behaves as if its value is zero, but at the server or queue level it behaves as if it were infinite. An unset string or string array resource cannot be matched by a job’s resource request. An unset boolean resource behaves as if it is set to “False”. See section 7.9.2 “Unset Resources” on page 219.

3.2.4 Better Management of Resources Allocated to Jobs

The resources allocated to a job from vnodes will not be released until certain allocated resources have been freed by all MOMs running the job. The end of job accounting record will not be written until all of the resources have been freed. The “end” entry in the job end (‘E’) record will include the time to stage out files, delete files, and free the resources. This will not change the recorded “walltime” for the job.

3.2.5 Support for Large Page Mode on AIX

PBS Professional supports Large Page Mode on AIX. No additional steps are required from the PBS administrator.

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 vnodes 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 Professional. Also review the Glossary of terms prior to installation and configuration of PBS Professional. (See also section "Concepts and Terms" on page 5.)

3.3.1 Resources Required by PBS

The amount of memory required by the PBS server and scheduler depends on the number of hosts and the number of jobs to be queued or running. Each host will need less than 512 bytes. The number of jobs is the important factor, since each job needs about 5 KB. The number of processors in the complex is not a factor.

3.3.2 Hostnames

The IP address of each machine in the complex should resolve to the fully qualified domain name for that machine. If you are using hostfiles, the FQDN should appear in `/etc/hosts`.

3.3.3 Network Configuration

Given that PBS is a distributed networked application, it is important that the network on which you will be deploying PBS is configured according to IETF standards. Specifically, forward and reverse name lookup should operate according to the standard.

3.3.4 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 Professional, as such decisions can change which parameters are selected for tuning PBS. For details, see the MOM configuration parameter "usecp" in section 8.2.2 "Syntax and Contents of Default Configuration File" on page 260 and section 12.3 "The pbs_rcp vs. scp Command" on page 494. The impact of file location and delivery are discussed further in the **PBS Professional User's Guide** in section 8.5 "Delivery of Output Files" on page 133.

3.3.5 SGI Altix cpuset Feature Requires ProPack Library

Customers who intend to run PBS Professional on SGI Altix systems using cpusets should note that there are strict requirements for the SGI ProPack (containing the cpuset API). ProPack 2.4 or greater is required. The library is required on MOM vnodes where cpuset functionality is desired. To test if the library is currently installed, execute the following command:

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

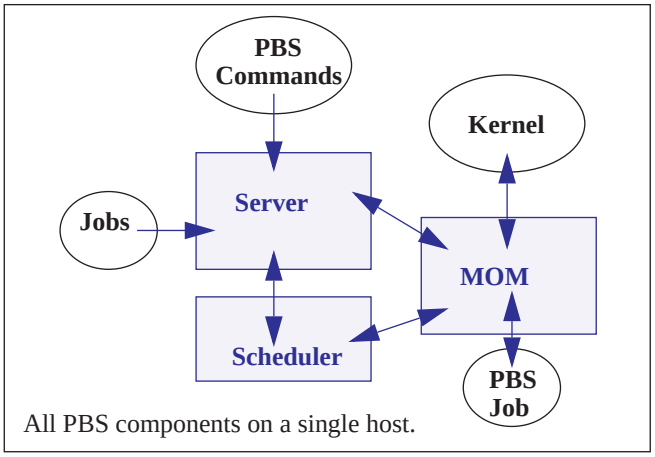
Important: The PBS Professional MOM binary for SGI's ProPack 2.4 and greater is pbs_mom.cpuset.

3.3.6 Using Comprehensive System Accounting on SGI Altix

Comprehensive System Accounting (CSA) on SGI Altix requires that both the Linux job container facility and CSA support be either built into the kernel or available as a loadable module. It also requires SGI ProPack 2.4 or greater. See section 8.10.4 "Configuring MOM for Comprehensive System Accounting" on page 300.

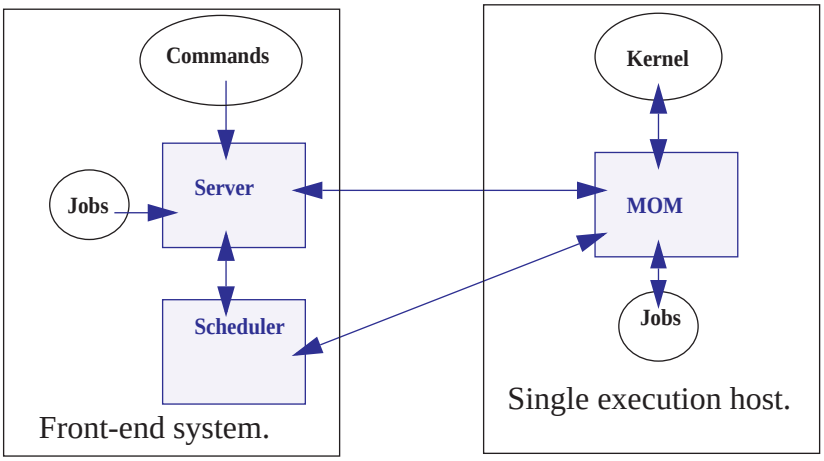
3.4 Single Execution System

If PBS is to be installed on a single system, all three components 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.



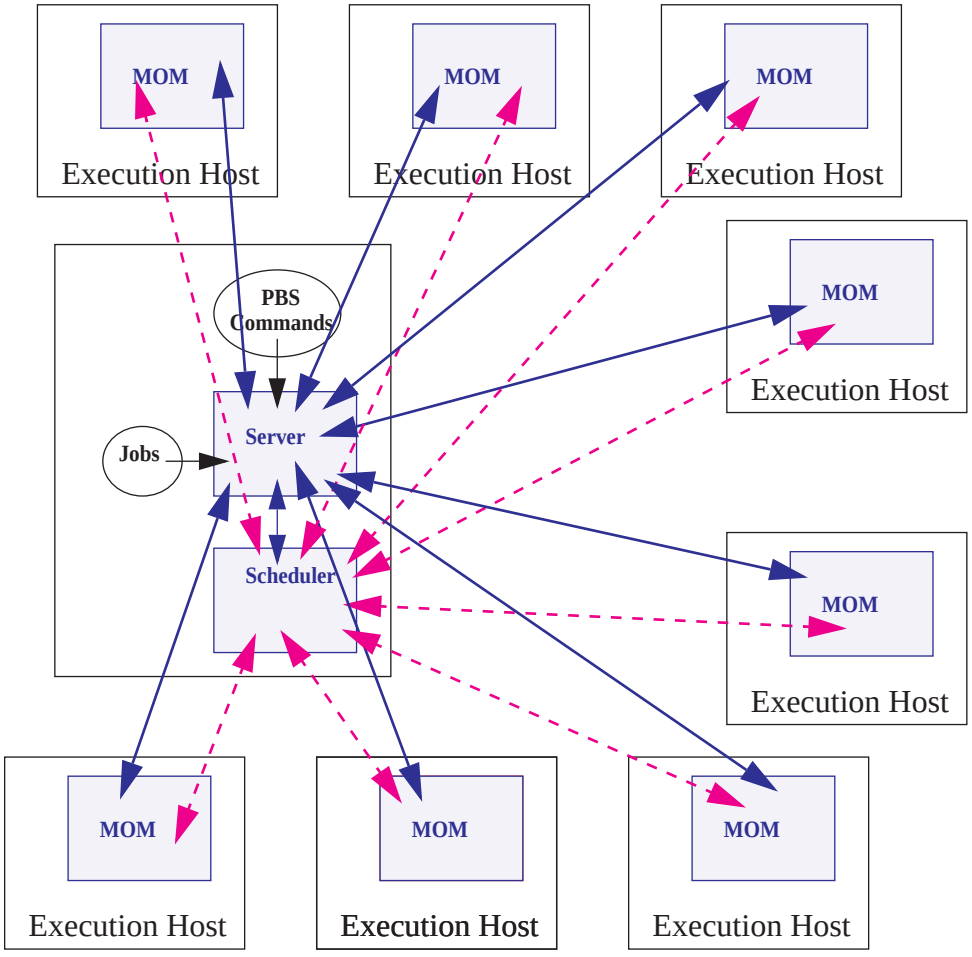
3.4.1 Single Execution System with Front-end

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 complex) of systems, normally the Server (`pbs_server`) and the Scheduler (`pbs_sched`) are 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 host (i.e. each system where jobs are to be executed). The following diagram illustrates this for an eight host complex.



3.6 UNIX User Authorization

When the user submits a job from a system other than the one on which the PBS Server is running, system-level user authorization is required. This authorization is needed for submitting the job and for PBS to return output files (see also section 8.5 "Delivery of Output Files" on page 133 and section 8.6 "Input/Output File Staging" on page 134 in the **PBS Professional User's Guide**).

Important: The username under which the job is to be executed is selected according to the rules listed under the "`-u`" option to `qsub` (as discussed in the **PBS Professional User's Guide**). The user submitting the job must be authorized to run the job under the execution user name (whether explicitly specified or not).

Such authorization is provided by any of the following methods:

1. The host on which `qsub` is run (i.e. the submission host) is trusted by the server. This permission may be granted at the system level by having the submission host as one of the entries in the server's `host.equiv` file naming the submission host. For file delivery and file staging, the host representing the source of the file must be in the receiving host's `host.equiv` file. Such entries require system administrator access.

2. The host on which `qsub` is run (i.e. the submission host) is explicitly trusted by the server via the user's `.rhosts` file in his/her home directory. The `.rhosts` must contain an entry for the system from which the job is submitted, with the user name portion set to the name under which the job will run. For file delivery and file staging, the host representing the source of the file must be in the user's `.rhosts` file on the receiving host. It is recommended to have two lines per host, one with just the "base" host name and one with the full hostname, e.g.: `host.domain.name`.

3. PBS may be configured to use the Secure Copy (scp) for file transfers. The administrator sets up SSH keys as described in "Enabling Hostbased Authentication on Linux" on page 426. See also section 8.5 "Delivery of Output Files" on page 133 of

the **PBS Professional User's Guide**.

4. User authentication may also be enabled by setting the server's `flatuid` attribute to "True". See"flatuid" on page 186 and "User Authorization" on page 426. Note that flatuid may open a security hole in the case where a vnode has been logged into by someone impersonating a genuine user.

3.7 Recommended PBS Configurations for Windows

This section presents the recommended configuration for running PBS Professional under Windows.

3.7.1 Definitions

Active Directory — Active Directory is an implementation of LDAP directory services by Microsoft to use in Windows environments. It is a directory service used to store information about the network resources (e.g. user accounts and groups) across a domain. It was released first with Windows 2000 Server, and extended/improved under Windows Server 2003. Active Directory is fully integrated with DNS and TCP/IP; DNS is required. To be fully functional, the DNS server must support SRV resource records or service records.

Install Account — The account used to install PBS.

pbsadmin — The account that is used to execute the PBS daemons pbs_server, pbs_mom, pbs_sched, and pbs_rshd via the Service Control Manager on Windows. It is currently set to "pbsadmin".

Domain User Account — It is a domain account on Windows that is a member of the "Domain Users" group.

Domain Admin Account — It is a domain account on Windows that is a member of the "Domain Admins" group.

Admin — As referred to in various parts of this document, it is a user logged in from an account who is a member of any group that has full control over the local computer, domain controller, or is allowed to make domain and schema changes to the Active directory.

Domain Admins — A global group whose members are authorized to administer the domain. By default, the Domain Admins group is a member of the Administrators group on all computers that have joined a

domain, including the domain controllers.

Domain Users A global group that, by default, includes all user accounts in a domain. When you create a user account in a domain, it is added to this group automatically.

Administrators A group that has built-in capabilities that give its members full control over the local system, or the domain controller host itself.

Enterprise Admins A group that exists only in the root domain of an Active Directory forest of domains. The group is authorized to make forest-wide changes in Active Directory, such as adding child domains.

Schema Admins A group that exists only in the root domain of an Active Directory forest of domains. The group is authorized to make schema changes in Active Directory.

Delegation A capability provided by Active Directory that allows granular assignment of privileges to a domain account or group. So for instance, instead of adding an account to the “Account Operators” group which might give too much access, then delegation allows giving the account read access only to all domain users and groups information. This is done via the Delegation wizard.

3.7.2 Windows Configuration in a Domained Environment

3.7.2.1 Machines

- The PBS clients, server, MOMs, scheduler, and rshds must run on a set of Windows machines networked in a single domain.
- The machines must be members of this one domain, and they must be dependent on a centralized database located on the primary/secondary domain controllers.
- The domain controllers must be running on a Server type of Windows host, using Active Directory configured in "native" mode.
- The choice of DNS must be compatible with Active Directory.

- The PBS server and scheduler must be run on a Server type of Windows machine that is not the domain controller and is running Active Directory.

- PBS must not be installed or run on a Windows machine that is serving as the domain controller (running Active Directory) to the PBS hosts.

3.7.2.2 Installation Account

- The installation account is the account from which PBS is installed. The installation account must be the only account that will be used for all steps of PBS installation including modifying configuration files, setting up failover, and so on. If any of the PBS configuration files are modified by an account that is not the installation account, permissions/ownerships of the files could be reset, rendering them inaccessible to PBS. For domained environments, the installation account must be a local account that is a member of the local Administrators group on the local computer.

3.7.2.3 The PBS Service Account

- The service account is the account under which the PBS services (pbs_server, pbs_mom, pbs_sched, pbs_rshd) will run. This account is called "pbsadmin".

- This account must exist while any PBS services are running.

- The password for this account should not be changed while PBS is running.

- Create the service account before installing PBS.

- For domained environments, the service account must:
  1. be a domain account
  2. be a member of the "Domain Users" group, and **only** this group

3. have "domain read" privilege to all users and groups.

- For a domained environment, delegate “read access to all users and groups information” to the pbsadmin account. See section 3.7.2.4 “Delegating Read Access” on page 24.

- If the pbsadmin account is set up with no explicit domain read privilege, MOM may hang. This happens when XP users submit jobs from a network mapped drive without the -o/-e option for redirecting files. When this happens, bring up Task manager, look for a "cmd" process by the user who owned the job, and kill it. After the first cmd process is killed, you may have to look for a second one (the first one copies the output file, the second one does the error file). This should un-hang the MOM.

3.7.2.4 Delegating Read Access

- To delegate “read access to users and groups information” to the pbsadmin account:

  1. On the domain controller host, bring up Active Directory Users and Computers.

  2. Select <domain name>, right mouse click, and choose "Delegate Control". This will bring up the "Delegation of Control Wizard".

  3. When it asks for a user or group to which to delegate control, select “pbsadmin”.

  4. When it asks for a task to delegate, specify "Create a custom task to delegate".

  5. For active directory object type, select the "this folder, existing objects in this folder, and creation of objects in this folder" button.

  6. For permissions, select “Read” and “Read All Properties”.

  7. Exit out of Active Directory.

3.7.2.5 User Accounts

- Each user must explicitly be assigned a HomeDirectory sitting on some network path. PBS does not support a HomeDirectory that is not network-mounted. PBS currently supports network-mounted directories that are using the Windows network share facility.

- If a user was not assigned a HomeDirectory, then PBS uses `PROFILE_PATH\My Documents\PBS Pro`, where `PROFILE_PATH` could be, for example, "`\Documents and Settings\`*`username`*".

3.7.2.6 User Jobs

- All users must submit and run PBS jobs using only their domain accounts (no local accounts), and domain groups. If a user has both a domain account and local account, then PBS will ensure that the job runs under the domain account.

- Each user must always supply an initial password in order to submit jobs. This is done by running the pbs_password command at least once to supply the password that PBS will use to run the user's jobs.

- Access by jobs to network resources, such as a network drive, requires a password.

- All job scripts, as well as input, output, error, and intermediate files of a PBS job must reside in an NTFS directory.

3.7.2.7 Adding to Local Administrators Group

- Add the PBS service account "pbsadmin" to the local Administrators group:

  ```
  net localgroup Administrators <domain name>\pbsadmin /add
  ```

3.7.2.8 Installation Path

- The destination/installation path of PBS must be NTFS. All PBS configuration files must reside on an NTFS filesystem.

3.7.2.9 Installation

- The “pbsadmin” account must be used as the service account in future invocations of the install program when setting up a complex of PBS hosts.

- The install program requires the installer to supply the password for “pbsadmin”. This same password must be supplied to future invocations of the install program on other servers/hosts.

- The install program will enable the following rights to pbsadmin: "Create Token Object", "Replace Process Level Token", "Log On As a Service", and "Act As Part of the Operating System".

- The install program will enable Full Control permission to local "Administrators" group on the install host for all PBS-related files.

- The install program will give you a specific error if it fails to make the service account be a member of the local Administrators group on the local computer. It will quit at this point, and you must go back and make the service account be a member of the local Administrators group on the local computer. See section 3.7.2.7 “Adding to Local Administrators Group” on page 25. Then re-run the install program.

3.7.3 Windows Configuration in a Standalone Environment

3.7.3.1 Machines

- PBS must be run on a set of machines that are not members of any domain (workgroup setup); no domain controllers/ Active Directory are involved.

3.7.3.2 Installation Account

- The installation account is the account from which PBS is installed. The installation account must be the only account that will be used in all aspects of PBS installation including modifying configuration files, setting up failover, and so on. If any of the PBS configuration files are modified by an account that is not the installation account, permissions/ownerships of the files could be reset, rendering them inaccessible to PBS.

- For standalone environments, the installation account must be a local account that is a member of the local Administrators group on the local computer.

3.7.3.3 The pbsadmin Service Account for Standalone Environments

- The service account is the account under which PBS services (pbs_server, pbs_mom, pbs_sched, pbs_rshd) will run. This account is called "pbsadmin".

- This account must exist while any PBS services are running.

- The password for this account should not be changed while PBS is running.

- Create the pbsadmin service account before installing PBS.

- For standalone environments, the pbsadmin account must be a local account that is a member of the local Administrators group on the local computer. See section 3.7.3.6 "Adding to Local Administrators Group" on page 28.

3.7.3.4 User Accounts

- Each user should be assigned a local NTFS HomeDirectory.
- If a user was not assigned a HomeDirectory, then PBS uses `PROFILE_PATH\My Documents\PBS Pro`, where `PROFILE_PATH` could be, for example, "`\Documents and Settings\`*`username`*".
- A local account/group having the same name must exist for the user on all the execution hosts.

3.7.3.5 User Jobs

- Users must submit and run PBS jobs using their local accounts and local groups.
- Users should supply a password when submitting jobs. That password must be the same on all the execution hosts. (See also section 7.15 "Password Management for Windows" on page 240).
- Job intermediate, input, output, and error files must reside on an NTFS filesystem.
- For a password-ed job, users can access (within the job script) folders in a network share. (See also the discussion of the single-signon feature in section 7.15 "Password Management for Windows" on page 240).
- Each user must always supply an initial password to submit jobs. This is done by running the pbs_password command at least once to supply the password that PBS will use to run the user's jobs.
- Access by jobs to network resources (such as a network drive), requires a password.

3.7.3.6 Adding to Local Administrators Group

- Add the PBS service account "pbsadmin" to the local Administrators group:

```
net localgroup Administrators pbsadmin /
add
```

3.7.3.7 Installation Path

- The destination/installation path of PBS must be NTFS. All PBS configuration files must reside on an NTFS filesystem.

3.7.3.8 Installation

- The "pbsadmin" account must be used as the service account in future invocations of the install program when setting up a complex of PBS hosts.

- The install program will require the installer to supply the password for the "pbsadmin" account. The same password must be supplied to future invocations of the install program on other Servers/hosts.

- The install program will enable the following rights to the "pbsadmin" account: "Create Token Object", "Replace Process Level Token", "Log On As a Service", and "Act As Part of the Operating System".

- The install program will enable Full Control permission to local "Administrators" group on the install host for all PBS-related files.

3.7.4 Unsupported Windows Configurations

The following Windows configurations are currently unsupported:

- PBS running on a set of Windows 2000, Windows XP, Windows 2000 server hosts that are involved in several "domains" via any trust mechanism.

- Using NIS/NIS+ for authentication on non-domain accounts.

- Using RSA SecurID module with Windows logons as a means

of authenticating non-domain accounts.

3.7.5 Sample Windows Deployment Scenario

For planning and illustrative purposes, this section describes deploying PBS Professional on a complex of 20 machines networked in a single domain, with host 1 as the server host, and hosts 2 through 20 as the execution hosts.

For this configuration, the installation program is run 20 times, invoked once per host. "All" mode (Server, Scheduler, MOM, and `rshd`) installation is selected only on host 1, and "Execution" mode (MOM and `rshd`) installs are selected on the other 19 hosts. The pbsadmin account must exist in advance. This account is used for running the PBS services.

The user who runs the `install` program will supply the password for the pbsadmin account. The installation program will then propagate the password to the local Services Control Manager database.

A reboot of each machine is necessary at the end of each install.

3.8 Windows User Authorization

When the user submits a job from a system other than the one on which the PBS Server is running, system-level user authorization is required. This authorization is needed for submitting the job and for PBS to return output files (see also section 8.5 "Delivery of Output Files" on page 133 and section 8.6 "Input/Output File Staging" on page 134 in the **PBS Professional User's Guide**).

If running in a domained environment, then a password is also required for user authorization. See the discussion of single-signon in section section 7.15 "Password Management for Windows" on page 240.

Important: The username under which the job is to be executed is selected according to the rules listed under the "`-u`" option to `qsub` (as discussed in the **PBS Professional User's Guide**). The user submitting the job must be authorized to run the job under the execution user name (whether explicitly specified or not).

Such authorization is provided by any of the following methods:

1. The host on which `qsub` is run (i.e. the submission host) is trusted by the execution host. This permission may be granted at the system level by having the submission host as one of the entries in the execution host's `host.equiv` file naming the submission host. For file delivery and file staging, the host representing the source of the file must be in the receiving host's `host.equiv` file. Such entries require system administrator access.

2. The host on which `qsub` is run (i.e. the submission host) is explicitly trusted by each execution host via the user's `.rhosts` file in his/her home directory. The `.rhosts` must contain an entry for the system on which the job will execute, with the user name portion set to the name under which the job will run. For file delivery and file staging, the host representing the source of the file must be in the user's `.rhosts` file on the receiving host. It is recommended to have two lines per host, one with just the "base" host name and one with the full hostname, e.g.: `host.domain.name`.

3.8.1 Windows hosts.equiv File

The Windows `hosts.equiv` file determines the list of non-Administrator accounts that are allowed access to the local host, that is, the host containing this file. This file also determines whether a remote user is allowed to submit jobs to the local PBS Server, with the user on the local host being a non-Administrator account.

This file is usually: `%WINDIR%\system32\drivers\etc\hosts.equiv`.

The format of the `hosts.equiv` file is as follows:

```
[+|-] hostname username
```

'+' means enable access, whereas '-' means to disable access. If '+' or '-' is not specified, then this implies enabling of access. If only *`hostname`* is given, then users logged into that host are allowed access to like-named accounts on the local host. If only *`username`* is given, then that user has access to all accounts (except Administrator-type users) on the local host. Finally, if both *`hostname`* and *`username`* are given, then user at that host has access to like-named account on local host.

Important: The `hosts.equiv` file must be owned by an admin-type user or group, with write access granted to an admin-type user or group.

3.8.2 Windows User's HOMEDIR

Each Windows user is assumed to have a home directory (`HOMEDIR`) where his/her PBS job will initially be started. (The home directory is also the starting location of file transfers when users specify relative path arguments to `qsub/qalter -W stagein/stageout` options.)

The home directory can be configured by an Administrator via setting the user's HomeDirectory field in the user database, via the User Management Tool. It is important to include the drive letter when specifying the home directory path. The directory specified for the home folder must be accessible to the user. If the directory has incorrect permissions, PBS will be unable to run jobs for the user.

Important: You must specify an already existing directory for home folder. If you don't, the system will create it for you, but set the permissions to that which will make it inaccessible to the user.

If a user has not been explicitly assigned a home directory, then PBS will use this Windows-assigned default, local home directory as base location for its default home directory. More specifically, the actual home path will be:

```
[PROFILE_PATH]\My Documents\PBS Pro
```

For instance, if a *userA* has not been assigned a home directory, it will default to a local home directory of:

```
\Documents and Settings\userA\My Documents\PBS Pro
```

UserA's job will use the above path as working directory, and any relative pathnames in stagein, stageout, output, error file delivery will resolve to the above path.

Note that Windows can return as `PROFILE_PATH` one of the following forms:

```
\Documents and Settings\username
\Documents and Settings\username.local-hostname
\Documents and Settings\username.local-hostname.00N where N is a number
\Documents and Settings\username.domain-name
```

A user can be assigned a HomeDirectory that is network mounted. For instance, a user's directory can be: "`\\fileserver_host\sharename`". This would cause PBS to map this network path to a local drive, say `G:`, and allow the working directory of user's job to be on this drive. It is important that the network location (file server) for the home directory be on a Server-type of Windows machine like Windows 2000 Server or Windows 2000 Advanced Server. Workstation-type machines like Windows 2000 Professional or Windows XP Professional have an inherent limit on the maximum number of outgoing network connections (10) which can cause PBS to fail to map or even access the user's network HomeDirectory. The net effect is the job's working directory ends up in the user's default directory: `PROFILE_PATH\My Documents\PBS Pro`.

If a user has been set up with a home directory network mounted, such as referencing a mapped network drive in a HOMEDIR path, then the user must submit jobs with a password either via `qsub -Wpwd=""` (see the discussion of `qsub` in the **PBS Professional User's Guide**) or via the single-signon feature (see section section 7.15 "Password Management for Windows" on page 240). When PBS runs the job, it will change directory to the user's home directory using his/her credential which must be unique (and passworded) when network resources are involved.

To avoid having to require passworded jobs, users must be set up to have a local home directory. Do this by accessing `Start Menu->Settings->Control Panel->Administrative Tools->Computer Management` (Win2000) or `Start Menu->Control Panel->Performance and Maintenance->Administrative Tools->Computer Management` (Windows XP), and selecting `System Tools->Local Users and Groups`, double clicking `Users` on the right pane, double clicking on the *username* which brings up the user properties dialog from which you can select `Profile`, and specify an input for Local path (Home path). Be sure to include the drive information.

Chapter 3
Pre-Installation Planning

Chapter 4
Installation

The license server must be installed before installing PBS. This chapter shows how to install PBS Professional. You should read the Release Notes and Chapter 3: Planning before installing the software.

4.1 Overview of Installing PBS

There are two steps to installing PBS. The first step is to install the license server package, and the second step is to install PBS itself on a selected platform. Note that if the license server is not started first, jobs may be delayed before being run.

If you are going to use a trial license, you do not need to download or install the FLEXlm license server. See section 4.12.1 "Trial Licenses" on page 80.

Download both the PBS Professional package and the License Server package. The license server package contains a license server daemon and an Altair vendor daemon. The PBS Professional and Altair License Server packages are available on the PBS/Altair download page at http://www.pbspro.com/UserArea/Software/.

The License Server package must be installed on the license server host(s) and configured before installing the PBS Professional package. PBS will not run jobs without an installed and configured license server.

When you run the PBS install program on platforms that don't support FLEX-licensed applications, you will not be prompted for a "server type" installation option; only for "commands-only" or "execution type". The server/scheduler component in the PBS Professional package will not be included in platforms that don't support FLEXlm-licensed applications. A list of these platforms is provided in the Release Notes.

4.2 FLEX Licensing

4.2.1 Steps in Downloading and Licensing PBS Professional

The administrator of the site downloads the PBS and License Server packages.

The administrator of the site installs the License Server package on the selected host, HostB.

The site administrator gets the **lmhostid** (not hostid) of **HostB** (the selected server host), and provides this to Altair with the request for X number of CPU licenses.

Altair gives the requesting site a license file valid only on HostB, and this file contains the license key/signature for PBS Professional authorizing use of X licenses, which will license X CPUs.

The administrator of the site puts the license file on HostB and configures the License Server.

Finally, the administrator installs the PBS package on the selected host, HostA, configuring it to point to a license server on HostB. HostA may be the same as HostB.

4.2.2 Redundant License Servers

You may wish to set up redundant license servers. These must be set up before starting PBS. See section 5.5 “Redundant License Servers” on page 93.

4.3 License Server Installation

4.3.1 Overview

There are four main components of FLEXnet.

License manager daemon (lmgrd)
Vendor daemon (altair_lm)
License file (altair_lic.dat)
Licensed software (PBS Professional)

The license file is a text file containing the license data, such as information about the license server's host and the vendor daemon. This file is called altair_lic.dat. The default location for the license file is:

UNIX
[User-defined licensing directory]/altair/security/altair_lic.dat
Windows
[User-defined drive]:[User-defined licensing directory]\security\altair_lic.dat

It is recommended that you run the startup script, which will then start the licensing daemons as a non-root user. See section 4.3.2.7 "Modifying the Startup Script" on page 42.

The administrator starts lmgrd using the altairlmgrd startup script. The lmgrd daemon in turn starts the altair_lm daemon.

Steps in installing the FLEXlm license server:
Select a host for the license server(s)
Download the license server package.
Untar the license server
Obtain your lmhostid.
Get and install your license file.
Install the startup script.
Modify the startup script.
Start the license daemon.
Follow the above for each license server.

4.3.2 Installing the FLEX License Server on UNIX/Linux

In the following instructions, substitute the architecture/OS you'll be using for "$(ARCH)".

4.3.2.1 Selecting a Host for the License Server

Use the following requirements for license server host(s) for more than about 100 licenses.

Sockets

Each licensed application connected to a license server uses one or more sockets. The number of sockets available to the license server is defined by the per-process system limit for file descriptors. The total number of sockets used by the license server is slightly larger than the total number needed by the licensed applications. If the number of sockets required by the license server on a single machine becomes excessive, you can split the license file into more than one file on different servers, to lighten networking traffic. PBS Professional can then check out licenses from multiple servers using a license-file list via the pbs_license_file_location attribute. See section 5.4.3.1 “Setting the License File Location in pbs_license_file_location” on page 88.

CPU Time

For small numbers of clients, the license servers use very little CPU time. The servers might have only a few seconds of CPU time after many days. For a large number of clients, or for high checkout/checkin activity levels (hundreds per second), the amount of CPU time consumed by the server may start to become significant, although, even here, CP U usage is normally not high. In this case, you may need to ensure that the server machine you select has enough CPU cycles to spare.

Disk Space

The only output files created by the license servers are the debug and report log files. If you have a lot of license activity, these log files grow very large. You need to consider where to put these files, how often to rotate and archive them. The license administrator has the option to suppress log file output if disk space is at a premium. It is recommended that the log files are local files on the server machine(s) to avoid network dependencies.

Memory

The license manager daemons use little memory. On SunOS, lmgrd uses approximately 2MB and the vendor daemons use approximately 2MB each, although memory usage increases in the vendor daemon with the size of the license file, size of the options file and the number of concurrent users.

Network Bandwidth

FLEXlm sends relatively small amounts of data across the network. Each transaction, such as a checkout or checkin, is typically satisfied with less than 1KB of data transferred. This means that FLEXlm licensing can be effectively run over slow networks (such as dial-up SLIP lines) for a small number of clients. For a large number of FLEXlm applications (hundreds), the network bandwidth may start to become significant. In this case, run the client application and license server on the same local area network, which may require splitting licenses between two files for two servers. PBS Professional can use the pbs_license_file_location attribute to have effective access to both servers.

Remote Mounted Disks
Macrovision recommends that you do not use remote mounted disks when you run the license server. In other words, lmgrd, altair_lm, the license file, the debug log file and the report log files should be on local mounted disks. If any of these files are on a remote mounted disk, you double the number of points of failure during which you could temporarily lose all your licenses. When all files are mounted locally, the licenses are available as long as the server host is up, but when the files are on a different machine, then the loss of either the license server machine, or the file server machine causes licenses to be unavailable.

Redundant License Servers
If you want to use redundant servers, select stable systems as server machines. In other words, do not pick systems that are frequently rebooted or shut down. FLEXlm supports redundancy via a license-file list in the pbs_license_file_location server attribute, or via a set of three redundant license servers. Redundant license servers are covered in section 5.5 "Redundant License Servers" on page 93.

4.3.2.2 Downloading the License Server Package

Download the package that is appropriate for the host that you have chosen to be your FlexLM license server. Go to:

http://www.pbspro.com/UserArea/Software/

4.3.2.3 Untarring the License Server

1. Create the directory where the license manager files will reside. If you are installing or running daemons via a non-root user, choose a directory that is writeable by that non-root user. If this directory is writeable only by root, then you must be root to do the installation.

The license daemon is called "lmgrd" and is located at <installation location>/altair/scripts/lmgrd. It is recommended although not required that you run lmgrd as a non-root user. If you choose to run it as a non-root user, please make sure that the non-root user has write access to the current working directory. The lmgrd daemon will write log files to the directory where it runs. This directory must be <installation location>/altair/security.

Make the installation location directory:

```
mkdir <installation location>
```

2. Copy the tarball to this directory:

```
cp <path-to-tarball>/altair_flexlm.$(ARCH).tar.gz \
   <installation location>
```

3. Change directory:

```
cd <installation location>
```

4 Use `gunzip` to uncompress the archive:

```
gunzip altair_flexlm.$(ARCH).tar.gz
```

5. Use `tar` to extract the files:

```
tar -xf altair_flexlm.$(ARCH).tar
```

4.3.2.4 Obtaining Your lmhostid

You need the lmhostid of the machine on which the FLEXlm server will run. Run the **lmhostid** script on that machine to obtain your lmhostid:

```
<installation location>/altair/scripts/lmhostid
```

4.3.2.5 Getting and Installing Your License File

Step 1 Go to the PBS Professional User Area download page:
http://www.pbspro.com/UserArea/

Step 2 Click on the License Manager link.

Step 3 Click “Create New License”.

Step 4 Choose 1 server or 3 servers.

Step 5 Enter your host type(s), lmhostid(s), and host name(s). If you are using the three-server redundant setup, enter this information for all three servers.

Step 6 Enter the number of CPUs.

Step 7 Click "Generate License" button.

Step 8 Download the license file.

Step 9 If you are not using a trial license, copy the license file you saved from the PBS Professional User Area to <installation location>/altair/security/altair_lic.dat:

```
cp /tmp/altair_lic.dat \
    <installation location>/altair/ \
    security/altair_lic.dat
```

Step 10 If you will use the three-server redundant license server setup, make sure that each server is listed in the license file. To set up three-server redundancy, you must provide the lmhostid for each of the three servers to Altair Engineering to obtain a license file with three SERVER lines. Each license server requires lmgrd, the vendor daemon altair_lm, and the license file to be on the local file system.

Step 11 If you are using a trial license, see section 4.12.1 "Trial Licenses" on page 80.

4.3.2.6 Installing the Startup Script

1. Become root:

```
su
```

2. Run the install_altairlm.sh script to install the startup script:

```
<installation location>/altair/scripts/ \
   install_altairlm.sh
```

You should see a message like the following (installation location may vary by system):

```
<installation location>/altair/security/altairlm.init
has been installed into
   <path to startup script>/altairlmgrd
The Altair License Manager Daemon has been installed
```

4.3.2.7 Modifying the Startup Script

Follow these steps to start the license server as a non-root user (**recommended**, but not required):

1. Make sure that `<installation location>/altair/security` is writeable by the selected non-root user.

2. Become root if you are not already:

```
su
```

3. Go to the directory where the script is installed:

```
cd <path to startup script>
```

4. Edit the StartLm() function in `altairlm.init`:

Open the script in a text editor and search for StartLm. You will see comments that look like this:

--- begin comments ---

```
# If you would like to run as a custom user please set the following
# environment variable to the desired user, MAKE SURE that
# $ALTAIR_HOME/security is writeable by that user. Then uncomment the
# following two lines, and comment out the third.
#  DAEMON_USER="daemon"
#  su $DAEMON_USER -c "/bin/sh -c \" cd $ALTAIR_HOME/security &&
$ALTAIR_HOME/scripts/lmgrd -l  $ALTAIR_LOGFILE\""
  /bin/sh -c "cd $ALTAIR_HOME/security && $ALTAIR_HOME/scripts/lmgrd -l
$ALTAIR_LOGFILE"
```

--- end comments ---

5. Follow the instructions contained in the comments: When you are done the section should look like this:

--- begin comments ---

```
# If you would like to run as a custom user please set the following
# environment variable to the desired user, MAKE SURE that
# $ALTAIR_HOME/security is writeable by that user. Then uncomment the
# following two lines, and comment out the third.
  DAEMON_USER="daemon"
  su $DAEMON_USER -c "/bin/sh -c \" cd $ALTAIR_HOME/security &&
$ALTAIR_HOME/scripts/lmgrd -l  $ALTAIR_LOGFILE\""
#  /bin/sh -c "cd $ALTAIR_HOME/security && $ALTAIR_HOME/scripts/lmgrd -l
$ALTAIR_LOGFILE"
```

--- end comments ---

4.3.2.8 Starting and Stopping the Altair License Daemon

Once the startup script is installed and configured, you may start the Altair License Server.

Execute the following command as root:

```
<path to startup script>/altairlmgrd start
```

You may stop the Altair License Server by executing the following command as root.

```
<path to startup script>/altairlmgrd stop
```

4.3.3 Installing the FLEX License Server on Windows

4.3.3.1 Select a Host for the License Server

Select a host for the license server. See the considerations listed in section 4.3.2.1 “Selecting a Host for the License Server” on page 37.

4.3.3.2 Download the License Server Package

Download the package that is appropriate for the host that you have chosen to be your FlexLM license server. Go to:

http://www.pbspro.com/UserArea/Software/

You will get an executable named `fs8.0sr1_WIN32.exe.`

4.3.3.3 Install as Administrator

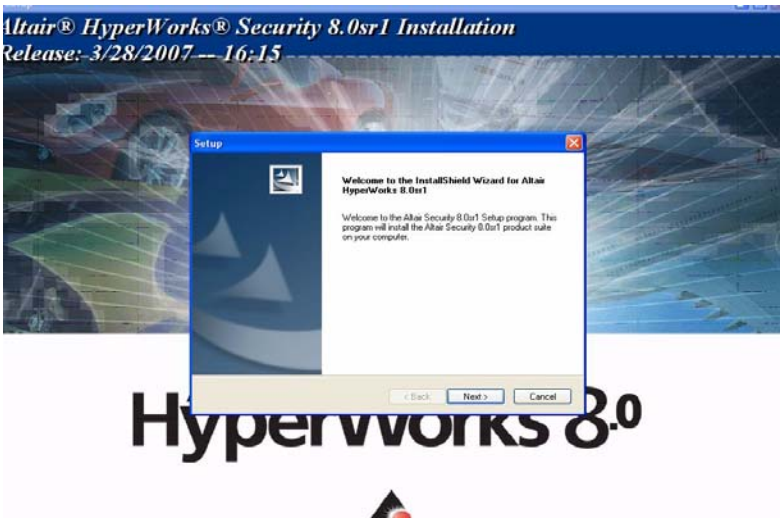
Log in as Administrator.

4.3.3.4 Start Installation Wizard

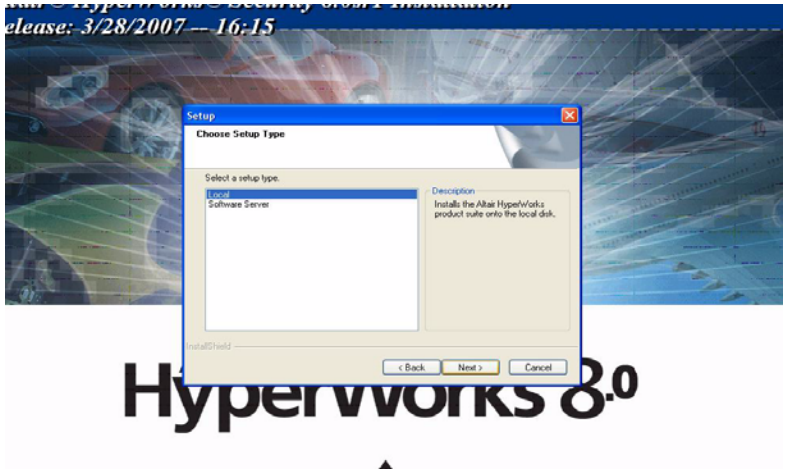
Double-click on the executable `fs8.0sr1_WIN32.exe.` The installation wizard should launch automatically. If the installation wizard does not start up, go to My Computer or Windows Explorer and run:

```
<drive letter>\<path to executable>\fs8.0sr1_WIN32.exe
```

4.3.3.5 Follow Installation Wizard

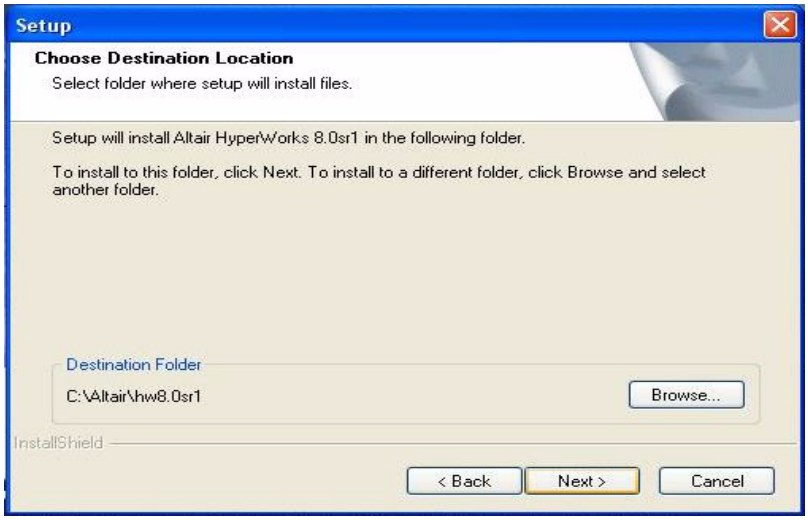


On the Welcome window, click Next.



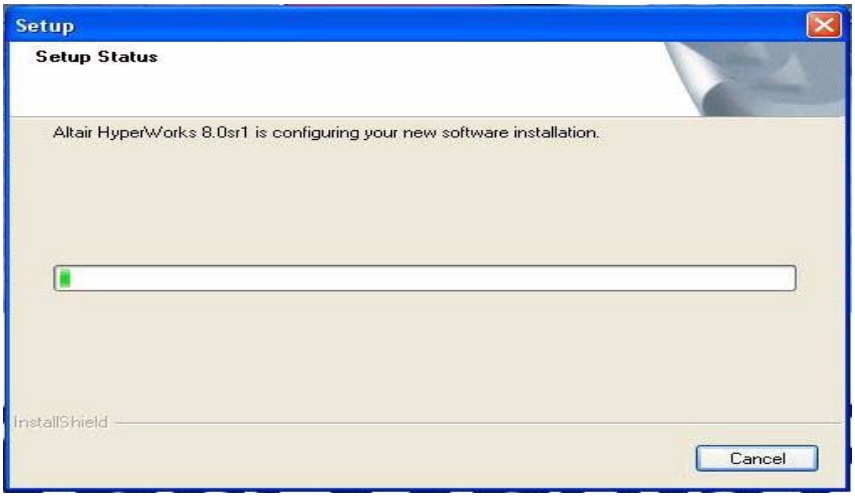
On the Choose Setup Type panel, select Local to install the license server.

Click Next.



On the Choose Destination Location dialog, select where the license server will be installed.

Click Next.



Wait for the Setup Status panel to indicate that the installation is complete.

In the next two panels, click OK. The first mentions the FLEXid driver, which is not supported by Altair and is not used for PBS Professional. The second panel is a short README.

On the InstallShield Wizard Complete dialog, click Finish to close the installation wizard.

4.3.3.6 Get lmhostid

Run the lmhostid script on the license server host:

```
<install_location>\security\WIN32\lmutil lmhostid
```

Use the first entry in the resulting list.
The first hostid must be for a hardware entity that is enabled, otherwise the license server won't start.

4.3.3.7 Get and Install License File

Step 1 Go to http://www.pbspro.com/UserArea/

Step 2 Click on the License Manager link.

Step 3 Click "Create New License".

Step 4 Choose 1 server or 3 servers.

Step 5 Enter your host type(s), lmhostid(s), and host name(s). If you are using the three-server redundant setup, enter this information for all three servers.

Step 6 Enter the number of CPUs.

Step 7 Click "Generate License" button.

Step 8 Download the license file.

Step 9 Copy the license file to <install location>\security\altair_lic.dat.

4.3.3.8 Copy License to Destination Directory

If you copied your license to a temporary directory, copy it from the temporary directory to the `<install_location>\security\altair_lic.dat` file.

4.3.3.9 Set Up Redundant License Servers

If you have redundant license servers, install the license server on each server using the above instructions.

4.3.3.10 Activate License

For server or network licenses, the first word in the license file is SERVER.

On the license server machine(s):

Step 1 Save the license file you received via e-mail as a text file. Make sure that the extension remains ".dat". It should be called
`<install_location>\security\altair_lic.dat.`

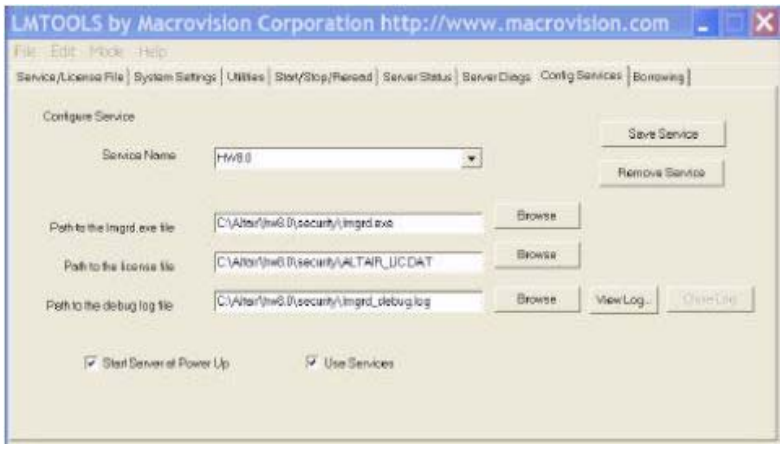
Step 2 If this is a three-server license setup, copy this license file to all the servers listed in the license file.

Step 3 Double click
`<install_location>\security\win32\lmtools.exe.`

Step 4 Click the Service/License File tab.

Step 5 Click the Configuration using Services option.

Step 6 Click the Config Services tab.



Step 7 Verify that all path names specified are correct. If necessary, use the respective Browse buttons to specify the path names. The path names should be similar to the following:

Path to the lmgrd.exe file:

```
<install_location>\
    security\WIN32\lmgrd.exe
```

Path to the license file:

```
<install_location>\
    security\altair_lic.dat
```

Path to the debug log file:

```
<install_location>\
    security\WIN32\lmgrd_debug.log
```

Step 8 Check the Use Services option.

Step 9 Click the options Use Server and Start Server at Power Up.

Step 10 Click Save Service.

Step 11 Go to the Start/Stop/Reread tab.

Step 12 Click Start Server.

4.3.3.11 Activate License on Redundant Servers

If this installation is part of a redundant license server setup, repeat the steps to activate the license on all of the license server machines.

4.4 Installing PBS

4.4.1 Steps in Installing PBS

The PBS software can be installed from the PBS CD-ROM or downloaded from the User 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	Set the FLEXlm license file location on the PBS server

4.4.2 Installation of PBS with License Server

During installation of server and scheduler or MOM, you will be prompted with the following:

```
PBS Professional version 9.0 and later is licensed
via the Altair License Manager.

The Altair License Manager can be downloaded from:
http://www.pbspro.com/UserArea/Software/

For more information, please refer to the PBS
Professional Administrator's Guide, or contact pbssup-
port@altair.com.

Continue with the installation? (y or n)
```

The PBS install program will prompt you for a list of license server file locations:

```
Please enter the list of Altair License File location(s), in
a space-separated list of entries of the form:
     <port>@<host>
     @<host>
     <license-file-path>
```

The install program then lists some examples:

```
Examples:
     @fest
     27100@aspasia
     @perikles 27000@aspsaia
     @127.3.4.5
     /usr/local/altair/security/altair_lic.dat

     Enter License File Location(s):  <>
```

NOTE: The "Enter License File Location(s):" is the actual prompt; the lines above it are informational.

The install program will take care of setting the `pbs_license_file_location` attribute to the file location(s) entered.

If you do not input any entry for the "Enter License File Location(s)" prompt, then the install program will not initialize the `pbs_license_file_location` attribute. This means the `pbs_license_file_location` attribute value is left as is, which could be set to some previous value or unset. It is usually set to some previous value when doing an overlay or migration upgrade.

If the license file location is incorrectly initialized (e.g. the host name or port number is incorrect), PBS may not be able to pinpoint the misconfiguration as the cause of the failure to reach a license server. The PBS server's first attempt to contact the license server will result in the following message on the server's log file:

```
"unable to connect to license server at host <H>, port <P>.
```

4.5 Installation Considerations

4.5.1 Amount of Memory in Complex

If the sum of all memory on all vnodes in a PBS complex is greater than 2 terabytes, then the Server (pbs_server) and Scheduler (pbs_sched) must be run on a 64-bit architecture host, using a 64-bit binary.

4.5.2 Adequate Space for Logfiles

PBS logging can fill up a filesystem. For customers running a large number of array jobs, we recommend that the filesystem where $PBS_HOME is located has at least 2 GB of free space for log files. It may also be necessary to rotate and archive log files frequently to ensure that adequate space remains available. (A typical PBS Professional complex will generate about 2 GB of log files for every 1,000,000 subjobs and/or jobs.)

4.5.3 Installing on Multiple Machines

Instead of running the installer by hand on each machine, you can use a command such as pdsh with NFS-mounted common directories on all hosts to distribute the installation to each host. The general form to distribute commands to a large number of hosts is

```
# for hosts node001-005
pdsh -w node[001-005] command
```

When using the PBS INSTALL command you can make a simple text file called “answers” with the answers to the installation prompts, and redirect them into the INSTALL script:

```
./INSTALL < answers
```

where `answers` contains the answers for a MOM-only installation:

```
2
y
<server name>
y
```

To feed your file of answers to the install script:

```
pdsh -w node[001-005] ./INSTALL < answers
```

4.6 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 403.

4.6.1 Default Installation Locations

The default installation directories for PBS are determined by the operating system being used. The directories are shown in the following table.

During installation, if pbs.conf or the administrator specifies the location of PBS_HOME, PBS_HOME will be put there.

OS	Location of PBS_HOME	Location of PBS_EXEC
AIX	`/usr/local/spool/PBS`	`/usr/local/pbs`
bluegene	`/var/spool/PBS`	`/usr/pbs`
HP-UX	`/usr/spool/PBS`	`/usr/local/pbs`
IRIX	`/usr/spool/PBS`	`/usr/pbs`
Linux	`/var/spool/PBS`	`/usr/pbs`
NEC	`/var/spool/PBS`	`/usr/local/pbs`
OSF1	`/usr/spool/PBS`	`/usr/local/pbs`
Solaris	`/usr/spool/PBS`	`/opt/pbs`
Tru64	`/usr/spool/PBS`	`/usr/local/pbs`
Windows	`C:\Program Files\PBS Pro\home`	`C:\Program Files\PBS Pro\exec`

4.7 Pathname Conventions

During the installation process, you will be prompted for the location into which to install the various components of PBS. In this document, we use two abbreviations to correspond to installation locations. The term PBS_HOME refers to the location where the daemon/service configuration files, accounting logs, etc. are located. The term PBS_EXEC refers to the location where the executable programs were installed. Furthermore, directory and file pathnames used in this manual are written such that they can be interpreted on either UNIX or Windows systems. For example, the path reference "PBS_HOME/bin/pbs_server" represents either:

(UNIX) `$PBS_HOME/bin/pbs_server`
or
(Windows) `"%PBS_HOME%/bin/pbs_server"`

where the double quotes in the Windows case are necessary to handle both white space and the forward slash.

4.8 Installation on UNIX/Linux Systems

This section describes the installation process for PBS Professional on UNIX and Linux systems.

4.8.1 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_9.1.0
```

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 or the **PBS Professional Quick Start Guide**.)

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_9.1.0-arch.tar.gz
tar -xvf /tmp/pbspro_9.1.0-arch.tar
cd PBSPro_9.1.0
```

4.8.2 Installation Overview

First, the FLEXlm license server must be installed and configured. See section 4.3 "License Server Installation" on page 36. Then you can install and configure PBS Professional. After installing PBS Professional, you must set the PBS server's pbs_license_file_location attribute to point to the FLEXlm license server. See section 4.12 "Setting the pbs_license_file_location Attribute" on page 80.

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.

Important: Most operating systems allow you to specify an alternative location for the installation of the PBS Professional software binaries (`PBS_EXEC`) and private directories (`PBS_HOME`). Such locations should be owned and writable by root, and not writable by other users. (See Appendix C of this manual for a complete listing of all file permissions and ownerships.)

The following example 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 complex. There are three possibilities: a Server host, an execution host, or a client host. If you are going to run all the PBS components on a single host, install the full Server package (option 1). If you are going to have a complex 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 execution hosts in the complex. 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.

4.8.3 Using pkgadd on Solaris

The default behavior of a PBS Professional installation on Solaris will prompt for the package base directory. The pkgadd administration settings on your system will determine how the installation behaves. The pkgadd command reads an administration file, which determines whether it performs various checks or prompts.

A default administration file is shipped with the SunOS operating system. The file is in /var/adm/install/admin/default. The file has these contents:

```
#ident "@(#)default
1.4 92/12/23 SMI"       /* SVr4.0 1.5.2.1       */
mail=
instance=unique
partial=ask
runlevel=ask
idepend=ask
rdepend=ask
space=ask
setuid=ask
conflict=ask
action=ask
basedir=default
```

You may edit this file, or specify another via the -a option to the pkgadd command. For more information, see the admin(4) man page.

You can specify how the base directory will be derived during installation via the basedir parameter in the administration file. You can leave the default, or set it to one of the following:

* ask (always ask for a base directory)
* An absolute path name
* An absolute path name containing the $PKGINST construction (always install to a base directory derived from the package instance)

Warning: If the pkgadd command is called with the argument -a none, it always asks for a base directory, and sets all parameters in the file to the default value of quit. This can cause problems.

4.8.4 Installation on a Standalone 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 components will run, and from which users will submit jobs. An example of such a system is an SGI Altix. To choose this, we select option **1** to the question shown in the example above.

Important: Some systems' installation programs (e.g. Solaris `pkgadd`) will ask you to confirm that it is acceptable to install setuid/set-

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

## Executing postinstall script.
*** PBS Installation Summary
***
*** PBS Server has been installed in /opt/pbs/sbin.
*** 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.
*** PBS Scheduler has been installed in /opt/pbs/sbin.
***
Installation of <pbs64> was successful.
```

4.8.5 Installing on a Linux Machine

Step 1 Download the PBS tar.gz package to /tmp

Step 2 Change directory to /tmp:

```
cd /tmp
```

Step 3 Extract the tarfile:

```
tar zxvf PBSPro_9.1.0-linux26_i686.tar.gz
```

Step 4 Change directory:

```
cd PBSPro_9.1.0
```

Step 5 Execute INSTALL

```
./INSTALL
```

Step 6 Answer the questions and set the server's pbs_license_file_location attribute by giving the location of the license file or license server. See section 4.12 "Setting the pbs_license_file_location Attribute" on page 80.

Step 7 Start PBS:

```
/etc/init.d/pbs start
```

Step 8 Check to see that the server, scheduler and MOM daemons are running:

```
ps -ef | grep pbs
```

You should see that there are three daemons running: `pbs_mom`, `pbs_server`, `pbs_sched`

Step 9 Test that a normal user can submit a job:

```
echo "sleep 60" | /usr/pbs/bin/qsub
```

This will submit a job in the 'workq' queue because it is the default queue defined within qmgr

Step 10 Verify that the jobs are running:

```
/usr/pbs/bin/qstat -an
```

4.8.6 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 (as most sites tend *not* to want to run batch jobs on the front-end vnode). The MOM is then installed on each execution host within the cluster.

In either case, you will need to run the INSTALL program multiple times in order to install PBS Professional on your cluster system. (Alternatively, if all execution hosts are identical, you could install one of the execution hosts, and then distribute the installation to other hosts via a program such as rdist, or via tar plus scp/rcp.)

First, install PBS on the cluster's front-end machine, following the instructions given in section 4.8.4 "Installation on a Standalone System" on page 57. Enter "**no**" when asked if you want to start PBS. Then, if you do *not* want to run batch jobs on the front-end host, edit the newly installed /etc/pbs.conf file, setting PBS_START_MOM=0, indicating that you do not want a PBS MOM started on this system.

Lastly, start the PBS software on the Server machine by running the PBS startup script, the location for which varies depending on system type. (See "Starting and Stopping PBS: UNIX and Linux" on page 405.)

Next, create the list of machines PBS will manage. Use the qmgr command to add each execution machine in your cluster. See section 7.1 "The qmgr Command" on page 173.

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

4.8.7 PBS man Pages on SGI Irix Systems

If PBS is being installed on SGI systems, it is recommended that you verify that /usr/bsd/ is in the MANPATH setting for users and administrators in order to locate and use the PBS man pages.

4.8.8 Installing on IBM Blue Gene

The Blue Gene system is made up of one service node, one or more front-end nodes, a shared storage location (referred to as the CWFS -- cluster wide file system), dozens or hundreds of I/O nodes, thousands of compute nodes, and various networks that keep everything together. The front-end node, service node, and I/O node run the Linux SUSE Enterprise 9 OS; the compute node runs a lightweight OS called OSK.

Run the PBS Professional server/scheduler/clients on one of the Blue Gene front-end nodes, and run MOM on the service node. The front-end node and service node are running Linux SuSE 9 on an IBM power processor server. There's no need to allow submission of jobs from a non-front end, non-IBM machine (e.g. desktop).

The Blue Gene PBS packages are named:

```
1     PBSPro_9.1.0-linux26_ppc64-bgl64r2.tar.gz
2     PBSPro_9.1.0-linux26_ppc64-bgl64r3.tar.gz
```

The standard `pbs_mom` is replaced by the Blue Gene `pbs_mom`, and the standard `pbs_mom` is saved as "`pbs_mom.standard`". This includes both the 64-bit `pbs_mom` compiled against the V1R2M1 Blue Gene software, and the 64-bit `pbs_mom` compiled against V1R3M0 Blue Gene software.

For a typical installation, install the server/scheduler/clients on the Blue Gene front-end node and MOM on the Blue Gene service node.

1. Before installing, determine the version of the Blue Gene software running on your site. Check whether it's running V1R2* or V1R3*. This can be determined by checking this link:

   ```
   ls -l /bgl/BlueLight/ppcfloor
   ```

2. Choose the PBS Professional Blue Gene package to use.

3. Install the PBS Professional package on the Blue Gene front-end node specifying the "server" type of installation. You can ignore the following error if it is encountered during installation:

   ```
   /etc/init.d/pbs
   Starting PBS
   /usr/pbs/sbin/pbs_mom: error while load-
   ing shared libraries: libdb2.so.1: cannot
   open shared object file: No such file or
   directory
   PBS mom
   ```

   The reason for the above is that a Blue Gene MOM was not

started on the service node. If you need to run a regular MOM on the front-end node, then use `pbs_mom.standard`:

```
cd /usr/pbs/sbin
cp pbs_mom pbs_mom.bgl
ln -s pbs_mom.standard pbs_mom
/etc/init.d/pbs restart
```

4 Install the PBS Professional package on the Blue Gene service node, specifying the “Execution host” type of installation, and the hostname of the front-end node in step 3 must be the PBS_SERVER to talk to.

5 On the Blue Gene service node, wrap the Blue Gene `mpirun`. If you wish to limit mpirun so that it will only execute inside the PBS environment, wrap the mpiruns on the front-end node and the service node by specifying pbsrun_wrap -s, to ensure no Blue Gene partitions are spawned outside of PBS. See section 11.10.6 “The pbsrun_wrap Mechanism” on page 440.

```
/usr/pbs/bin/pbsrun_wrap [-s]\
    /bgl/BlueLight/ppcfloor/bglsys/bin/mpirun \
    pbsrun.bgl
```

WARNING: If the Blue Gene service node and the front-end node both NFS mount the same /bgl, then the /bgl/BlueLight/ppcfloor/bglsys/bin/mpirun on the front-end node will end up with the dead link to pbsrun.bgl. This prevents users on the front-end node from being able to run mpirun outside of PBS, but the wrapped mpirun on the service node will continue to function.

6 On the server host in step 4, add this service node hostname to the list of nodes:

```
qmgr
Qmgr: create node <service_node_hostname>
```

7 The following is recommended if Blue Gene `mpirun` was configured to run with rsh (See section 11.11.3 “Configuration on Blue Gene” on page 454):

On the service node host, add the following to the /etc/

hosts.equiv file:

```
<service_node_hostname>
```

8 If you also want pbs_mom on the service node to copy output files back to the submission host, which is the front-end host, add the same entry to the /etc/hosts.equiv file on the front-end host.

4.8.9 Uninstalling on IBM Blue Gene

If Blue Gene's mpirun was wrapped, be sure to unwrap it via pbsrun_unwrap. Otherwise, if PBS was uninstalled but pbsrun_unwrap wasn't called, then to manually restore Blue Gene's mpirun, simply do:

```
cd /bgl/BlueLight/ppcfloor/bglsys/bin
```

Make sure that this is a symbolic link to $PBS_EXEC/bin/pbsrun.bgl:

```
ls -l mpirun
rm mpirun
mv mpirun.actual mpirun
```

4.8.10 Installing on AIX

When you download the AIX package, you will get two versions of pbs_mom. One is for using with the HPS switch, and the other is the standard pbs_mom. The installer automatically looks for the HPS switch, and if it finds it, installs the version of pbs_mom that manages the switch. If the installer doesn't find the switch, it installs the standard pbs_mom.

4.8.11 Installing on an Altix Running SuSE

1 Download the PBS tar.gz package to /tmp.

2 Change directory to /tmp:

```
cd /tmp
```

3 Extract from the package:

```
tar zxvf \
 PBSPro_9.1.0-linux26_ia64_altix.tar.gz
```

4 Change directories:

```
cd PBSPro_9.1.0
```

5 Execute installation script:

```
./INSTALL
```

6 Answer the questions; supply the license string; do not start the daemons yet.

7 Change directories:

```
cd /usr/pbs/sbin
```

8 Rename the standard PBS MOM:

```
mv pbs_mom pbs_mom.bak
```

9 Copy the cpuset PBS MOM to `pbs_mom`:

```
cp -rp pbs_mom.cpuset pbs_mom
```

10 Start PBS. If the PBS startup script is not used on the Altix, pbs_mom will not start:

```
/etc/init.d/pbs start
```

11 Check to see that vnode definitions for pbs_mom have been generated:

```
/usr/pbs/sbin/pbs_mom -s list
```

If you are using ProPack 2 or 3, you will need to generate the vnode definitions file manually. Follow the steps in section 6.5.6.13 “Generate Vnode Definitions File for ProPack 2, 3” on page 134.

12 Check to see that the PBS daemons are running. You should see that there are three daemons running: `pbs_mom, pbs_server, pbs_sched`:

```
ps -ef | grep pbs
```

13 Submit jobs as a normal user.

Submit a job to the default queue:

```
echo "sleep 60" | /usr/pbs/bin/qsub
```

14 Verify that the jobs are running:

```
/usr/pbs/bin/qstat -an
```

4.8.12 Installing MOM with SGI cpuset Support

PBS Professional for SGI systems provides site-selectable support for IRIX and Altix cpusets. A cpuset in an SGI system is a named region containing a specific set of CPUs and associated memory. PBS uses the cpuset feature to "fence" PBS jobs into their own cpusets. This helps to prevent jobs from interfering with each other. In order to use this feature, you must run a different PBS MOM binary. Stop the MOM, follow the steps shown below, and then run this new `pbs_mom`. (See also section 11.4 "Starting and Stopping PBS: UNIX and Linux" on page 405.)

You must **copy**, not move, `pbs_mom`.cpuset to `pbs_mom`. If `pbs.conf` is not in `/etc`, look at the `PBS_CONF_FILE` environment variable for its location. Look in `pbs.conf` for the location of `$PBS_EXEC`.

```
cd $PBS_EXEC/sbin
rm pbs_mom
cp pbs_mom.cpuset pbs_mom
```

Additional information on configuring and using SGI cpusets can be found in section 8.9 "Configuring MOM for Machines with cpusets" on page 291.

4.9 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 canonical host name is used to authenticate messages, and 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.

Port numbers can be set via /etc/services, the command line, or in pbs.conf. If not set by any of these means, they will be set to the default values. The PBS components and the commands will attempt to use the system `services` file to identify the standard port numbers to use for communication. If the port number for a PBS service can't be found in the system file, a default value for that service will be used. Each daemon has startup options for setting port numbers. See the manual pages pbs_mom(8B), pbs_sched(8B), pbs_server(8B).

The table below shows the valid PBS service names together with their default port numbers for that service.

Table 1: Ports Used by PBS Daemons

Daemon Listening at Port	Port Number	Protocol	Type of Communication
pbs_server	15001	TCP	All TCP communication to server
pbs_server	15001	UDP	Server to MOM via RPP
pbs_mom	15002	TCP	MOM to/from Server
pbs_resmon	15003	TCP	MOM resource requests
pbs_resmon	15003	UDP	MOM resource requests
pbs_sched	15004	TCP	PBS Scheduler
pbs_mom_globus	15005	TCP	MOM Globus
pbs_mom_globus	15006	TCP	MOM Globus resource requests
pbs_mom_globus	15006	UDP	MOM Globus resource requests

The scheduler uses any any privileged port (less than 1024) as the outgoing port to talk to the Server.

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 PBS. If you change them, be careful to use the same numbers on all systems. The port number for pbs_resmon must be one higher than for pbs_mom.

Communication between the scheduler and server is via TCP.

4.10 Installation on Windows 2000 and XP Systems

When PBS is installed on a complex, the MOM must be run on each execution host. The Server and Scheduler only need to be installed on one of the hosts or on a front-end system. For Windows 2000 and XP clusters, PBS is provided in a single package containing:

PBS Professional Quick Start Guide in PDF format,
PBS Professional Administrator's Guide in PDF format,
PBS Professional User's Guide in PDF format,
PBS Professional software, and
supporting text files (software license, README, release notes, etc.)

4.10.1 PBS Windows Considerations

PBS Professional is supported on the following operating systems: Windows 2000 Pro, Windows XP Pro, and both Windows 2000 Server and Windows 2003 Server if the domain controller server configured "native". While PBS Professional supports Active Directory Service domains, it does not support Windows NT domains. Running PBS in an environment where the domain controllers are configured in "mixed-mode" is not supported.

For Windows 2003 Server, because of its enhanced security, only jobs with passwords are allowed (see the discussion of Windows security in section 3.7.2 "Windows Configuration in a Domained Environment" on page 22 and the single-signon feature discussed in section 7.15 "Password Management for Windows" on page 240).

Important: Install PBS Professional from an Administrator account.

Before you install PBS on Windows, make sure you are using the correct type of account. See section 3.7.2 “Windows Configuration in a Domained Environment” on page 22.

PBS Professional requires that the drive that PBS was installed under (e.g. \Program Files\PBS Pro") be configured as an NTFS filesystem.

Before installing PBS Professional, be sure to uninstall any old PBS Professional files. For uninstalling versions 5.4.2 through 8.0, use a domain admin account. For details see “Uninstalling PBS Professional on Windows” on page 77.

You can specify the destination folder for PBS using the “Ask Destination Path” dialog during setup. After installation, icons for the `xpbs` and `xpbsmon` GUIs will be placed on the desktop and a program file menu entry for PBS Professional will be added. You can use the GUIs to operate on PBS or use the command line interface via the command prompt.

This version of PBS Professional for Windows includes both `pbs_rcp` and `pbs_rshd` for allowing copy of output/error files from remote hosts to local Windows host.

4.10.2 Pre-installation Configuration

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

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 be installed on all the vnodes in the complex 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 errors for PBS. Make sure the following are done:

   a. Configure your system to talk to a properly configured and functioning DNS server

   b. Add the correct host entries to the following files:

win2000: `c:\winnt\system32\drivers\etc\hosts`

winXP: `c:\windows\system32\drivers\etc\hosts`

For example, if your Server is fifi.forway.com with address 192.0.0.231, then add the entry:

```
192.0.0.231 fifi.forway.com fifi
```

2. Set up any user accounts that will be used to run PBS jobs. They should not be Administrator-type of accounts, 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
   ```

4.10.3 Installation Account vs. Service Account

There are two accounts used when installing PBS: an *installation account*, and a *service account*. The installation account is that from which you will execute the PBS `install` program; the service account will actually run the PBS services: `pbs_server`, `pbs_mom`, `pbs_sched`, and `pbs_rshd`. The service account is also recommended for performing any updates of the PBS configuration files. For installation in a domained

environment, see section 3.7.2.2 “Installation Account” on page 23 and section 3.7.2.3 “The PBS Service Account” on page 23. For installation in a standalone environment, see section 3.7.3.2 “Installation Account” on page 27 and section 3.7.3.3 “The pbsadmin Service Account for Standalone Environments” on page 27.

4.10.4 Software Installation

Next you will need to install the PBS software on each execution host of your complex. The PBS Professional installation program will walk you through the installation process.

Important: PBS must be installed from a local or domain account. A local account has local administrator privilege on the install host (i.e. the account must be a member of the local "Administrators" group on the local machine). A domain account has either domain administrator privileges (i.e. a member of "Domain Admins" group) or is a member of the local Administrators group on the local (install) host.The `install` program must be executed from a domain account which is a member of the “Domain Admins”. See section 3.7 “Recommended PBS Configurations for Windows” on page 21.

1. If you are installing from the PBS Professional CD-ROM, insert the CD-ROM into your computer’s CD-ROM drive, browse to your CD-ROM drive, and click on the PBS Professional program icon.
   Alternatively, you can download the latest PBS Professional package from the PBS Web site, and save it to your hard drive. Run the self-extracting `pbspro.exe` package, and then the installation program, as shown below.

```
Admin> PBSPro_9.1.0-windows.exe
```

Important: On Windows XP, Service Pack 2 (SP2), upon launching the installer, a window may be displayed saying the program is from an unknown publisher. In order to proceed with the installation of PBS, click the “Run” button.

2. Review and accept the License Agreement, then click Next.

3. Supply your Customer Information, then click Next.

4. Review the installation destination location. You may change it to a location of your choice, provided the new location meets the requirements stipulated in section 3.7 "Recommended PBS Configurations for Windows" on page 21. Then click Next.

5. When installing on an execution host in the complex, select the "Execution" option from the `install` tool, then click Next.

6. You will then be prompted to enter a password for the special "pbsadmin" account (as discussed in the previous section). The password typed will be masked with "*". An empty password will not be accepted. Enter your chosen password twice as prompted, then click Next.

   You may receive the following "error 2245" when PBS creates the pbsadmin account. This means "The password does not meet the password policy requirements. Check the minimum password length, password complexity and password history requirements."

Important: You *must* use the same password when installing PBS on additional execution hosts as well as on the PBS Server host.

7. The installation tool will show two screens with informative messages. Read them; click Next on both.

8. On the "Editing PBS.CONF file" screen, specify the hostname on which the PBS Server service will run, then click Next.

9. On the "Editing HOSTS.EQUIV file" screen, follow the directions on the screen to enter any hosts and/or users that will need access to this local host. Then click Next.

10. On the "Editing PBS MOM config file" screen, follow the directions on the screen to enter any required MOM configuration entries (as discussed in section 8.2.2 "Syntax and Contents of Default Configuration File" on page 260). Then click Next.

11. Lastly, when prompted, select Yes to restart the computer and

click Finish.

Repeat the above steps for each execution host in your complex. When complete you are ready to install PBS Professional on the host that will become the PBS Server host.

1. Install PBS Professional on hostA, selecting the "All" option. Next, you will be prompted for your software license file or server location. 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, clicking Next to move between the different input screens.

2. Finally, run `pbsnodes -a` on hostA to see if it can communicate with the execution hosts in your complex. If some of the hosts are seen to be down, then go to the problem host and restart the MOM, using the commands:

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

4.10.5 Post Installation Considerations

The installation process will automatically create the following file,

```
[PBS Destination folder]\pbs.conf
```

containing at least the following entries:

```
PBS_EXEC=[PBS Destination Folder]\exec
PBS_HOME=[PBS Destination Folder]\home
PBS_SERVER=server-name
```

where `PBS_EXEC` will contain subdirectories where the executable and scripts reside, `PBS_HOME` will house the log files, job files, and other processing files, and *`server-name`* will reference the system running the PBS Server. The `pbs.conf` file can be edited by calling the PBS program "`pbs-config-add`". For example,

```
\Program Files\PBS Pro\exec\bin\pbs-config-add "PBS_SCP=\winnt\scp.exe"
```

Don't edit `pbs.conf` directly as the permission on the file could get reset causing other users to have a problem running PBS.
The auto-startup of the services is controlled by the PBS `pbs.conf` file as well as the Services dialog. This dialog can be invoked via selecting `Settings->Control Panel->Administrative Tools->Services`. If the services fail to start up with the message, "incorrect environment", it means that the *PBS_START_SERVER*, *PBS_START_MOM*, and *PBS_START_SCHED* `pbs.conf` variables are set to 0 (false).

Upon installation, special files in PBS home directory are set up so that some directories and files are restricted in access. The following directories will have files that will be readable by the \\Everyone group but writable only by Administrators-type accounts:

PBS_HOME`/server_name`
PBS_HOME`/mom_logs/`
PBS_HOME`/sched_logs/`
PBS_HOME`/spool/`
PBS_HOME`/server_priv/accounting/`

The following directories will have files that are only accessible to Administrators-type of accounts:

PBS_HOME`/server_priv/`
PBS_HOME`/mom_priv/`
PBS_HOME`/sched_priv/`

Important: The PBS administrator should review the recommended steps for setting up user accounts and home directories, as documented in section 3.8 "Windows User Authorization" on page 30, and Chapter 3 of the **PBS Professional User's Guide**.

4.10.6 Windows XP SP2 Firewall

Under Windows XP service pack 2 (SP2) the Windows Firewall may have been turned on by default. If so, it will block incoming network connections to all services including PBS. Therefore after installing PBS Professional, to allow `pbs_server`, `pbs_mom`, `pbs_sched`, and `pbs_rshd` to accept incoming connections:

Access `Settings->Control Panel->Security Center->Windows Firewall`, and verify that the Windows Firewall has been set to "ON" to block incoming network connections.

From this panel, you can either turn Windows Firewall "off", or click on the `Exceptions` tab and add the following to the list:

```
[INSTALL PATH]\exec\sbin\pbs_server.exe
[INSTALL PATH]\exec\sbin\pbs_mom.exe
[INSTALL PATH]\exec\sbin\pbs_sched.exe
[INSTALL PATH]\exec\sbin\pbs_rshd.exe
```

where `[INSTALL PATH]` is typically `C:\Program Files\PBS Pro`

4.10.7 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`, which is what PBS uses for delivering job output and error files to destination hosts. (Keep in mind that `pbs_rshd` does not allow normal `rsh` activities but only `rcp`.)

`pbs_rshd` will read either the `%WINDIR%\system32\drivers\etc\hosts.equiv` file or the user's `.rhosts` file for determining the list of accounts that are allowed access to the localhost during remote file copying. PBS uses this same mechanism for determining whether a remote user is allowed to submit jobs to the local Server. `pbs_rshd` is started automatically during installation but can also be started manually by typing either of the following two commands:

```
net start pbs_rshd
```

-or-

```
pbs_rshd -d
```

This latter form of invocation runs `pbs_rshd` in debug mode where logging output will be displayed on the command line.

If user on *hostA* uses `pbs_rcp` to copy a file to *hostB* (running `pbs_rshd`) as shown:

```
pbs_rcp file1 hostB:file2
```

the behavior will be as follows. 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 *hostB's* `hosts.equiv` file; or (2) *userA@hostA* is authenticated via user's `[PROFILE_PATH]/.rhosts` on *hostB*. (See also section 3.8.2 "Windows User's HOMEDIR" on page 32.)

The format of the `hosts.equiv` file is:

```
[+|-] hostname username
```

'+' means enable access whereas '-' means to disable access. If '+' or '-' is not specified, then this implies enabling of access. If only *hostname* is given, then users logged into that host are allowed access to like-named accounts on the local host. If only *username* is given, then that user has access to all accounts (except Administrator-type users) on the local host. Finally, if both *hostname* and *username* are given, then user at that host has access to like-named account on local host.

The format of the user's *`.rhosts`* file is simply:

```
hostname username
```

The `hosts.equiv` file is consulted first and then, if necessary, the user's `.rhosts` file is checked. If *username* contains special characters like spaces, be sure to quote it so that it will be properly parsed by `pbs_rshd`:

```
hostname "username"
```

For the above `pbs_rcp` request, you will either need the system-wide `hosts.equiv` file on *hostB* to include as one of its entries:

```
hostA
```

or, `[PROFILE_PATH]\.rhosts` on *userA's* account on *hostB* to include:

```
hostA userA
```

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

```
pbs_rcp file1 userB@hostB:file2
```

then use of the account's `[PROFILE_PATH]\.rhosts` file is the only way to authenticate, and it needs to have the entry:

```
hostA userA
```

These two methods of authentication are further discussed in the **PBS Professional User's Guide**.

4.10.8 Network Drives and File Delivery

If users require jobs to have output or error files going into some network location, and that network location is mapped to the same local drive (for instance drive `Q`), then you need to put the following two lines in MOM's config file. (For additional information on MOM configuration parameters, see section 8.2.2 "Syntax and Contents of Default Configuration File" on page 260.)

```
$usecp *:Q: Q:
$usecp *:q: q:
```

The above causes any job output or error files having the form, "`<hostname>:` the letter "q"`:`*`file-path`*" to be passed to `xcopy` as:

`Q:`*`file-path`* or `q:`*`file-path`*

instead of being passed to `pbs_rcp/pbs_rshd`.

The reason for putting a wildcard entry for *hostname* in `$usecp` is to get around the possibility of MOM seeing different permutations of hostname for the destination host. The upper and lower cases of "q" are needed in order to get a match in all possible situations.

The example above will result in the following translations:

```
pbs_rcp job_output_file host2:Q:\output
```

is translated to: `xcopy job_output_file Q:\output`

```
pbs_rcp job_output_file host3.test.domain.com:Q:\output
```

is translated to: `xcopy job_output_file Q:\output`

```
pbs_rcp job_output_file host4.domain.com:q:\output
```

is translated to: `xcopy job_output_file q:\output`

4.10.9 Changing the pbsadmin Password

Normally, the "pbsadmin" password must not be changed. But if it is deemed necessary to change it perhaps due to a security breach, then do so using the following steps:

First, change the "pbsadmin" service account's password on a machine in a command prompt from an admin-type of account by typing:

domain environments:

```
net user pbsadmin * /domain
```

non-domain environment:

```
net user pbsadmin *
```

Then the Service Control Manager (SCM) must be provided with the new password specified above. This can be done via the GUI-based Services application found as one of the Administrative Tools, or unregister and re-register the PBS services with password:

```
pbs_account --unreg "\Program Files\PBS Pro\exec\sbin\pbs_server.exe"
pbs_account --unreg "\Program Files\PBS Pro\exec\sbin\pbs_mom.exe"
pbs_account --unreg "\Program Files\PBS Pro\exec\sbin\pbs_sched.exe"
pbs_account --unreg "\Program Files\PBS Pro\exec\sbin\pbs_rshd.exe"

pbs_account --reg "\Program Files\PBS Pro\exec\sbin\pbs_server.exe"
pbs_account --reg "\Program Files\PBS Pro\exec\sbin\pbs_mom.exe"
pbs_account --reg "\Program Files\PBS Pro\exec\sbin\pbs_sched.exe"
pbs_account --reg "\Program Files\PBS Pro\exec\sbin\pbs_rshd.exe"
```

The register form (last four lines above) can take an additional argument `-p password` so that you can specify the password on the command line directly.

4.10.10 Uninstalling PBS Professional on Windows

For uninstalling the old PBS with version between 5.4.2 and 8.0, an account that is a member of the "Domain Admins" group will still have to be used.

To remove PBS from a Windows system, either (1) Go to `Start Menu->Settings->Control Panel->Add/Remove Program` (Win2000) or `Start Menu->Control Panel->Add/Remove Programs` (Windows XP) menu and select the PBS Professional entry and click "Change/Remove"; or (2) double click on the PBS Windows installation package icon to execute. This will automatically delete any previous installation.

If the uninstallation process complains about not completely removing the PBS installation directory, then remove it manually, for example by typing:

```
cd \Program Files
rmdir /s "PBS Pro"
```

Under some conditions, if PBS is uninstalled by accessing the menu options (discussed above), the following error may occur:

```
...Ctor.dll: The specified module could not be found
```

To remedy this, do the uninstall by running the original PBS Windows installation executable (e.g. `PBSPro_9.1.0-windows.exe`), which will remove any existing instance of PBS.

During uninstallation, PBS will not delete the "pbsadmin" account because there may be other PBS installations on other hosts that could be depending on this account. However, the account can be deleted manually using an administrator-type of account as follows:

In a domain environment:

```
net user pbsadmin /delete /domain
```

In a non-domain environment:

```
net user pbsadmin /delete
```

At the end of uninstallation, it is recommended to check that the PBS services have been completely removed from the system. This can be done by opening up the Services dialog:

(Windows 2000):

```
Start Menu->Settings->Control Panel->Administrative Tools->
Services
```

(Windows XP):

```
Start Menu->Control Panel->Performance and Maintenance->
Administrative Tools->Services
```

and check to make sure PBS_SERVER, PBS_MOM, PBS_SCHED, and PBS_RSHD entries are completely gone. If any one of them has a state of "DISABLED", then you must restart the system to get the service removed.

4.11 Post Installation Validation

If you wish to validate the installation of PBS Professional, 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 12.4 “The pbs_probe Command” on page 494. (The `pbs_probe` command is not available under Windows.)

Use the `qstat` command to find out what version of PBS Professional you have.

```
qstat -fB
```

4.12 Setting the pbs_license_file_location Attribute

This points the pbs_license_file_location server attribute to the license server(s) in your installation. See section 5.4.3.1 “Setting the License File Location in pbs_license_file_location” on page 88.

Set the server’s `pbs_license_file_location` attribute to point to the license file or license server:

```
Qmgr> set server pbs_license_file_location=\
<installation location>/altair/security/altair_lic.dat
```

If you are using a three-server redundant license server arrangement, follow the steps in section 5.5.1 “To Use the Three-server Redundant Setup” on page 93.

4.12.1 Trial Licenses

If you obtained a trial license, then you have to ensure that:

1. The `pbs_license_file_location` server attribute is unset
2. No value is entered into the "Enter License File Location(s)" prompt of the install program.
3. The trial license must still be put in PBS_HOME/server_priv/license_file either manually (Windows) or by calling pbs_setlicense (Linux/Unix only). PBS must be restarted after this.

See section 5.2 “Trial Licenses” on page 82.

Chapter 5
Licensing

PBS depends on a FLEX/Altair license server that hands out license tokens, which are Gridworks units. These tokens have a corresponding mapping to PBS CPU licenses to be assigned to PBS jobs.

5.1 FLEX Licensing Feature

5.1.1 Overview of FLEXlm Licensing

PBS is using a FLEX licensing system, employing a FLEXlm/Altair license server. This system makes a number of floating licenses available to run PBS jobs on hosts in a network. PBS uses a units-based or token-based licensing scheme. The number of license tokens needed is proportional to the number of CPUs requested by a job. All PBS licenses are now floating licenses, and do not depend on which host is being used. Licenses are used for the CPUs used by a job, instead of for hosts. It is no longer the hosts that are licensed, it is the CPUs used by a job.

Multiple PBS complexes can use the same FLEXlm license server.

5.1.2 License Server Versions and PBS Releases

Each feature in the license file has a version number associated with it. You need to have a license file in which the feature version is at least as new as the PBS Professional version. You can use PBS Professional when its version is older than or the same as the features in the license file.

You will need a FLEX license file when you upgrade PBS Professional or when your license expires. If you need additional licenses in the mean time, contact Altair Support.

5.1.3 Old Licensing Model Used for Pre-9.0 Versions and Trial Licenses

The old PBS licensing scheme required that a license key be provided to PBS via "PBS_SERVER/server_priv/license_file". This key determined whether a job could be run on a number of CPUs on one or more hosts. This proprietary license model accepted either a regular license or a trial license, which had a short expiration time. For regular licenses, there were 3 types: UNIX/Windows license (type '5'), Linux licenses (type 'L'), and floating licenses (type 'F').

The old PBS proprietary license model will continue to license PBS Professional versions 8.0 and earlier. Customers who upgrade from version 8.0 and earlier to 9.0 and later will need to download, install and configure the FLEX license server. See section 4.2 “FLEX Licensing” on page 36.

5.2 Trial Licenses

The old PBS proprietary license model will only be used for sites needing the temporary, trial licenses (type "T"). Previous (pre-9.0) “T” licenses will also work with PBS version 9.0 and beyond, until the license expiration date. In 9.0, these “T-“ licenses behave like proprietary floating licenses with short expiration times rather than like node-locked licenses with short expiration times.

Trial licenses in PBS versions 9.0 and later:

Float among the hosts in the PBS complex; are no longer node-locked but floating.
Are not bound to the PBS server hostid info.
Are valid for 90 days after installation of PBS (not installation of license.)
Have a defined limit on the number of CPUs which they can license.

If you obtained a trial license, then you simply have to ensure that:

1. The `pbs_license_file_location` server attribute is unset
2. No value is entered into the "Enter License File Location(s)" prompt of the install program.
3. The trial license must still be put in PBS_HOME/server_priv/license_file either manually (Windows) or by calling pbs_setlicense (Linux/Unix only). PBS must be restarted after this.

When the PBS server comes up, if there are both a trial license key and a `pbs_license_file_location` server attribute set, then the latter takes precedence. The server will get the licenses from the external license server. Even if `pbs_license_file_location` has a bad value (i.e. bad <port> or <host>), no attempt will be made to use any trial licenses to run jobs.

The Windows package will continue to ship with a trial license. Under Windows, `pbs_setlicense` is not available and you must edit the license file directly.

5.3 Definitions

Floating License — A unit of license dynamically allocated (checked out) when a user begins using an application on some host (when the job starts), and deallocated (checked in) when a user finishes using the application (when the job ends). The floating licenses discussed in this chapter license PBS.

FLEXlm — Software license manager for granting floating licenses to multiple end-users in a network.

FLEXnet — The new name for FLEXlm.

Vendor Daemon — A FLEXlm concept in which this daemon keeps track of the number of licenses that have been checked out, and who has them. For PBS Professional, the vendor daemon is supplied by Altair and is called "altair_lm".

License Manager Daemon (lmgrd) — Handles the initial contact with the FLEXlm-licensed applications passing the connection on to the appropriate vendor daemon. It also takes care of starting/restarting the vendor daemon.

License Server — Comprised of the FLEXlm daemon (lmgrd) and the vendor daemon.

LICENSE_FILE — A text file created by the software vendor and installed by the license administrator. The license file stores data about the server machines, vendor daemons, and license keys or signatures for each licensed product.

Memory-only Vnode Represents a node board that has only memory resources (no CPUs), for example, an Altix memory-only blade.

hostid An identifier unique to a particular computer. A FLEXlm key is registered against a license server host's hostid, which can be based on the MAC address (physical address of the license server's ethernet card), the host's actual host ID (Solaris), or some Disk Serial Number (Windows).

Token Also called “GridWorks Unit”, a unit of value which is checked out from FLEXlm. The number of PBS tokens will be related to the number of CPUs requested by a job that is being executed.

Redundant License Server Configuration Allows licenses to continue to be available should one or more license servers fail. There are two types: 1) license server list configuration, and 2) three-server configuration.

License Server List Configuration A collection of license server files, or "<port>@<host>" settings, pointing to license server managers servicing Altair licenses. The FLEXlm license application tries each server on the list until it succeeds or gets to the end of the list. There could be X licenses on <server1>, Y licenses on <server2>, and Z licenses on <server3>, and the total licenses available would actually be X+Y+Z, but a request must be satisfied only by one server at a time.

Three-server Configuration Means that if any 2 of the 3 license servers are up and running (referred to as a quorum), the system is functional, with 1 server acting as master who can issue licenses. If the master goes down, then another server must take over as master. This is set up as a license file on each of the 3 redundant servers containing:

SERVER <server1> ... <port1>
SERVER <server2> ... <port2>
SERVER <server3> ... <port3>

PBS Professional can point to a license server host that has these license file entries.

5.4 Configuring PBS for Licensing

5.4.1 Configuring the Server for Licensing

To configure the PBS server for licensing:

Step 1 Specify the license server port and hostname by setting the server's `pbs_license_file_location` attribute. See "Server Licensing Attributes" on page 85 and "Setting Server Licensing Attributes" on page 88.

Step 2 Optionally, specify the minimum number of CPUs to keep permanently licensed by setting the server's `pbs_license_min` attribute. It is recommended that you set this to the total number of CPUs in the complex. This is the total for all `resources_available.ncpus` configured for each vnode. See section 5.4.3.2 "Setting pbs_license_min" on page 90.

Also see section 5.9.1.3 "Licensing and Advance Reservations" on page 103.

Step 3 Optionally, specify the maximum number of CPUs to be licensed at any time by setting the server's `pbs_license_max` attribute.

Step 4 Optionally, specify the number of seconds to keep an unused license before returning it to the pool when there are more than the minimum number of licenses checked out by setting the server's `pbs_license_linger_time` attribute.

Step 5 Configure a redundant license server setup. See section 5.5 "Redundant License Servers" on page 93.

5.4.2 Server Licensing Attributes

There are five server attributes for licensing. They are `pbs_license_file_location`, `pbs_license_min`, `pbs_license_max`, `pbs_license_linger_time`, and `license_count`. They are listed below with other server attributes which control licensing.

pbs_license_file_location

Hostname of license server, or local pathname to the actual license file(s), which is associated with a license server. String. Set by PBS Manager. Readable by all. Default value: empty string, meaning no server to contact.

To set `pbs_license_file_location` to the hostname of the license server:

```
qmgr> set server
pbs_license_file_location=<port1>@<host1>:
<port2>@<host2>:…:<portN>@<hostN>
```

where <host1>, <host2>, …, <hostN> can be IP addresses.

Windows: Use semicolons (;) instead of colons (:), and enclose the path list in double quotes if it contains any spaces or when you are listing more than one port/server. For example:

```
qmgr> set server
pbs_license_file_location="<port1>@<host1>
;<port2>@<host2>;…;<portN>@<hostN>"
```

To set `pbs_license_file_location` to a local path:

```
qmgr> set server
pbs_license_file_location=<path_to_local_l
icense_file>[[:<path_to_local_license_file
2>]:…:<path_to_local_license_fileN>]]
```

To unset `pbs_license_file_location` value:

```
Qmgr> unset server \
    pbs_license_file_location
```

pbs_license_linger_time

The number of seconds to keep an unused CPU license, when the number of licenses is above the value given by

pbs_license_min. Time. Set by PBS Manager. Readable by all. Default: 3600 seconds.

To set pbs_license_linger_time:

```
Qmgr> set server \
    pbs_license_linger_time=<Z>
```

To unset pbs_license_linger_time:

```
Qmgr> unset server \
    pbs_license_linger_time
```

pbs_license_max

Maximum number of licenses to be checked out at any time, i.e maximum number of CPU licenses to keep in the PBS local license pool. Sets a cap on the number of CPUs that can be licensed at one time. Long. Set by PBS Manager. Readable by all. Default: maximum value for an integer.

To set pbs_license_max:

```
Qmgr> set server pbs_license_max=<Y>
```

To unset pbs_license_max:

```
Qmgr> unset server pbs_license_max
```

pbs_license_min

Minimum number of CPUs to permanently keep licensed, i.e. the minimum number of CPU licenses to keep in the PBS local license pool. This is the minimum number of licenses to keep checked out. It is recommended that you set pbs_license_min to the total number of CPUs in your complex. This is the total for all resources_available.ncpus configured for each vnode. Long. Set by PBS Manager. Readable by all. Default: zero.

To set pbs_license_min:

```
Qmgr> set server pbs_license_min=<X>
```

To unset pbs_license_min:

```
Qmgr> unset server pbs_license_min)
```

license_count	license_count= Avail_Global:<X> Avail_Local:<Y> Used:<Z> High_Use:<W> Avail_Global is the number of PBS CPU licenses still kept by the Altair License Server (checked in). Avail_Local is the number of PBS CPU licenses in the internal PBS license pool (checked out). Used is the number of PBS CPU licenses currently in use. High_Use is the highest number of CPU licenses checked out and used at any given time while the current instance of the PBS server is running. "Avail_Global" + "Avail_Local" + "Used" is the total number of CPU licenses configured for one PBS complex. Integer. Set by Server. Readable by all. Default: zero.
FLicenses	The number of floating CPU licenses available to PBS. Equal to the Avail_Global + Avail_Local of the `license_count` attribute. Integer. Set by the server. Readable by all. Default: zero.

5.4.3 Setting Server Licensing Attributes

The administrator can define the following server attributes via qmgr:

5.4.3.1 Setting the License File Location in `pbs_license_file_location`

To set `pbs_license_file_location` to the hostname of the license server(s), run the following as a PBS manager or administrator (i.e. root):

```
qmgr> set server pbs_license_file_location=\
        <port1>@<host1>:<port2>@<host2>:…:<portN>@<hostN>
```

where <host1>, <host2>, …, <hostN> can be IP addresses.

To set pbs_license_file_location to a local path:

```
qmgr> set server pbs_license_file_location=\
        <path_to_local_license_file>\
        [[:<path_to_local_license_file2>]\
        :…:<path_to_local_license_fileN>]]
```

The default value is empty, meaning no external license server is contacted.

These actions follow:

If the pbs_license_file_location is currently set to a non-empty value, then it is set to the new value. All previous licenses are checked back into the previous FLEX server, and the connection to that server is terminated.

If the pbs_license_file_location is currently unset, then it is set to the new value. If the PBS complex was licensed by a trial license key, then the trial licensing is discontinued. An attempt is made to initialize connection to the license server whose location is described in pbs_license_file_location. If the connection fails, qmgr and server_logs contain a warning:

```
"Unable to connect to license server at
pbs_license_file_location=<X>"
```

The attribute is still set to this new value.

Upon successful connection to the license server, PBS tries to re-license the running jobs (which ran previously via trial licenses). Any jobs that are currently running continue to run, even if not all the necessary licenses are obtained for these jobs.

Windows: Use semicolons (;) instead of colons (:), and enclose the path list in double quotes if it contains any spaces or when you are listing more than one port/server. For example:

qmgr> **set server pbs_license_file_location="<port1>@<host1>;<port2>@<host2>;…;<portN>@<hostN>"**

To unset pbs_license_file_location, simply run the following as PBS Manager or Administrator (i.e. root):

```
Qmgr> unset server pbs_license_file_location
```

These actions follow:

The pbs_license_file_location attribute is set to the empty string, previous licenses are checked back into the previous FLEX server, and connection to this FLEX server is shut down.

The PBS server looks for a valid trial license key and re-licenses the vnodes in the complex with floating licenses.

Whether or not there are trial licenses available, currently running jobs are allowed to run to completion.

Once PBS is installed, the license file location can be changed (if, for example, the Altair FLEX server is moved to another host), by setting the following server attribute via qmgr:

```
qmgr> set server \
   pbs_license_file_location=<port1>@<host1>
```

5.4.3.2 Setting pbs_license_min

It is recommended that you set pbs_license_min to the total number of CPUs in your complex. This is the total for all resources_available.ncpus configured for each vnode. To set pbs_license_min, run the following as a PBS manager or administrator (i.e. root):

```
Qmgr> set server pbs_license_min=<X>
```

These actions follow:

If <X> is not numeric, or is less than 0, or is greater than the value of pbs_license_max, then the attribute is not set, and qmgr outputs an error and returns a non-zero value.

The next time PBS updates its license pool (usually every 5 minutes or when a job is started/exited), the new value for pbs_license_min is known internally to the PBS server.

If the server cannot obtain the amount of CPU licenses given by pbs_license_min from the FLEX server, then it will try to obtain as many as possible, log the error, and keep trying to get more up to the correct value over some period of time.

To unset pbs_license_min, run the following as a PBS manager or administrator (i.e. root):

```
Qmgr> unset server pbs_license_min
```

This will cause pbs_license_min to revert to its default value.

The next time PBS updates its license pool (usually every 5 minutes or when a job is started/exited), the value of pbs_license_min will have a default value, and unused licenses are returned to the FLEX server. The return of the license(s) to the pool is constrained by pbs_license_linger_time.

5.4.3.3 Setting pbs_license_max

To set pbs_license_max, run the following as a PBS manager or administrator (i.e. root):

```
qmgr> set server pbs_license_max=<Y>
```

These actions follow:

If <Y> is not numeric, or is less than 0, or is less than the value of pbs_license_min, then the attribute is not set, and qmgr outputs an error and returns a non-zero value.

The next time PBS updates its license pool (usually every 5 minutes or when a job is started/exited), the new value for pbs_license_max is known internally to the PBS server.

If the new value for pbs_license_max is less than the previous value, the value is not adjusted by taking away the licenses that are currently in use by running jobs. This allows the jobs to continue to run. However, as jobs exit, the PBS server will adjust the internal license pool to have no more than the number of licenses given by pbs_license_max.

To unset pbs_license_max, run the following as a PBS manager or administrator (i.e. root):

```
Qmgr> unset server pbs_license_max
```

This causes the pbs_license_max value to revert to its default value. The next time PBS updates its license pool (usually every 5 minutes or when a job is started/exited), the value of pbs_license_max is set to its default.

5.4.3.4 Setting pbs_license_linger_time

To set pbs_license_linger_time, run the following as a PBS manager or administrator (i.e. root):

```
Qmgr> set server pbs_license_linger_time=<Z>
```

These actions follow:

If <Z> is not numeric, or is less than or equal to 0, then the attribute is not set, and qmgr outputs an error and returns a non-zero value.

The next time PBS updates its license pool (usually every 5 minutes or when a job is started/exited), the new value for pbs_license_linger_time is known internally to the PBS server.

To unset pbs_license_linger_time, run the following as PBS manager or administrator (i.e. root):

```
Qmgr> unset server pbs_license_linger_time
```

This causes the value of pbs_license_linger_time to revert to its default. The next time PBS updates its license pool (usually every 5 minutes or when a job is started/exited), the value of pbs_license_linger_time is at the default.

5.4.4 Licensing and PBS Server Failover

The server attribute values for pbs_license_file_location, pbs_license_min, pbs_license_max, and pbs_license_linger_time are set through the primary server. Since these values are saved in PBS_HOME/server_priv/serverdb, and PBS_HOME is in a shared location the secondary server can use these licensing parameters. No additional licensing steps are needed for the secondary server to work properly.

5.5 Redundant License Servers

If you have redundant license servers, some or all of the licenses used by your site will be available if a server host crashes. You can use the PBS Server/Scheduler host as one of the license server hosts. There are two ways to set up your FLEXlm license servers for redundancy.

1. The FLEXlm three-server redundant setup
2. The FLEXlm license server list

In the three-server setup, as long as at least two of the servers are up, PBS can get all the licenses. In this arrangement, each of the servers has the same license file. One server acts as "master".
In the license server list arrangement, each server has some of the licenses, and PBS will try each in turn until it gets the licenses it needs or there are no more servers on the list. This arrangement is better where the network is unreliable.

Set up redundancy before starting PBS.

When choosing license server hosts, choose machines that are stable and unlikely to be rebooted, and that have excellent communication.

5.5.1 To Use the Three-server Redundant Setup

Step 1 Make sure that the three servers are configured correctly according to the guidelines for the FLEXlm three-server arrangement. Each host should already have the license server and license file installed using the steps in section 4.3.2 "Installing the FLEX License Server on UNIX/Linux" on page 37 or section 4.3.3 "Installing the FLEX License Server on Windows" on page 44. Each server has a copy of the license, and the license lists all three servers.

Step 2 Make sure that the PBS server has a file with the contents:

```
SERVER  <host1> ...<port1>
SERVER  <host2>…<port2>
SERVER  <host3>…<port3>
USE_SERVER
```

This file can have any name and any location as long as it resides on the PBS server. It is recommended that you put it in $PBS_HOME/server_priv.

Step 3 Set the server's `pbs_license_file_location` attribute to point to the file described in Step 2:

```
Qmgr> set server \
pbs_license_file_location=\
<location of  three-server host file>
```

5.5.2 To Use the FLEXlm License Server List

Step 1 Make sure that the servers are configured correctly according to the guidelines for the FLEXlm server list. Each host should already have the license server and license file installed using the steps in section 4.3.2 "Installing the FLEX License Server on UNIX/Linux" on page 37 or section 4.3.3 "Installing the FLEX License Server on Windows" on page 44. Each of these servers will have a portion of the total amount of licenses.

Step 2 Use the qmgr command:

```
qmgr> set server \
    pbs_license_file_location=\
    <port1>@<host1> :<port2>@<host2>\
    :<port3>@<host3>:..:<portN>@<hostN>
```

For Windows, use semicolons instead of colons, and enclose the path list in double quotes if any of the paths contains spaces.

5.6 Environment Variables and Licensing

ALTAIR_LM_LICENSE_FILE The environment variable ALTAIR_LM_LICENSE_FILE affects only how Altair applications run inside a PBS job. The variable does not affect the licensing of PBS jobs themselves, and does not affect the value of `pbs_license_file_location`. The ALTAIR_LM_LICENSE_FILE environment variable is set by the server to the same value as the

`pbs_license_file_location` server attribute. You can still set the ALTAIR_LM_LICENSE_FILE environment variable inside the PBS_HOME/pbs_environment file if you wish to affect the licensing of Altair applications invoked inside a PBS job. In the user's PBS job environment, the ALTAIR_LM_LICENSE_FILE variable will not appear unless specifically set in pbs_environment.

ALM_ERROR_TRACKING This environment variable is useful for debugging license checkout failures. When set to an integer greater than 0, all debug messages from the security library will be printed to stdout. This variable only needs to be set on client machines

ALTAIR_LOGFILE The fully qualified name of the Altair debug log file, containing debug information from both lmgrd and altair_lm. This needs to be set on the license server(s), before starting the license server machine(s).

ALTAIR_LOGINTERVAL Sets the time interval in minutes when the usage log will be updated by the license server. By default, the time interval is set to 60 minutes. The minimum time interval is 15 minutes and the maximum time interval is 30 days. This environment variable needs to be set on the license server machine(s), before starting the license server(s).

ALTAIR_NOLOG When set, all logging will be disabled. This needs to be set on the license server machine(s), before starting the license server(s).

FLEXLM_DIAGNOSTICS When set to an integer between 1 and 3, this provides diagnostic information when a checkout fails. On UNIX, the diagnostic message is sent to stderr. On Windows, the diagnostic message is in a file flex<pid>.log in the current working directory.

HW_DEBUG_INFO (UNIX only) When set, this environment variable provides debug information about all environment variables that are set by scripts that launch this application. This environment variable also turns on ALM_ERROR_TRACKING. This variable only needs to be set on the PBS server host.

LM_SERVER_HIGHEST_FD Used to set the highest file descriptor value, above which the license server will not access.

TCP_NODELAY Improves FLEXnet license server system performance when processing license requests. Set to 1 to enable performance enhancements. Use with caution: when enabled, it may cause an increase in network traffic.

5.6.1 Windows Registry

Registry Setting (Windows only)
On Windows, the FLEXlm registry is at HKEY_LOCAL_MACHINE->Software->FLEXlm License Manager ->ALTAIR_LM_LICENSE_FILE

5.7 Replacing Existing Licenses

This section describes how to replace a FLEXlm license, not a pre-9.0 PBS license.

5.7.1 UNIX

Step 1 Go to the license server machine.

Step 2 Replace the existing license in the altair_lic.dat file with the new one. The altair_lic.dat file is located in the <install_location>/altair/security directory. This is where the Altair FLEXlm server is installed.

Step 3 Edit the new altair_lic.dat file by replacing “hostname” in the SERVER line with the server machine's IP address. For redundant server licenses, do this for each SERVER line in the license file.

Step 4 Enter the following at the UNIX command prompt:

```
<install_location>/altair/security/bin/$ARCH/lmutil \
    lmreread -c <license file> -all
```

The license daemon rereads the license file altair_lic.dat. If the reread is successful, the message "lmreread successful" is returned in the UNIX window. See page 67 of the HyperWorks 8.0 Installation Guide.

5.7.2 Windows

Step 1 Go to the license server machine.

Step 2 Replace the existing license in the altair_lic.dat file with the new one. The altair_lic.dat file is located in the <install_location>/security directory. FLEXlm is case- and space-sensitive. The license in the altair_lic.dat file should look exactly as it does in the e-mail.

Step 3 Edit the new altair_lic.dat file by replacing "hostname" in the SERVER line with the server machine's IP address. For redundant server licenses, do this for each SERVER line in the license file.

Step 4 Execute <install_location>/security/lmtools.exe or select Start Menu/Programs/Altair HyperWorks 8.0/Altair Tools/FLEXlm Utilities.

Step 5 In the LMTOOLS dialog, click the Start/Stop/Reread tab.

Step 6 Click the Reread License File button.

Step 7 The license daemon rereads the license file altair_lic.dat. If the reread is successful, a message at the bottom of the dialog box states "Reread Server License File Completed." If the reread is unsuccessful, shut down the server(s) and restart them. If the restart fails, call Altair Support with the error message.

Step 8 Click File.

Step 9 Click Exit to close LMTOOLS.

5.7.3 License Expiration Notice

If the license has an expiration date, the server will log its upcoming expiration 30 days before the expiration date and then each time it checks out licenses.

5.8 Displaying Licensing Information

5.8.1 Viewing License Information in Server Attributes

To see the information in the server attributes including Flicenses, `pbs_license_file_location`, `pbs_license_min`, etc, run:

```
qstat -Bf
```

or

```
qmgr -c "list server"
```

5.8.2 Viewing the Number of Available Licenses/Tokens

If FLEXlm tools such as `lmutil lmstat` are installed on the PBS server host, the user can use them to discover the number of actual tokens/licenses available These are reported in units that are a multiple of the number of CPUs.

5.9 PBS Jobs and Licensing

5.9.1 Examples of Licensing PBS Jobs

The following examples show how PBS licenses jobs.

1. Fit in a single host such as an Altix, packed on the fewest vnodes:

```
qsub -l ncpus=10:mem=20gb -l place=pack
```

This job requests 10 CPUs, so PBS will check out 10 licenses.

2. Request 4 chunks, each with 1 CPU and 4 gb of memory taken from anywhere:

```
qsub -l select=4:ncpus=1:mem=4gb -l place=free
```

The job is requesting 4 CPUs, so PBS will check out 4 licenses.

3. Request 4 vnodes where the arch is linux and each vnode is on a separate host:

```
qsub -l select=4:mem=2gb:ncpus=2:arch=linux \
```

```
-l place=scatter
```

Each vnode has 2 CPUs and 2GB memory allocated to the job. The job requests 8 CPUs so PBS will check out 8 licenses.

4. Request to run on a specific host:

```
qsub -l select=1:ncpus=2:mem=50gb:host=zooland
```

The job requests 2 CPUs, so PBS needs to check out 2 licenses.

5. Request resources on different hosts:

```
qsub -l select=2:ncpus=3:mem=6gb -l place=scatter
```

The job requests 6 CPUs, so PBS will check out 6 licenses to run the job.

6. Cpusets: An odd-size job that will fit on a single Altix, but not on any one nodeboard, and the request is not shared:

```
qsub -l select=1:ncpus=3:mem=6gb -l place=pack:excl
```

The job is specifically requesting 3 CPUs, so PBS needs to check out 3 licenses.

7. Cpusets: Request a small number of CPUs but a large amount of memory, exclusively:

```
qsub -l select=1:ncpus=1:mem=25gb -l place=pack:excl
```

The job is requesting 1 CPU, so PBS checks out 1 license.

8. Cpusets: Align a large job within one router, if it fits within a router.

```
qsub -l select=1:ncpus=100:mem=200gb \
   -l place=pack:group=router
```

The job requests 100 CPUs, requiring PBS to check out 100 licenses.

9. IBM Blue Gene example:

```
qsub -l select=640:ncpus=2 <job.script>
```

This job requests 640*2=1280 CPUs so PBS needs to check out 1280 licenses.

10. Assignment of resources involving memory-only vnodes (nodeboard):

Given 3 vnodes available where
V1 has 4 CPUs and 2 GB
V2 has no CPUs and 3 GB
V3 has no CPUs and 4 GB
V4 has 2 CPUs and 5GB
V1, V2, V3, V4 have sharing attribute set to “default_shared”

Case 1: Job requesting no CPUs:

```
qsub -l select=1:ncpus=0:mem=3gb
```

This job would not run since it asks for no CPUs. No licenses would be checked out. Note that that even though a chunk within a job can ask for 0 CPUs, the job as a whole must request at least one CPU.

Case 2: Job requesting a combination of vnodes with CPUs and memory-only vnodes, non-exclusively:

```
qsub -l select=1:ncpus=1:mem=7gb
```

Even though this job may get assigned 2 GB of V1, 3 GB of V2, and 2 GB of V3, the job has requested 1 CPU so PBS needs to check out only 1 license.

Case 3: Job requesting a combination of vnodes with CPUs and memory-only vnodes EXCLUSIVELY:

```
qsub -l select=1:ncpus=1:mem=14gb -l place=pack:excl
```

Even though the job may get assigned V1, V2, V3, and V4, the job has requested only 1 CPU so PBS needs to check out 1 licenses.

Case 4: Job not requesting memory-only vnodes:

```
qsub -l select=1:ncpus=2 -lplace=pack:excl
```

Even though this job may get assigned all CPUs (4) of V1 exclusively, this job has only requested 2 CPUs so PBS will check out 2 licenses.

Sub-Case 4a:

```
qsub -lselect=1:ncpus=2:mem=10gb -lplace=excl
```

Scheduler runs job on:
V2: 0 CPU 1 GB
V3: 0 CPU 4 GB
V4: 3 CPUs 5 GB

The job has exclusive use of the vnodes, but the user only requested 2 CPUs, so 2 licenses are needed to run the job.

Sub-Case 4b:

```
qsub -l select=1:ncpus=2:mem=10gb -lplace=excl
```

Scheduler runs job on:
V1: 0 CPU 1 GB
V3: 0 CPU 4 GB
V4: 3 CPUs 5 GB

It doesn't matter how the scheduler assigned the vnodes, the CPUs requested is always honored for figuring the number of license tokens to get. This also requires 2 licenses.

11. Complex request:

```
qsub -l select=2:ncpus=1+5:ncpus=2
```

This requires 12 licenses.

12. Licensing virtual CPUs:

Given hostA which has 2 physical CPUs, with resources_available.ncpus set to 4 on the vnode representing the host:

```
qsub -l select=1:ncpus=3:host=hostA
```

PBS will check out 3 licenses for the job.

13. Licensing on a Hyperthreaded Machine:

A hyperthreaded hostB has 2 physical CPUs, where the OS reports 4 logical CPUs. The vnode representing the host has the default value for resources_available.ncpus of 2 and pcpus=2. Now set resources_available.ncpus=4:

```
qsub -l select=ncpus=4:host=hostB
```

PBS will need 4 licenses to run the job.

14. Licensing multi-core systems:

On multi-core systems, the number of CPUs requiring licenses is the number requested by the job.

On a host with two dual core chips (4 CPUs), a job asking for 2 CPUs would require 2 licenses.

5.9.1.1 Licensing and Job States

When a job finishes execution (i.e. job has exited), the licenses assigned to the job is returned to the PBS license pool. The pool is then subject to the constraints given in `pbs_license_min` and `pbs_license_linger_time`.

A job that is held is treated as if the job has finished execution. Licenses assigned to it are released to the PBS local license pool.

The licenses assigned to a suspended job also return to the PBS license pool in order to be available to other PBS jobs. The returned licenses are also subject to the `pbs_license_min` and `pbs_license_linger_time` constraints.

The scheduler makes sure licenses are available before resuming any job. If the licenses are not available, a scheduler log message and job comment are provided to warn of the situation.

Example:
JobA has been submitted as follows:

```
qsub -l select=4:ncpus=1:mem=2gb <job.script>
```

The PBS server checked out 4 licenses and executed JobA.

At some point, the scheduler decided to suspend JobA to execute higher priority jobs. The 4 licenses checked out initially are checked back into the PBS license pool (not the FLEXlm server.) They can be reused by other jobs, and also be made available when JobA gets resumed.

JobA successfully resumes after high priority jobs complete, and PBS was able to get 4 licenses from its pool of licenses.

5.9.1.2 PBS Jobs and HyperWorks

PBS jobs and Hyperworks applications check out licenses from the same server, but the licenses have distinguishing "features", as in "PBSprofessonal" feature GWUs (GridWorks units), and "Hyperworks" feature HWUs (HyperWorks Units).

5.9.1.3 Licensing and Advance Reservations

For advance reservations to work, set the `pbs_license_min` to the total number of CPUs, including virtual CPUs, in the PBS complex.

When the scheduler confirms a reservation, the server makes sure the resources requested by the reservation (e.g. vnodes/CPUs) be available for the reservation. If the PBS server always has licenses for the total number of CPUs in the complex always checked out, this guarantees that there will be licenses available to satisfy these reservations.

Upon confirming a reservation, the PBS server will give a warning if `pbs_license_min` is less than the total number of CPUs in the complex.

If the `pbs_license_min` attribute has not been set in the manner recommended, then when the reservation starts, there's a chance that some of the reservation jobs won't run due to shortage of licenses available. In this case, a job comment and scheduler log entry will be provided.

Example:
The user creates the following reservation requesting 10 CPUs:

```
pbs_rsub -R 1500 -E 1600 -lselect=ncpus=10
<resv-id> confirmed
```

The user submits jobs to the reservation queue:

```
qsub -q <resv_queue> -l select=ncpus=5 compute_job
qsub -q  <resv_queue> -l select=ncpus=5 compute_job2
```

At the start of the reservation, the two jobs run, each getting 5 licenses.

5.9.1.4 Suspended Jobs

If you set the server's pbs_license_min attribute to the number of CPUs in the complex, suspended jobs will be able to resume whey they are ready.

5.10 Upgrading

5.10.1 Overlay Upgrades

On an overlay upgrade, if the old MOM was killed and restarted with the "pbs_mom -p" option to allow existing jobs to run, and a post 8.0 (9.0 and later) server comes up, the new server will also allow already running jobs to continue. PBS will try to re-license the job, but won't kill the job if there aren't enough licenses.

Except on Solaris, this involves running the new "INSTALL" program over the currently installed (old version) of PBS.

5.10.1.1 Upgrading the Server

When you run the "INSTALL" program to install the new server, it will issue a warning saying that "this new installation would require a FLEX type of license, and if you don't have one, here's how to obtain it". It will also prompt you to "continue installation", or "abort installation". Aborting the installation leaves any previously installed PBS server, scheduler, MOM, or clients intact.

5.10.1.2 Upgrading the MOM

When you run the "INSTALL" program to install the new MOM, a warning message will also be issued saying "PBS MOM in this package expects a server that supports FLEX type of license. Continue installation or abort installation?" If the installation aborted, the currently installed MOM remains intact.

5.10.2 Migration Upgrade

There are no new or different upgrading steps for migration upgrades.

5.11 Stopping the License Server

5.11.1 UNIX

Step 1 Go to the license server machine.

Step 2 At the UNIX prompt, enter the following:
ps -ef | grep lmgrd
The "|" is the pipe sign. To type it, press SHIFT-[\]. The lmgrd process is returned.

Step 3 Kill all Altair HyperWorks lmgrd processes by typing the following at the UNIX command prompt:
kill <lmgrd process ids>

Step 4 At the UNIX prompt, enter the following:
ps -ef | grep altair_lm or for Linux ps ax | grep | lmgrd.
The altair_lm process is returned.

Step 5 Kill all Altair HyperWorks altair_lm processes by entering the following at the UNIX command
prompt:
kill <altair_lm process ids>

5.11.2 Windows

Step 1 Go to the license server machine.

Step 2 Execute <install_location>/security/lmtools.exe.

Step 3 Click the Service/License File tab.

Step 4 Click the Configuration using Services option to activate it.

Step 5 Click the Start/Stop/Reread tab.

Step 6 Click Stop Server.

Step 7 Click File.

Step 8 Click Exit to close LMTOOLS.

5.12 The lmgrd Daemon

Upon startup, the lmgrd daemon reads the license file. On UNIX systems, it is strongly recommended that lmgrd be run as a non-privileged user (not root).
Usage:

```
lmgrd [-c license_file_list] [-l [+]debug_log_path]
    [-2 -p] [-local] [-nfs_log] [-x lmdown]
    [-x lmremove] [-z ] [-v]
```

–c license_file_list	Use the specified license file(s).
–l [+]debug_log_path	Write debugging information to file debug_log_path. This option uses the letter l, not the numeral 1. Prepending debug_log_path with the + character appends logging entries.
–2 -p	Restricts usage of lmdown, lmreread, and lmremove to a FLEXlm administrator who is by default root. If there a UNIX group called “lmadmin,” then use is restricted to only members of that group. If root is not a member of this group, then root does not have permission to use any of the above utilities. If –2-p is used when starting lmgrd, no user on Windows can shut down the license server with lmdown.
-local	Restricts the lmdown command to be run only from the same machine where lmgrd is running.
–nfs_log	It is not recommended to write a debug log to an NFS-mounted or Windows network -mounted disk since this can significantly slow the license server. If you choose to write to a mounted disk and the speed of the license server is too slow, you can use this flag to cache debug info before it is written out (when approxi-

mately 1kb of data is reached) thereby improving license server performance.

–x lmdown — Disable the lmdown command (no user can run lmdown). If lmdown is disabled, you will need to stop lmgrd via kill pid (UNIX) or stop the lmgrd and vendor daemon processes through the Windows Task Manager or Windows service. On UNIX, be sure the kill command does not have a –9 argument.

–x lmremove — Disable the lmremove command (no user can run lmremove).

-z — Run in foreground. The default behavior is to run in the background. If -l debug_log_path is present, then no windows are used, but if no -l argument specified, separate windows are used for lmgrd and each vendor daemon.

-v — Prints lmgrd version number and copyright and exits.

5.13 Tools

5.13.1 The lmutil Utility

The lmutil command is the main FLEXnet Publisher license management utility. For information on how to use lmutil, see the HyperWorks 8.0 Installation Guide, both "License Administration Tools" on p. 66 and Appendix F.

5.13.2 The lmhostid Utility

The lmhostid utility reports the lmhostid of this machine, if it is a supported platform. The default lmhostid type is displayed for a platform, unless an optional lmhostid type is specified and supported by that platform.
Usage:

lmhostid [-n] [type]

type — type is one of:
[-internet] (Optional on all platforms)
[-vsn]
[-flexid]

-n — No header is printed, only the lmhostid is printed.

-internet	IP address in ###.###.###.### format.
-vsn	Volume Serial Number of the Windows C:\ drive.
-flexid	Macrovision dongle-based lmhostid. (Windows)

The output of this command looks like this:
lmhostid - Copyright (c) 1989, 2002 Macrovision Software, Inc.
The FLEXlm lmhostid of this machine is "69021c89"

5.13.3 FLEXnet Publisher Options File

The options file allows the license administrator to control various operating parameters of FLEXnet. Users can be identified by their user name, host name, display, or IP address. Specifically, the license administrator can:

Allow the use of features
Deny the use of features
Reserve licenses
Control the amount of information logged about license usage
Enable a report log file

Options files allow you, as the license administrator, to be as secure or open with licenses as you like. For detailed information on the options file, see “FLEXnet Publisher Options File” on page 67 of the HyperWorks 8.0 Installation Guide.

The default name of the file is altair_lm.opt, and it is recommended that the options file be placed in the same directory as the license. You can set the path to the file in the license file using the options_file_path setting.

5.14 Logging for Licensing

The license server provides two log files: the debug log and the report log. The report log contains usage information written by altair_lm, and the debug log(s) contain status and error messages from lmgrd and altair_lm. By default, these log files are written to the directory in which lmgrd is started, so the directory must be writeable by that user if you use the default. You can specify where the log files are written. The lmgrd daemon also writes an encrypted usage file for use by MacroVision products. The location of this file is specified in the options file. See Macrovison’s FLEXnet Licensing End User Guide for details.

Macrovision recommends that lmgrd, altair_lm, the license file, the debug log file and the report log files should be on locally mounted disks. If any of these files are on a remote mounted disk, there are twice as many points of failure where licenses could be lost.

You can turn off all logging by lmgrd by setting the ALTAIR_NOLOG environment variable before starting the license server.

When altair_lm is started, by default it writes its debug information to standard out. It must have write permission for the directory where the log file is written. The log file contains checkout information. The lmgrd daemon will by default write an encrypted log file.

5.14.1 Report Logging

By default, altair_lm's report logging is disabled. To enable report logging, either add REPORTLOG to the options file, or use "lmutil lmswitchr". These are described below.

5.14.1.1 Options to lmutil for Report Logging

lmutil lmnewlog	Moves the existing report log information to a new file, then starts a new report log with the original report log file name. See p. 66 in HyperWorks 8.0 Installation Guide.
lmutil lmswitchr	Closes the existing report log and starts a new report log with a new file name. It also starts a new report log file if one does not already exist. If using the REPORTLOG line in options file, you must change the filename in that line. See p. 67 in HyperWorks 8.0 Installation Guide
lmreread	altair_lm is signaled to reread the license file and options file for changes in feature licensing information or option settings.

5.14.1.2 Options File for Report Logging

REPORTLOG	Specify the report log file for altair_lm. It is recommended that you precede the report_log_path with a + character to append logging entries, otherwise the file is overwritten each time altair_lm is started. On Windows, path names which include spaces have to be enclosed in double quotes. See p. 129 in HyperWorks 8.0 Installation Guide.

5.14.2 Debug Logging

The lmgrd daemon will write its debug log output to standard out by default To write lmgrd and altair_lm debug log output to the same file, either redirect the output of the license server system to a file or start lmgrd with the -l debug_log_path option. See section 5.12 “The lmgrd Daemon” on page 106.

You can specify where the altair_lm debug output should be written using the DEBUGLOG line in the options file, or lmutil lmswitch. Use the NOLOG line in the options file to control what is logged for altair_lm. These are described below.

By default, the altair_lm daemon writes its debug information to standard out. You can specify the log file name and location by setting the environment variable ‘ALTAIR_LOGFILE’ before starting the server.

On Windows, the path to the lmgrd debug log file is:

<install_location>\security\WIN32\lmgrd_debug.log

It is not recommended to write a debug log to an NFS-mounted or Windows network-mounted disk since this can significantly slow the license server. If you choose to write to a mounted disk and the speed of the license server is too slow, you can use:

lmgrd -nfs_log

This flag will cause the server to cache debug info before it is written out (when approximately 1kb of data is reached) thereby improving license server performance.

5.14.2.1 Options to lmutil for Debug Logging

lmswitch	Closes the existing debug log file written by altair_lm and starts a new debug log with a new file name. It also starts a new debug log file written by altair_lm if one does not already exist. See page 67 of the HyperWorks 8.0 Installation Guide.

5.14.2.2 Options File for Debug Logging

DEBUGLOG [+]debug_log_path	Specify a location for the debug log output from altair_lm. Takes effect when altair_lm is started or its options file is read. See page 68 of the HyperWorks 8.0 Installation Guide.
NOLOG	Suppress logging selected type of event in the debug log file. For example, the line: NOLOG QUEUED would cause QUEUED messages to be omitted from the debug

log file. See page 129 of the HyperWorks 8.0 Installation Guide.

5.15 Licensing Errors

5.15.1 Licensing and Loss of Communication to License Server

If PBS loses contact with the Altair License Server, any jobs currently running will not be interrupted or killed. The PBS server will continually attempt to reconnect to the license server, and re-license the assigned vnodes once the contact to the license server is restored.

No new jobs will run if PBS server loses contact with the License server.

If PBS cannot detect a license server host and port when it starts up, the server logs an error message:

"Did not find a license server host and port (pbs_license_file_location=<X>). No external license server will be contacted"

If the PBS scheduler cannot obtain the licenses to run or resume a job, the scheduler will log a message:

"Could not run job <job>; unable to obtain <N> CPU licenses. avail licenses=<Y>"
"Could not resume <job>; unable to obtain <N> CPU licenses. avail licenses=<Y>"

If PBS cannot contact the license server, the server will log a message:

"Unable to connect to license server at pbs_license_file_location=<X>"

If the value of the `pbs_license_min` attribute is less than the number of CPUs in the PBS complex when a reservation is being confirmed, the server will log a warning:

"WARNING: reservation <resid> confirmed, but if reservation starts now, its jobs are not guaranteed to run as pbs_license_min=<X> < <Y> (# of CPUs in the complex)"

If the PBS server cannot get the number of licenses specified in `pbs_license_min` from the FLEX server, the server will log a message:

"checked-out only <X> CPU licenses instead of pbs_license_min=<Y> from license server at host <H>, port <P>. Will try to get more later."

If the PBS server encounters a proprietary license key that is of not type “-T”, then the server will log the following message:

“license key #1 is invalid: invalid type or version".

5.15.2 Duplicate License Daemon

Here's what happens when the Altair license daemons starts successfully, but another license daemon is already running:

```
$ ../scripts/lmgrd -c <installation location>/altair/security/altair_lic.dat
$ 17:34:17 (lmgrd) -----------------------------------------------
17:34:17 (lmgrd)   Please Note:
17:34:17 (lmgrd)
17:34:17 (lmgrd)   This log is intended for debug purposes only.
17:34:17 (lmgrd)   In order to capture accurate license
17:34:17 (lmgrd)   usage data into an organized repository,
17:34:17 (lmgrd)   please enable report logging. Use Macrovision's
17:34:17 (lmgrd)   software license administration  solution,
17:34:17 (lmgrd)   FLEXnet Manager, to  readily gain visibility
17:34:17 (lmgrd)   into license usage data and to create
17:34:17 (lmgrd)   insightful reports on critical information like
17:34:17 (lmgrd)   license availability and usage. FLEXnet Manager
17:34:17 (lmgrd)   can be fully automated to run these reports on
17:34:17 (lmgrd)   schedule and can be used to track license
17:34:17 (lmgrd)   servers and usage across a heterogeneous
17:34:17 (lmgrd)   network of servers including Windows NT, Linux
17:34:17 (lmgrd)   and UNIX. Contact Macrovision at
17:34:17 (lmgrd)   www.macrovision.com for more details on how to
17:34:17 (lmgrd)   obtain an evaluation copy of FLEXnet Manager
17:34:17 (lmgrd)   for your enterprise.
17:34:17 (lmgrd)
17:34:17 (lmgrd) -----------------------------------------------
17:34:17 (lmgrd)
17:34:17 (lmgrd)
17:34:17 (lmgrd) The TCP port number in the license, 7788, is already in use.
17:34:17 (lmgrd) Possible causes:
```

17:34:17 (lmgrd) 1) The license server manager (lmgrd) is already running for this license.
17:34:17 (lmgrd) 2) The OS has not "cleared" this port since lmgrd died.
17:34:17 (lmgrd) 3) Another process is using this port number (unlikely).
17:34:17 (lmgrd) Solutions:
17:34:17 (lmgrd) 1) Make sure lmgrd and all vendor daemons for this
17:34:17 (lmgrd) license are not running.
17:34:17 (lmgrd) 2) You may have to wait for the OS to clear this port.
17:34:17 (lmgrd) Retrying for about 5 more minutes
17:34:35 (lmgrd) Still trying...
17:34:53 (lmgrd) Still trying...
17:35:11 (lmgrd) Still trying...

5.15.3 The altair_lm Daemon Dies Soon After License Server is Started on UNIX

This may be due to a port number already in use or an existing lockaltair_lm file. Sometimes when the lmgrd goes down or is killed the port status remains "active" or in use because the operating system does not release the port in a timely manner. If the port number is already in use, then either wait for the port to free or go to the installation instructions, license file section for instructions to change the port number in the SERVER line of the license file to eliminate conflicts. This lockaltair_lm lock file exists when the vendor daemon is running. The file is removed when the vendor daemon is shut down. Sometimes, if the daemon is shut down by a fatal error, the lockaltair_lm file still exists. This file cannot exist before the daemon is restarted. Remove the lockaltair_lm file and restart FLEXnet as follows:

Step 1 Follow the steps to stop the license server in section 5.11 "Stopping the License Server" on page 105.

Step 2 Look for a lockaltair_lm file in the following directories:
/tmp/
/etc/
/var/
/usr/
/tmp/usr

Step 3 Delete the lockaltair_lm file.

Step 4 Restart FLEXnet. See the instructions in section 4.3.2.8 "Start-

ing and Stopping the Altair License Daemon” on page 43.

5.15.4 The lmgrd Script Does Not Start the FLEXnet License Server

The license server and the license seem to be installed correctly. First, kill any existing lmgrd or altair_lm processes. See section 5.11 “Stopping the License Server” on page 105. Then, try starting the FLEXnet license server manually. Make sure that a lockaltair_lm file does not exist. See section 5.15.3 “The altair_lm Daemon Dies Soon After License Server is Started on UNIX” on page 113.

5.15.5 User Error Messages

If a user's job could not be run due to unavailable licenses, the job will get a comment: “Could not run job <job>; unable to obtain <N> CPU licenses. avail_licenses=<Y>”

Chapter 6
Upgrading PBS Professional

This chapter shows how to upgrade from a previous version of PBS Professional. If PBS Professional is not installed on your system, you can skip this chapter.

6.1 Types of Upgrades

There are two types of upgrades available for PBS Professional:

overlay upgrade	Installs the new binaries on top of the old ones. Jobs stay in place, and can continue to run, except on the Altix.
migration upgrade	Installs the new version in a separate location. This can be the standard location if the old version has been moved. Jobs are moved from the old server to the new one, and cannot be running during the move. Must be used for Windows.

Usually, UNIX systems can have overlay upgrades. Migration upgrades are necessary when moving between 32-bit and 64-bit versions of PBS, and when upgrading Windows.

When upgrading on an Altix that will be using cpusets, follow the instructions in section 6.5.6 “Upgrading on an Altix or a Complex Containing One or More Altixes” on page 127. When upgrading on an Altix that will not be using cpusets, follow the instructions in section 6.5.7 “Migration Upgrade Under UNIX” on page 136.

When upgrading an IRIX machine, follow the instructions in section 6.5.7 “Migration Upgrade Under UNIX” on page 136.

For specific upgrade recommendations and updates, see the Release Notes.

6.2 Differences from Previous Versions

PBS is now licensed using a FLEX license server. This license server must be installed and configured before installing PBS. See section 4.2 “FLEX Licensing” on page 36 and section “Licensing” on page 81.

The server will convert the old style properties, used in PBS Professional 7.0 and before, in the nodes file to boolean resources. However, the server updates the nodes file only when vnodes are created, deleted or modified via qmgr. You will not see an updated nodes file until after the server is restarted.

The default `PBS_HOME` directory for AIX was changed from `/usr/spool/PBS` to `/usr/local/spool/PBS`.

6.2.1 Caution

Starting with version 9.0, the PBS server and all MOMs must be upgraded at the same time.

Do not unset the value for the `default_chunk.ncpus` server attribute. It is set by the server to 1. You can set it to another non-zero value, but a value of 0 will produce undefined behavior. When the PBS Server initializes and the Server attribute "default_chunk" has not been specified during a prior run, the Server will internally set the following:

```
default_chunk.ncpus=1
```

This ensures that each "chunk" of a job's select specification requests at least one CPU. If the Administrator explicitly sets the Server attribute "default_chunk", that setting will be retained across server restarts.

It is strongly advised not to set "default_chunk.ncpus=1" to zero. The attribute may be set to a higher value if appropriate.

6.3 FLEX Licensing

The server's pbs_license_file_location attribute may be unset or set to some previous value. Set it to the correct value. See section 5.4.3 "Setting Server Licensing Attributes" on page 88. You must start the license server before starting PBS, so follow the steps for installing and starting the license server before upgrading.

6.3.1 Overlay Upgrades

On an overlay upgrade, the old MOM is killed, and the new MOM is started with the "pbs_mom -p" option to allow existing jobs to run. When the 9.0 or later server comes up, the new server will also allow already running jobs to continue. PBS will try to re-license the jobs, but won't kill the jobs if there are not enough licenses.

Except on Solaris, this involves running the new "INSTALL" program over the currently installed (old version) of PBS.

6.3.1.1 Upgrading the Server

When you run the "INSTALL" program to install the new server, it will issue a warning saying that "this new installation would require a FLEX type of license, and if you don't have one, here's how to obtain it". It will also prompt you to "continue installation", or "abort installation". Aborting the installation leaves any previously installed PBS server, scheduler, MOM, or clients intact.

6.3.1.2 Upgrading the MOM

When you run the "INSTALL" program to install the new MOM, a warning message will also be issued saying "PBS MOM in this package expects a server that supports FLEX type of license. Continue installation or abort installation?" If the installation is aborted, the currently installed MOM remains intact.

6.3.2 Migration Upgrade

There are no new or different upgrading steps for migration upgrades.

6.4 After Upgrading

If you were using vmem at the queue or server level before the upgrade, then after upgrading you must add vmem to the new `resource_unset_infinite` sched_config option. See section 9.3 “Scheduler Configuration Parameters” on page 315. Otherwise jobs requesting vmem will not run.

6.5 Upgrading Under UNIX and Linux

When you get your new version of PBS, unpack it (unzip, untar) as a non-privileged user. When you follow the upgrading instructions below, all of the steps should be performed as root.

6.5.1 Directories

The locations of `PBS_HOME` and `PBS_EXEC` are specified in the file `/etc/pbs.conf`. In the following instructions, replace `PBS_HOME` or `PBS_EXEC` with the appropriate values.

For example, if `pbs.conf` specifies `PBS_HOME` as `/var/spool/PBS`, and an instruction says

“`mv PBS_HOME PBS_HOME.old`”,

then type

“`mv /var/spool/PBS /var/spool/PBS.old`”.

6.5.2 Managing Integrations

If you used the `pbsrun_wrap` mechanism with your old version of PBS, you must first unwrap any MPIs that you wrapped. This would include MPICH-GM, MPICH-MX, Intel MPI, MPICH2, etc.

If you are upgrading from a version prior to 9.0 and you had linked the `poe` command to `pbs_poe`, you must remove the link. The new version of PBS uses the `pbsrun_wrap` method; see section 11.10.15 “Wrapping IBM's poe” on page 450.

You can wrap your MPIs after upgrading PBS. See section 11.10 “Support for MPI” on page 433.

6.5.3 Upgrading on Multiple Machines

Instead of running the installer by hand on each machine, you can use a command such as pdsh with NFS-mounted common directories on all hosts to distribute the installation to each host. The general form to distribute commands to a large number of hosts is

```
# for hosts node001-005
pdsh -w node[001-005] command
```

When using the PBS INSTALL command you can make a simple text file called "answers" with the answers to the installation prompts, and redirect them into the INSTALL script:

```
./INSTALL < answers
```

where `answers` contains the answers for a MOM-only installation:

```
2
y
<server name>
y
```

The complete command line would be:

```
pdsh -w node[001-005] ./INSTALL < answers
```

6.5.4 Overlay Upgrade Under UNIX and Linux

Except when using Solaris and the Altix, use the following steps to perform an overlay upgrade. For Solaris, see section 6.5.5 "Overlay Upgrade under Solaris" on page 123. for the Altix, see section 6.5.6 "Upgrading on an Altix or a Complex Containing One or More Altixes" on page 127. You will probably want to keep any running jobs running.

The following commands must be run as "root".

6.5.4.1 Back Up Your Existing PBS

Make a tarfile of the `PBS_HOME` and `PBS_EXEC` directories.

1 Make a backup directory:

```
mkdir /tmp/pbs_backup
```

2 Make a tarfile of PBS_HOME:

```
cd PBS_HOME/..

tar -cvf \
    /tmp/pbs_backup/PBS_HOME_backup.tar \
    PBS_HOME
```

3 Make a tarfile of PBS_EXEC:

```
cd PBS_EXEC/..

tar -cvf /tmp/pbs_backup/PBS_EXEC_backup.tar
PBS_EXEC
```

4 Make a copy of your configuration file:

```
cp /etc/pbs.conf \
    /tmp/pbs_backup/pbs.conf.backup
```

5 If they exist (PBS 8.0 and later), make a copy of the site-defined configuration files:

```
mkdir /tmp/pbs_backup/mom_configs

$PBS_EXEC/sbin/pbs_mom -s list \
    | egrep -v '^PBS' | while read file
do
    $PBS_EXEC/sbin/pbs_mom -s show $file \
       > /tmp/pbs_backup/mom_configs/$file
done
PBS_EXEC
```

6 Make a copy of the scheduler's directory to modify:

```
cp PBS_HOME/sched_priv \
    /tmp/pbs_backup/sched_priv.work
```

6.5.4.2 Shut Down Your Existing PBS

Shut down PBS, keeping running jobs running. The `qterm` command will use the `-t quick` option unless you specify otherwise.

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

If you wish to requeue and/or kill running jobs during shutdown, see "Stopping PBS" on page 417.

6.5.4.3 Install the New Version of PBS

Install the new version of PBS on all hosts without uninstalling the previous version. The installation program will read your existing installation parameters from `/etc/pbs.conf`, and prompt you to confirm that you wish to use them.

On each host, go to the directory where you put the PBS installation script. Type:

```
./INSTALL
```

When the the install script prompts you whether or not to start PBS, if you are on a system running a MOM, answer "n" for "no". Do **not** start pbs_mom now. Instead use pbs_mom -p, which will preserve running jobs. This is shown in section 6.5.4.7 "Start the New PBS" on page 123.

6.5.4.4 Prepare the New Scheduler's Configuration File

1 Make a copy of the new `sched_config`, which is in `PBS_EXEC/etc/pbs_sched_config`.

```
cp PBS_EXEC/etc/pbs_sched_config \
    PBS_EXEC/etc/pbs_sched_config.new
```

2 Update `PBS_EXEC/etc/pbs_sched_config.new` with any modifications that were made to the current `PBS_HOME/sched_priv/sched_config`.

3 If it exists, replace the `strict_fifo` option with `strict_ordering`. If you do not, a warning will be printed in the log when the scheduler starts.

4 If you copied over your scheduler log filter setting, make sure the new configuration file has 1024 added to it. If the value is less than 1024, add 1024 to it:

Edit `PBS_EXEC/etc/pbs_sched_config.new.`

If your previous log filter line was:

```
log_filter:256
```

change it to:

```
log_filter:1280
```

If you do not, you will be inundated with logging messages.

5 Move `PBS_EXEC/etc/pbs_sched_config.new` to the correct name and location, i.e. `PBS_HOME/sched_priv/sched_config`.

```
mv PBS_EXEC/etc/pbs_sched_config.new \
    PBS_HOME/sched_priv/sched_config
```

6.5.4.5 Modify the New Server's Resource File

Add the "h" flag to those vnode-level resources listed in the server's `PBS_HOME/server_priv/resourcedef` file that have the "n" or "`f`" flag. For example, if you had:

```
switch type=string flag=n
```

This would become:

```
switch type=string flag=nh
```

See "Resource Flags" on page 225.

6.5.4.6 Update Fairshare Entities and Database

This step is only necessary if you were not using egroup:euser as your fairshare entity, but wish to do so now. Follow the procedures in section 6.5.10 "Updating Fairshare Definitions for egroup:euser" on page 149.

6.5.4.7 Start the New PBS

1 Start PBS on the execution hosts. On each machine, type:

```
PBS_EXEC/sbin/pbs_mom -p
```

2 Start the scheduler and server on the server's host. No options are required:

```
PBS_EXEC/sbin/pbs_sched
```

```
PBS_EXEC/sbin/pbs_server
```

6.5.4.8 Set the Server's pbs_license_file_location Attribute

Set the server's pbs_license_file_location attribute to point to the license file or license server. See section 5.4.3.1 "Setting the License File Location in pbs_license_file_location" on page 88.

6.5.5 Overlay Upgrade under Solaris

You will probably want to leave running jobs in the running state.

6.5.5.1 Back Up Your Existing PBS

Make a tarfile of the PBS_HOME and PBS_EXEC directories:

1 Make a backup directory:

```
mkdir /tmp/pbs_backup
```

2 Make a tarfile of PBS_HOME:

```
cd PBS_HOME/..
```

```
tar -cvf /tmp/pbs_backup/ \
    PBS_HOME_backup.tar PBS_HOME
```

3 Make a tarfile of PBS_EXEC:

```
cd PBS_EXEC/..

tar -cvf \
    /tmp/pbs_backup/PBS_EXEC_backup.tar \
     PBS_EXEC
```

4 Make a copy of your configuration file:

```
cp /etc/pbs.conf \
    /tmp/pbs_backup/pbs.conf.backup
```

5 If they exist (PBS 8.0 and later), make a copy of the site-defined configuration files:

```
mkdir /tmp/pbs_backup/mom_configs

$PBS_EXEC/sbin/pbs_mom -s list \
    | egrep -v '^PBS' | while read file
do
    $PBS_EXEC/sbin/pbs_mom -s show $file \
       > /tmp/pbs_backup/mom_configs/$file
done
```

6 Make a copy of the scheduler's directory to modify:

```
cp PBS_HOME/sched_priv \
    /tmp/pbs_backup/sched_priv.work
```

6.5.5.2 Shut Down PBS

Shut down PBS. The `qterm` command will use the default `-t quick` option, which leaves running jobs in the running state.

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

If you wish to requeue or kill running jobs during shutdown, see "Stopping PBS" on page 417.

6.5.5.3 Remove the Old PBS Package

You must remove the PBS package, which is named either "pbs32" or "pbs64".

1 Find the name for the old package:

```
pkginfo | grep -i pbs
```

2 Remove the old PBS package:

```
pkgrm  pbs64
```

-or-

```
pkgrm  pbs32
```

6.5.5.4 Install the New Version of PBS

Install the new PBS Professional version. From the directory containing the installation script:

```
./INSTALL
```

The installation program will pick up your existing installation parameters from `/etc/pbs.conf`, and prompt you to confirm that you wish to use them.

6.5.5.5 Prepare the New Scheduler's Configuration File

1 Make a copy of the new `sched_config`, which is in `PBS_EXEC/etc/pbs_sched_config`.

```
cp PBS_EXEC/etc/pbs_sched_config \
    PBS_EXEC/etc/pbs_sched_config.new
```

2 Update `PBS_EXEC/etc/pbs_sched_config.new`

with any modifications that were made to the current PBS_HOME/sched_priv/sched_config.

3 If it exists, replace the strict_fifo option with strict_ordering. If you do not, a warning will be printed in the log when the scheduler starts.

4 If you copied over the scheduler's log filter, and have not added 1024 to it, add 1024 to the scheduler's log filter. If the log filter is less than 1024, add 1024 to it. Edit PBS_EXEC/etc/pbs_sched_config**.new.** If your previous log filter line was:

```
log_filter:256
```

change it to:

```
log_filter:1280
```

If you do not, you will be inundated with logging messages.

5 Move PBS_EXEC/etc/pbs_sched_config.new to the correct name and location, i.e. PBS_HOME/sched_priv/sched_config.

```
mv PBS_EXEC/etc/pbs_sched_config.new \
    PBS_HOME/sched_priv/sched_config
```

6.5.5.6 Update Fairshare Entities and Database

This step is only necessary if you were not using egroup:euser as your fairshare entity, but wish to do so now. Follow the procedures in section 6.5.10 “Updating Fairshare Definitions for egroup:euser” on page 149.

6.5.5.7 Modify the Server’s Resource File

Add the “h” flag to those vnode-level resources listed in the server’s PBS_HOME/server_priv/resourcedef file that have the “n” flag.

6.5.5.8 Start the New PBS

1 Start PBS on the execution hosts:

PBS_EXEC/sbin/pbs_mom -p

2 Start the new scheduler and server on the server's host:

PBS_EXEC/sbin/pbs_sched

PBS_EXEC/sbin/pbs_server

6.5.5.9 Set the Server's pbs_license_file_location Attribute

Set the server's pbs_license_file_location attribute to point to the license file or license server. See section 5.4.3.1 "Setting the License File Location in pbs_license_file_location" on page 88.

6.5.6 Upgrading on an Altix or a Complex Containing One or More Altixes

This section contains instructions for an overlay upgrade of an Altix that will use cpusets. If you want to configure PBS on the Altix to support cpusets, run `pbs_mom.cpuset`. Jobs cannot be running on an Altix during the upgrade. Jobs on the Altix can be requeued, killed, or allowed to finish running.

If you want to use cpusets after the upgrade, you must have a vnode definitions file. The vnode definitions file is generated automatically for an Altix running ProPack 4 or greater. However, for an Altix running ProPack 2 or 3, this file must be generated by the administrator or by the PBS Professional support team. See section "Technical Support" on page ii.

6.5.6.1 Back Up Your Existing PBS Professional

Make a tarfile of the `PBS_HOME` and `PBS_EXEC` directories.

1 Make a backup directory:

mkdir /tmp/pbs_backup

2 Make a tarfile of `PBS_HOME`:

```
cd PBS_HOME/..

tar -cvf \
    /tmp/pbs_backup/PBS_HOME_backup.tar PBS_HOME
```

3 Make a tarfile of PBS_EXEC:

```
cd PBS_EXEC/..

tar -cvf \
    /tmp/pbs_backup/PBS_EXEC_backup.tar PBS_EXEC
```

4 Make a copy of your configuration file:

```
cp /etc/pbs.conf \
    /tmp/pbs_backup/pbs.conf.backup
```

5 If they exist (PBS 8.0 and later), make a copy of the site-defined configuration files on each machine:

```
mkdir /tmp/pbs_backup/mom_configs

$PBS_EXEC/sbin/pbs_mom -s list \
    | egrep -v '^PBS' | while read file
do
    $PBS_EXEC/sbin/pbs_mom -s show $file \
       > /tmp/pbs_backup/mom_configs/$file
done
```

6 Save a list of the hosts:

```
pbsnodes -a > /tmp/pbs_backup/hostlist
```

7 Save the server's nodes file:

```
Qmgr: print nodes @default > \
    /tmp/pbs_backup/newnodes
```

8 If you are upgrading from a pre-8.0 version, ensure that each Altix host has its values for resources_available.(mem|vmem|ncpus) unset:

```
Qmgr: unset node <hostname> \
    resources_available.mem
Qmgr: unset node <hostname> \
    resources_available.ncpus
Qmgr: unset node <hostname> \
    resources_available.vmem
```

9 Make a copy of the scheduler's directory to modify:

```
cp PBS_HOME/sched_priv \
    /tmp/pbs_backup/sched_priv.work
```

6.5.6.2 Stop New Jobs From Starting

You must stop any jobs from starting. Do this by stopping scheduling.

Stop scheduling:

```
Qmgr: set server scheduling=false
```

6.5.6.3 Stop Jobs Running on the Altix

Jobs cannot be running on an Altix during an upgrade from a version before 8.0 to 8.0 or later. You can a) requeue any jobs running on the Altix, b) drain the host by letting existing jobs on the Altix finish running, c) kill the jobs on the Altix, or d) requeue all jobs in the complex. If you choose d), you can skip this step. The next step has instructions for requeueing all jobs in the complex. If you choose to requeue jobs, those jobs that are marked non-rerunnable will be killed.

To requeue any jobs running on the Altix:

1 List the jobs on the Altix This will list some jobs more than once. You only need to requeue each job once:

```
pbsnodes <hostname> | grep Jobs
```

2 Requeue the jobs:

```
qrerun <job ID> <job ID> ...
```

To kill the jobs on the Altix:

1. List the jobs on the Altix This will list some jobs more than once. You only need to kill each job once:

   ```
   pbsnodes <hostname> | grep Jobs
   ```

2. Use the `qdel` command to kill each job by job ID:

   ```
   qdel <job ID> <job ID> ...
   ```

To drain the host, wait until any jobs running on the Altix have finished.

6.5.6.4 Shut Down Your Existing PBS

You can let any non-Altix jobs continue to run, or you can requeue or kill all jobs.

To let running jobs continue to run on non-Altix hosts:

Shut down PBS. The `qterm` command will use the `-t quick` option unless you specify otherwise.

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

If your server is not running in a failover environment, the "`-f`" option is not required.

To requeue or kill all jobs:

Shut down PBS. The `qterm -t immediate` command will requeue any jobs that can be requeued and kill those that cannot:

```
qterm -t immediate -m -s -f
```

If your server is not running in a failover environment, the "`-f`" option is not required.

6.5.6.5 Install the New Version of PBS

Install the new version of PBS on all hosts without uninstalling the previous version. The installation program will read your existing installation parameters from `/etc/pbs.conf`, and prompt you to confirm that you wish to use them.

1. On each host, go to the directory where you put the PBS installation script. Type:

   ```
   ./INSTALL
   ```

2. You must **copy**, not move, `pbs_mom`.cpuset to `pbs_mom`. If `pbs.conf` is not in `/etc`, look at the `PBS_CONF_FILE` environment variable for its location. Look in `pbs.conf` for the location of `$PBS_EXEC`.

   ```
   cd $PBS_EXEC/sbin
   rm pbs_mom
   cp pbs_mom.cpuset pbs_mom
   ```

6.5.6.6 Prepare the New Scheduler's Configuration File

1. Make a copy of the new `sched_config`, which is in `PBS_EXEC/etc/pbs_sched_config`.

   ```
   cp PBS_EXEC/etc/pbs_sched_config \
       PBS_EXEC/etc/pbs_sched_config.new
   ```

2. Update `PBS_EXEC/etc/pbs_sched_config`**`.new`** with any modifications that were made to the current `PBS_HOME/sched_priv/sched_config`.

3. If it exists, replace the `strict_fifo` option with `strict_ordering`. If you do not, a warning will be printed in the log when the scheduler starts.

4. If you are upgrading from a version prior to 7.1 and copied over your scheduler log filter setting, make sure the new configuration file has 1024 added to it. If the value is less than 1024, add 1024 to it:

Edit PBS_EXEC/etc/pbs_sched_config**.new.**

If your previous log filter line was:

```
log_filter:256
```

change it to:

```
log_filter:1280
```

If you do not, you will be inundated with logging messages.

5 Move PBS_EXEC/etc/pbs_sched_config.new to the correct name and location, i.e. PBS_HOME/sched_priv/sched_config.

```
mv PBS_EXEC/etc/pbs_sched_config.new \
    PBS_HOME/sched_priv/sched_config
```

6.5.6.7 Update Fairshare Entities and Database

This step is only necessary if you were not using egroup:euser as your fairshare entity, but wish to do so now. Follow the procedures in section 6.5.10 “Updating Fairshare Definitions for egroup:euser” on page 149.

6.5.6.8 Modify the New Server’s Resource File

If you are upgrading from a version prior to 7.1, add the “h” flag to those vnode-level resources listed in the server’s PBS_HOME/server_priv/resourcedef file that have the “n” or “f” flag. For example, if you had:

```
switch type=string flag=n
```

This would become:

```
switch type=string flag=nh
```

See “Resource Flags” on page 225.

6.5.6.9 Read the Nodes Information Back Into qmgr

Read the nodes information back into qmgr:

```
qmgr < /tmp/pbs_backup/newnodes
```

6.5.6.10 Edit the Altix Configuration File

When changing from ProPack 2 or 3 to Propack 4 or 5, remove any `cpuset_create_flags <flags>` initialization other than CPUSET_CPU_EXCLUSIVE from the default MOM config file. See the `pbs_mom(8B)` manual page.

6.5.6.11 Start the New PBS

1 On any non-Altix execution hosts, start PBS. On each of these machines, type:

```
PBS_EXEC/sbin/pbs_mom -p
```

2 On any Altixes, start PBS. For the location of the startup script, see "Starting and Stopping PBS: UNIX and Linux" on page 405:

```
<path to script>pbs start
```

3 If the server's host is not an Altix, start the scheduler and server using the following commands. No options are required:

```
PBS_EXEC/sbin/pbs_sched
```

```
PBS_EXEC/sbin/pbs_server
```

6.5.6.12 Set the Server's pbs_license_file_location Attribute

Set the server's pbs_license_file_location attribute to point to the license file or license server. See section 5.4.3.1 "Setting the License File Location in pbs_license_file_location" on page 88.

6.5.6.13 Generate Vnode Definitions File for ProPack 2, 3

If the Altix is running ProPack 2 or 3, generate a vnode definitions file for it. Support can help you create a preliminary file. See “Technical Support” on page ii.

1 Create the preliminary file `prelim_defs` with the help of the technical support group.

2 Add the definition of the natural vnode to `prelim_defs`. See section 7.7.2 “Natural Vnodes” on page 206.

3 Set the amount of memory on each vnode via `prelim_defs`.

3a Find the number of pages per vnode:

hinv -v -c memory

This will give you a list of vnodes and pages per vnode:

```
Node   Pages
 0    248836
 1    250880
 2    250880
 3    250880
 4    250880
 5    250880
 6    504831
 7    504831
 8    504832
 9    504832
10     504832
11     503671
```

3b Look in `/proc/meminfo` for the value of MemTotal. Use this value for main memory size:

```
cat /proc/meminfo

MemTotal:      72058142 kB
```

3c Calculate the amount of memory per vnode:

(main mem / total # pages) * (pages / vnode) = mem/vnode

If we use 72058142kB as the main memory size for our example, then for Vnode 0 in the example above, we would have:

(72058142kB / 4531065 total pages) * (248836) = 3957272kB

3d Set the amount of memory on each vnode. For each vnode, add a line of this form to `prelim_defs`:

```
<vnodename> resources_available.mem = \
    <MEM>
```

4 Define the placement sets you want via the `pnames` attribute. Add a line of this form to `prelim_defs`:

```
<natural vnode name> \
    pnames=<RESOURCE[,RESOURCE ...]
```

See section 9.6.9.2 "Examples of Configuring Placement Sets on an Altix" on page 334.

5 Use `pbs_mom -s insert` to create `scriptname` from `prelim_defs` and add it to the configuration files. See the section "-s script_options" on page 411 for `pbs_mom`.

```
pbs_mom -s insert <scriptname> \
    <prelim_defs>
```

6 Have the MOM re-read its configuration files:

```
pkill -HUP pbs_mom
```

6.5.6.14 Define Resources for the Altix

If you are upgrading an Altix from a pre-8.0 version of PBS Professional, you must change how resources are defined. The script "aunodeupdate.pl" does this for you. It takes the list of hosts as an argument, and defines each resource on the host's natural vnode, then defines the resources indirectly on each of the hosts' vnodes. It

does this for each resource defined on the host that is not `mem`, `vmem` or `ncpus`.

```
PBS_EXEC/etc/au_nodeupdate.pl /tmp/ \
    pbs_backup/hostlist
```

6.5.7 Migration Upgrade Under UNIX

Follow these instructions if you are upgrading an Altix that will not be using cpusets, or if you are upgrading an IRIX machine.

You can do a migration upgrade in two ways. The first way is to move your existing PBS from its place to another location, and install the new PBS in place of the old version. The steps below show how to do a migration upgrade the first way. The second way is to keep the old version of PBS where it is, and install the new version of PBS in a new location. This is useful if you want to let certain jobs complete execution.

You will probably want to move jobs from the old system to the new. During a migration upgrade, jobs cannot be running. You can checkpoint, terminate and requeue all possible jobs and requeue non-checkpointable but rerunnable jobs. Your options with non-rerunnable jobs are to either let them finish or kill them.

In the instructions below, file and directory pathnames are the PBS defaults. If you installed PBS in different locations, use your locations instead.

The following commands must be run as “root”.

6.5.7.1 Back Everything Up

Back up the server and vnode configuration information. You will use it later in the migration process.

1 On the server host, create a backup directory called `/tmp/pbs_backup`

```
mkdir /tmp/pbs_backup
```

2 Print the server attributes to a backup file in the backup directory:

```
qmgr -c “print server” > \
    /tmp/pbs_backup/server.backup
```

3 Make a copy of the server's configuration for the new PBS:

```
cp /tmp/pbs_backup/server.backup \
    /tmp/pbs_backup/server.new
```

3 Print the vnode attributes and creation commands for the default server to a backup file in the backup directory. The default server is specified in `/etc/pbs.conf`.

```
qmgr -c "print node @default" > \
    /tmp/pbs_backup/nodes.backup
```

4 Make a copy of the vnode attributes for the new PBS:

```
cp /tmp/pbs_backup/nodes.backup \
    /tmp/pbs_backup/nodes.new
```

5 Make a copy of the server's resourcedef file for the new PBS:

```
cp PBS_HOME/server_priv/resourcedef \
    /tmp/pbs_backup/resourcedef.new
```

6 Make a copy of the scheduler's sched_priv directory for the new PBS:

```
cp PBS_HOME/sched_priv/ \
    /tmp/pbs_backup/sched_priv.work
```

On each execution host, back up the MOM configuration files.

1 Make a copy of the default configuration file:

```
cp PBS_HOME/mom_priv/config \
    /tmp/pbs_backup/config.backup
```

2 If they exist (PBS 8.0 and later), make a copy of the site-defined configuration files:

```
mkdir /tmp/pbs_backup/mom_configs
```

```
$PBS_EXEC/sbin/pbs_mom -s list \
    | egrep -v '^PBS' | while read file
do
    $PBS_EXEC/sbin/pbs_mom -s show $file \
       > /tmp/pbs_backup/mom_configs/$file
done
```

Make a tarfile of the PBS_HOME and PBS_EXEC directories. This is a precaution.

1 Make a tarfile of PBS_HOME:

```
cd PBS_HOME/..

tar -cvf \
    /tmp/pbs_backup/PBS_HOME_backup.tar \
    PBS_HOME
```

2 Make a tarfile of PBS_EXEC:

```
cd PBS_EXEC/..

tar -cvf \
    /tmp/pbs_backup/PBS_EXEC_backup.tar \
     PBS_EXEC
```

Make a backup of the existing /etc/pbs.conf:

```
cp /etc/pbs.conf \
    /etc/pbs_backup/pbs.conf.backup
```

6.5.7.2 Replace Properties with Boolean Resources, Update Resources

You must replace properties with boolean resources.

1 Manually edit the vnodes configuration file for the new PBS, /tmp/pbs_backup/nodes.new. Where you find a line of the form:

```
set node NAME properties=PROP
```

or

```
set node NAME properties+=PROP
```

Replace with the line:

```
set node NAME \
    resources_available.PROP=True
```

For example, if the `qmgr` output contained the following lines:

```
set node node01 properties=red
set node node01 properties+=green
set node node02 properties=red
set node node02 properties+=blue
```

replace those lines with:

```
set node node01 \
    resources_available.red=True
set node node01 \
     resources_available.green=True
set node node02 \
    resources_available.red=True
set node node02 \
    resources_available.blue=True
```

2 Create boolean resources to replace properties. For each property being replaced, create or append to the new `/tmp/pbs_backup/resourcedef.new` file a line of the form:

```
PROP type=boolean flag=h
```

In the previous step's example, you would add the following lines to `/tmp/pbs_backup/resourcedef.new`:

```
red   type=boolean flag=h
green type=boolean flag=h
blue  type=boolean flag=h
```

You only need to add each former property once.

3 Add the “h” flag to the “n” or “f” flag for vnode-level resources listed in the new server’s `/tmp/pbs_backup/resource-def.new` file. For example, if you had:

```
switch type=string flag=n
```

This would become:

```
switch type=string flag=nh
```

See “Resource Flags” on page 225.

6.5.7.3 Remove Deprecated Terms From Server and Vnode Configurations

1 Manually edit the vnodes configuration file for the new PBS, `/tmp/pbs_backup/nodes.new`. Delete all occurrences of :

```
ntype=cluster
```

or

```
ntype=time-shared
```

Otherwise you will get a harmless error.

2 Manually edit the server’s configuration file for the new PBS, /tmp/pbs_backup/server.new. Delete all lines of the form:

```
resources_default.neednodes=X
```

6.5.7.4 Prevent Jobs From Being Enqueued or Started

You must deactivate the scheduler and queues. When the server’s `scheduling` attribute is false, jobs are not started by the scheduler. When the queues’ `enabled` attribute is false, jobs cannot be enqueued.

1 Prevent the scheduler from starting jobs:

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

2 Print a list of all queues managed by the server. Save the list of queue names for the next step.

```
qstat -q
```

3 Disable queues to stop jobs from being enqueued. Do this for each queue in your list from the previous step.

```
qdisable <queue name>
```

6.5.7.5 Shut Down PBS Professional

You can now shut down the server, scheduler, and MOM daemons. Use the `-t immediate` option to `qterm` so that all possible running jobs will be requeued.

1 Shut down PBS:

```
qterm -t immediate -m -s -f
```

If your server is not running in a failover environment, the "`-f`" option is not required.

2 Verify that PBS daemons are not running in the background:

```
ps -ef | grep pbs
```

If you see the `pbs_server`, `pbs_sched`, or `pbs_mom` process running, you will need to manually terminate that process:

```
kill -9 <pid>
```

6.5.7.6 Back Up the Server's Jobs Directory

You must back up any jobs that were queued when the server was shut down, in order to move them to the new version of PBS.

1 Make a tarfile of the jobs directory, and save it in the backup directory:

```
cd PBS_HOME/server_priv
```

```
tar -cvf \
```

```
/tmp/pbs_backup/pbs_jobs_save.tar jobs
```

6.5.7.7 Back Up PBS Directories and Configuration Files

Back up the PBS_HOME and PBS_EXEC directories and PBS configuration files. You will use this later.

1. Rename the PBS_HOME directory:

```
mv PBS_HOME PBS_HOME.backup
```

2. Rename the PBS_EXEC directory:

```
mv PBS_EXEC PBS_EXEC.backup
```

3. Copy the pbs.conf file to the backup directory:

```
cp /etc/pbs.conf \
    /tmp/pbs_backup/pbs.conf.backup
```

6.5.7.8 Install the New Version of PBS

1. On each host, go to the directory containing the package for the new version of PBS. Unzip and untar or otherwise unpack the new version of PBS. Install the new version of PBS.

```
cd <location of package>
```

```
tar -xvf <pbs package>
```

```
./INSTALL
```

2. If that machine will be the server and scheduler host, select option #1.

3. When asked for the license file location, give the location of the FLEXlm license file.

4. When asked whether to start PBS, DO NOT start the server. You will manually start the server later.

5. Restore the default MOM configuration file.

```
cp /tmp/pbs_backup/config.backup \
    PBS_HOME/mom_priv/config
```

6 If they exist (PBS 8.0 and later), restore the site-defined MOM configuration files. For each of these that you backed up:

```
cd /tmp/pbs_backup/mom_configs
for file in *
do
    $PBS_EXEC/sbin/pbs_mom -s insert $file
$file
done
```

6.5.7.9 Update Fairshare Entities and Database

This step is only necessary if you were not using egroup:euser as your fairshare entity, but wish to do so now. Follow the procedures in section 6.5.10 "Updating Fairshare Definitions for egroup:euser" on page 149.

6.5.7.10 Copy License and Resource Files

1 If you are upgrading from PBS 9.0 or later and using a three-server redundant license server setup, copy the old license server host file from the backup directory to the default directory:

```
cp PBS_HOME.backup/server_priv/< three-server
host file> \
    PBS_HOME/server_priv/
```

2 Copy the modified server resource file from the backup directory to the default directory:

```
cp /etc/pbs_backup/resourcedef.new \
    PBS_HOME/server_priv/resourcedef
```

6.5.7.11 Start the New Server Without Defined Queues or Vnodes

When the new server starts up it will have default queue workq and the server host already defined. You want to start the new server with empty configurations so that you can import your old settings.

1 Remove the new server's default nodes file:

```
rm PBS_HOME/server_priv/nodes
```

2 Start the new server with empty queue and vnode configurations:

```
PBS_EXEC/sbin/pbs_server -t create
```

A message will appear saying "`Create mode and server database exists, do you wish to continue?`"

Type "**`y`**" to continue.

Because of the new 9.0 licensing scheme an additional message appears:

"One or more PBS license keys are invalid, jobs may not run"

This message is expected. Continue to the next step in these instructions.

6.5.8 Set Server's License Location Attribute

Follow the steps in section 4.12 "Setting the pbs_license_file_location Attribute" on page 80

6.5.8.1 Replicate Queues and Server and Vnodes Configuration

1 Give the new server the old server's configuration, but modified for the new PBS:

```
PBS_EXEC/bin/qmgr < \
    /tmp/pbs_backup/server.new
```

2 Replicate vnodes configuration, also modified for the new PBS:

```
PBS_EXEC/bin/qmgr < \
    /tmp/pbs_backup/nodes.new
```

The new version of PBS will write out its nodes file in a new format, but only when the server is shut down or a vnode is added or deleted. Therefore you will see the old format until this happens.

3 Verify the original configurations were read in properly:

```
PBS_EXEC/bin/qmgr -c "print server"
PBS_EXEC/bin/pbsnodes -a
```

6.5.8.2 Start the Old Server

You must start the old server in order to move jobs to the new server. The old server must be started on **alternate** ports.

1 Start the old server daemon, and assign these port values:

```
PBS_EXEC.backup/sbin/pbs_server -p 13001 \
    -M 13002 -R 13003 -S 13004 -g 13005 \
    -d PBS_HOME.backup
```

-p	Port number on which server listens for batch requests
-M	Port number on which server connects to MOM daemon
-R	Port number on which server queries status of MOM
-S	Port number on which server connects to scheduler daemon
-g	Port number on which server connects to PBS MOM Globus daemon

-d	Path of directory containing server's configuration

For more information see "Manually Starting the Server" on page 412 and the `pbs_server(8B)` man page.

6.5.8.3 Move Existing Jobs to the New Server

You must move existing jobs from the old server to the new server. To do this, you run the move commands from the old server, and give the new server's port number , 15001, in the destination. See the `qmove(1B)` man page.

This is for the special case where your old PBS version is older than 5.4.0 and the old server's host also runs a MOM:

1 Delete the vnode on the server's host:

```
PBS_EXEC.backup/bin/qmgr -c \
    "d n <old server host>" \
    <old server host>:13001
```

If you see the message, "Cannot delete busy object", get a list of jobs running on that vnode:

```
PBS_EXEC.backup/bin/qstat \
    @<old server host>:13001
```

2 Either requeue or kill the jobs on the server's host:

```
PBS_EXEC.backup/bin/qrerun -W force \
    <job id>
```

For PBS versions 5.4.0 and later, if your old server's host also ran a MOM, you will need to delete that vnode from the old server.

Delete the vnode on the old server's host:

```
PBS_EXEC.backup/bin/qmgr -c \
    "d n <old server host>" \
    <old server host>:13001
```

Move jobs from the old server to the new one:

1 Print the list of jobs on the old server:

```
PBS_EXEC.backup/bin/qstat \
    @<old server host>:13001
```

2 Move each job from each queue:

```
PBS_EXEC.backup/bin/qmove \
    <new queue \
    name>@<new server host>:15001 \
    <job id>@<old server host>:13001
```

You can use qselect to select all the jobs within a queue instead of moving each job individually.

3 Move all jobs within a queue:

```
PBS_EXEC.backup/bin/qmove \
    <queue name>@<new server host>:15001 \
    `PBS_EXEC.backup/bin/qselect -q  \
    <queue name>@<old server host>:13001`
```

If you see the error message "Too many arguments...", there are too many jobs to fit in the shell's command line buffer. You can continue moving jobs one at a time until there are few enough.

6.5.8.4 Shut Down Old Server

Shut down the old server daemon:

```
PBS_EXEC.backup/bin/qterm \
    -t quick <old server host>:13001
```

6.5.8.5 Update New `sched_config`

Update the new scheduler's configuration file, in `PBS_HOME/sched_priv/sched_config`, with any modifications that were made to the old `PBS_HOME.old/sched_priv/sched_config`.

1 If you copied over your old scheduler log filter value, make sure that it has had 1024 added to it. If the value is less than 1024, add 1024 to it. For example, if the old log filter line is:

```
log_filter:256
```

change it to:

```
log_filter:1280
```

2 If it exists, replace the `strict_fifo` option with `strict_ordering`. If you do not, a warning will be printed in the log when the scheduler starts.

6.5.8.6 Start New Scheduler

Start the scheduler daemon, on the server's host:

```
PBS_EXEC/sbin/pbs_sched
```

6.5.8.7 Start New MOMs

On each execution host:

```
PBS_EXEC/sbin/pbs_mom
```

6.5.8.8 Optionally Start MOM on New Server's Host

If your old configuration had a MOM running on the server's host, and you wish to replicate the configuration, you can start a MOM on that machine.

Start the MOM daemon on the new server's host:

```
PBS_EXEC/sbin/pbs_mom
```

6.5.8.9 Enable Scheduling in New Server

You must set the new server's scheduling attribute to true so that the scheduler will start jobs.

Enable scheduling for the new server:

```
PBS_EXEC/bin/qmgr -c "s s scheduling=1"
```

6.5.9 Upgrading Under IRIX

See section 11.6.4 "Checkpointing Jobs Prior to SGI IRIX Upgrade" on page 424.

6.5.10 Updating Fairshare Definitions for egroup:euser

This step is only necessary if you were not using egroup:euser as your fairshare entity, but wish to do so now. See section 9.15.4.3 "Format for Describing the Tree" on page 357 for information on using egroup:euser.

On the server/scheduler host:

1. Update `/tmp/pbs_backup_sched_priv.work/resource_group`, with the new egroup:euser entities. See section 9.15.4.3 "Format for Describing the Tree" on page 357.

   If you previously had an euser named Bob, the new entry for Bob in the `resource_group` file will look like this:

   ```
   pbsgroup1:Bob      101     Phys  20
   ```

2. Restore the modified contents of the scheduler's directory:

   ```
   cp -r /tmp/pbs_backup/sched_priv.work \
       PBS_HOME/sched_priv \
   ```

If you wish to keep your old fairshare resource usage data, use the `pbsfs` command to update the data. See the pbsfs(8B) manual page.

When you display the usage data, both the old and the new entities will appear. Entities defined before the upgrade will appear in the unknown group. Entities defined in the previous step will be in the correct department but have no usage.

1. Display the usage data:

   ```
   % pbsfs -p
   ```

If the old entity was Bob:

```
Bob : group: 1   cgrp: 5   shares: 10  \
    Usage: 1005   Perc: 50%
```

You will see usage of 1005 for Bob, as shown above.

The newly defined entity is pbsgroup1:Bob, and it will appear like this:

```
pbsgroup1:Bob   5      4      root      10
```

2 Assign Bob's usage to pbsgroup1:Bob:

```
% pbsfs -s pbsgroup1:Bob 1005
```

3 Remove the old entities:

```
% pbsfs -e
```

If your previous entities were euser, and now you have egroup:euser, and you have a user who is in more than one group, you will need to make an entry in `/tmp/pbs_backup_sched_priv.work/resource_group` for Bob in each of his groups, then assign a portion of the Bob's usage to each group.

For example, if you wish to have Bob be in two groups, pbsgroup1 and pbsgroup2, you may want Bob's usage to be split between the groups. Bob will show up as pbsgroup1:Bob and pbsgroup2:Bob.

1 Update `/tmp/pbs_backup_sched_priv.work/resource_group`, putting Bob into pbsgroup1 and pbsgroup2. See section 9.15.4.3 "Format for Describing the Tree" on page 357.

The entries for Bob in the resource_group file will look like this:

```
pbsgroup1:Bob      101     Phys  20
pbsgroup2:Bob      102     Math  20
```

2 Assign half of Bob's usage to each of pbsgroup1:Bob and pbsgroup2:Bob:

```
% pbsfs -s pbsgroup1:Bob 502
% pbsfs -s pbsgroup1:Bob 503
```

6.6 Upgrading Under Windows

You must use a migration upgrade under Microsoft Windows.

When you do a migration upgrade under Windows, you can install the new version of PBS in the same place or in a new location.

You will probably want to move jobs from the old system to the new. During a migration upgrade, jobs cannot be running. You can requeue rerunnable jobs. Your can let non-rerunnable jobs finish, or you can kill them.

If you are migrating from a version before 9.0 to 9.0 or later, all the following steps must be done by a domain administrator. If the migration is taking place in a domain environment, this Administrator account should be a member of the "Domain Admins" group. If the migration is taking place in a standalone environment, this Administrator account should be a member of the local "Administrators" group.

If you want the PBS service account, "pbsadmin", not to be a domain administrator for the new version of PBS, follow the steps specified in section 6.6.24 "Optionally Change PBS Service Account "pbsadmin" to Non-domain Administrator Account" on page 172 before starting PBS.

In the instructions below, file and directory pathnames are the PBS defaults. If you installed PBS in different locations, use your locations instead. Where you see `%WINDIR%`, it will be automatically replaced by the correct directory. For Windows XP, that is `\WINDOWS`. For Windows 2000, it is `\WINNT`.

The default server is specified in `\Program Files\PBS Pro\pbs.conf`.

Note that in version 8.0 and later, job scripts under Windows are executed differently. Any .bat files that are to be executed within a PBS job script will have to be prefixed with "call" as in:

```
---[job_b.bat]----------
@echo off
call E:\step1.bat
call E:\step2.bat
-----------------------
```

Without the "call", only the first .bat file gets executed, and it doesn't return control to the calling interpreter.

6.6.1 Prevent Jobs From Being Enqueued or Started

You must deactivate the scheduler and queues. When the server's `scheduling` attribute is false, jobs are not started by the scheduler. When the queues' `enabled` attribute is false, jobs cannot be enqueued.

1 Prevent the scheduler from starting jobs:

qmgr -c "set server scheduling = false"

2 Print a list of all queues managed by the server. Save the list of queue names. You will need it in the next step and when moving jobs.

qstat -q

3 Disable queues to stop jobs from being enqueued. Do this for each queue in your list from the previous step.

qdisable <queue name>

6.6.2 Back Everything Up

Back up the server and vnode configuration information. You will use it later in the migration process.

1 Make a backup directory:

mkdir "%WINDIR%\TEMP\PBS Pro Backup"

2 Print the server attributes to a backup file in the backup direc-

tory:

```
qmgr -c "print server" >
    "%WINDIR%\TEMP\PBS Pro Backup\server.backup"
```

3 Make a copy of the server's configuration for the new PBS:

```
copy "%WINDIR%\PBS Pro
Backup\server.backup"
"%WINDIR%\TEMP\PBS Pro Backup\server.new"
```

4 Print the default server's vnode attributes to a backup file in the backup directory.

```
qmgr -c "print node @default" >
    "%WINDIR%\TEMP\PBS Pro Backup\nodes.backup"
```

5 Make a copy of the vnode attributes for the new PBS:

```
copy "%WINDIR%\TEMP\PBS Pro Backup\nodes.backup"
    "%WINDIR%\TEMP\PBS Pro Backup\nodes.new"
```

6 Make a backup of the existing `\Program Files\PBS Pro\pbs.conf`. This command is all one line:

```
copy "\Program Files\PBS Pro\pbs.conf"
    "%WINDIR%\TEMP\PBS Pro
    Backup\pbs.conf.backup"
```

7 Make a copy of pbs.conf for the new PBS. This command is all one line:

```
copy "%WINDIR%\TEMP\PBS Pro
    Backup\pbs.conf.backup"
    "%WINDIR%\TEMP\PBS Pro
    Backup\pbs.conf.new"
```

8 Make a copy of the server's resourcedef for the new PBS. This command is all one line:

```
copy "%WINDIR%\TEMP\PBS Pro
```

```
Backup\home\server_priv\resourcedef"

"%WINDIR%\TEMP\PBS Pro Backup
\home\server_priv\resourcedef.new"
```

9 Make a backup of the existing `\Program Files\PBS Pro\sched_priv`. This command is all one line:

```
copy "\Program Files\PBS Pro\sched_priv"
    "%WINDIR%\TEMP\PBS Pro
    Backup\sched_priv.backup"
```

10 Make a copy of the scheduler's directory for the new PBS. This command is all one line:

```
copy "%WINDIR%\TEMP\PBS Pro
    Backup\home\sched_priv.backup"

    "%WINDIR%\TEMP\PBS Pro Backup
    \home\sched_priv.work"
```

6.6.3 Shut Down PBS Professional

You can now shut down the server, scheduler, and MOM daemons. Use the `-t immediate` option to the `qterm` command so that all possible running jobs will be requeued.

1 Shut down PBS. If your server is not running in a failover environment, the "`-f`" option is not required.

```
qterm -t immediate -m -s -f
```

2 Stop the `pbs_rshd` daemon:

```
net stop pbs_rshd
```

6.6.4 Copy the Old Version of PBS to a Temporary Location

You will run the old PBS server from this temporary location in order to move jobs. You must do a copy rather than a move, because the installation software depends on the old version of PBS being available for it to remove.

On the Server vnode, copy the existing PBS_HOME and PBS_EXEC hierarchies to a temporary location. This command is all one line.

```
xcopy /o /e "\Program Files\PBS Pro"
    "%WINDIR%\TEMP\PBS Pro Backup"
```

Specify "D" for directory when prompted.

If you get an "access denied" error message while it is moving a file:

1. Bring up `Start menu->Programs->Accessories->Windows Explorer`,
2. Right-click to select this file and bring up a pop-up menu.
3. Choose "Properties", then "Security" tab, then "Advanced", then "Owners" tab.
4. Reset the ownership of the file to "Administrators". "Administrators" must have permission to read the file.
5. Rerun `xcopy`.

6.6.5 Replace Properties with Boolean Resources and Update Resources

You must replace properties with boolean resources, and add a new flag for vnode-level resources.

1. Manually edit the vnodes configuration file for the new PBS, %WINDIR%\TEMP\PBS Pro Backup\nodes.new. Where you find a line of the form:

   ```
   set node NAME properties=PROP
   ```

   or

   ```
   set node NAME properties+=PROP
   ```

   Replace with the line as a single line:

```
set node NAME
    resources_available.PROP=True
```

For example, if the `qmgr` output contained the following lines:

```
set node node01 properties=red
set node node01 properties+=green
set node node02 properties=red
set node node02 properties+=blue
```

replace those lines with these. Each is one line:

```
set node node01 \
    resources_available.red=True
set node node01 \
     resources_available.green=True
set node node02
resources_available.red=True
set node node02
resources_available.blue=True
```

2 Create boolean resources to replace properties. For each property being replaced, create or append to the new `%WINDIR%\TEMP\PBS Pro Backup\home\server_priv\resourcedef.new` file a line of the form:

```
PROP type=boolean flag=h
```

In the previous step's example, you would add the following lines to `%WINDIR%\TEMP\PBS Pro Backup\home\server_priv\resourcedef.new`:

```
red   type=boolean flag=h
green type=boolean flag=h
blue  type=boolean flag=h
```

You only need to add each former property once.

3 Add the "h" flag to the "n" or "f" flag for vnode-level resources

listed in the new server's
`%WINDIR%\TEMP\PBS Pro Backup\home\server_priv\resourcedef.new` file.

6.6.6 Remove Deprecated Terms from Server and Vnode Configurations

1 Manually edit the vnodes configuration file for the new PBS, `%WINDIR%\TEMP\PBS Pro Backup\nodes.new`. Delete all occurrences of :

`ntype=cluster`

or

`ntype=time-shared`

Otherwise you will get a harmless error.

2 Manually edit the server's configuration file for the new PBS, `%WINDIR%\TEMP\PBS Pro Backup\server.new`. Delete all lines of the form:

`resources_default.neednodes=X`

6.6.7 Install the New Version of PBS on Execution Hosts

You can install PBS from a CD or by downloading it.

If you are installing PBS from the CD, put it in the CD-ROM drive, browse to your CD-ROM drive, and click on the PBS Professional icon.

If you are installing PBS from a download, save it to your hard drive and run the self-extracting `pbspro.exe` package, either in the same directory, or by giving the path to it.

You must use the same password on all hosts. Do the following on each execution host, **except** the server's host.

Uninstall the old version of PBS:

1 Change your current directory so that it is not within `C:\Pro-`

gram Files\PBS Pro, and make sure there is no access occurring on any file in that hierarchy. Otherwise you will have to remove the hierarchy by hand.

2 Run the installation program by either clicking on the icon, or typing:

PBSpro_<version>-windows.exe

Under XP, SP2, you may see a warning saying, "The publisher could not be verified. Are you sure you want to run this software?" Ignore this message and click the "Run" button.

The installation package will ask whether you want to uninstall the previous version. Answer yes.

If you see a popup window saying the hierarchy could not be removed, remove the hierarchy manually by going to a command window and typing the following. Do not use the `del` command.

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

3 Reboot the execution host.

Install the new version of PBS:

1 Run the installation program by either clicking on its icon, or typing:

PBSpro_<version>-windows.exe

Under XP, SP2, you may see a warning saying, "The publisher could not be verified. Are you sure you want to run this software?" Ignore this message and click the "Run" button.

2 Check the installation location. You can change it, as long as the new location meets the requirements given in section 3.7 "Recommended PBS Configurations for Windows" on page 21.

3 Choose the "Execution" option.

4 Enter a non-empty password twice for the Administrator

account. If you see "error 2245", check the password's length, complexity and history requirements. Look in the Active Directory guide or the Windows help page.

5 Give the server's hostname in the "Editing PBS.CONF file" window.

6 In the "Editing HOSTS.EQUIV file" window, enter any hosts and/or users that will need access to this execution host.

7 In the "Editing PBS MOM config file" window, accept the defaults.

8 Restart the execution host and log into it.

9 Stop the PBS MOM:

```
net stop pbs_mom
```

6.6.8 Install the New Version of PBS on the Server's Host

You must use the same password for the Administrator account on all hosts. You can accept the trial license during installation.

If you are installing PBS from the CD, put it in the CD-ROM drive, browse to your CD-ROM drive, and click on the PBS Professional icon.

If you are installing PBS from a download, save it to your hard drive and run the self-extracting `pbspro.exe` package, either in the same directory, or by giving the path to it.

Uninstall the old version of PBS:

1 Change your current directory so that it is not within `C:\Program Files\PBS Pro`, and make sure there is no access occurring on any file in that hierarchy. Otherwise you will have to remove the hierarchy by hand.

2 Run the installation program by either clicking on the icon, or typing:

PBSpro_<version>-windows.exe

Under XP, SP2, you may see a warning saying, "The publisher could not be verified. Are you sure you want to run this software?" Ignore this message and click the "Run" button.

The installation package will ask whether you want to uninstall the previous version. Answer yes.

If you see a popup window saying the hierarchy could not be removed, remove the hierarchy manually by going to a command window and typing the following. Do not use the `del` command.

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

3 Reboot the server's host.

Install the new version of PBS:

1 Run the installation program by either clicking on the icon, or typing:

PBSpro_<version>-windows.exe

Under Windows XP SP2, you may see a warning saying, "The publisher could not be verified. Are you sure you want to run this software?" Ignore this message and click the "Run" button.

2 Check the installation location. You can change it, as long as the new location meets the requirements given in section 3.7 "Recommended PBS Configurations for Windows" on page 21.

3 Choose the "All" option.

4 When you are prompted for a license file location, enter the information.

5 Enter a non-empty password twice for the "pbsadmin" account. If you see "error 2245", check the password's length, complexity and history requirements. Look in the Active Directory guide or the Windows help page.

6 In the "Editing HOSTS.EQUIV file" window, enter any hosts and/or users that will need access to the server's host.

7 In the "PBS Server Nodes File" window, accept the defaults.

8 In the "PBS MOM Config File for local node" window, accept the defaults.

9 Restart the server's host, and log into it.

Stop the PBS MOM on the server's host:

```
net stop pbs_mom
```

6.6.9 Copy License and Resource Files

The new version of PBS will come with a trial license. If your license has expired, you can use this while you get anew one.

1 Save the trial license. This command is all one line:

```
copy "\Program Files\PBS Pro\
    home\server_priv\license_file"
    "\Program Files\PBS Pro\
    home\server_priv\license_file.trial"
```

2 If you are upgrading from PBS 9.0 or later and using a three-server redundant license server setup, copy the old license server host file from the backup directory to the default directory: Type this command in a single line:

```
copy "%WINDIR%\TEMP\PBS Pro Backup\
    home\server_priv\< three-server \
    host file>"\
    Program Files\PBS Pro\
    home\server_priv\< three-server \
    host file>"
```

3 Copy the server's resourcedef file from the backup directory to the default directory. Type this command in a single line:

```
copy "%WINDIR%\TEMP\PBS Pro Backup\
    home\server_priv\resourcedef.new"
    "\Program Files\PBS  Pro\
    home\server_priv\resourcedef"
```

6.6.10 Update Fairshare Definitions

This step is only necessary if you were not using egroup:euser as your fairshare entity, but wish to do so now. On the server/scheduler host:

1. Update `%WINDIR%\TEMP\PBS Pro Backup\sched_priv.work\resource_group`, with the new egroup:euser entities. See section 9.15.4.3 “Format for Describing the Tree” on page 357.

2. Restore the scheduler’s modified directory:

   ```
   copy "%WINDIR%\TEMP\PBS Pro Backup\
       home\sched_priv.work"
       "\Program Files\PBS  Pro\
       home\sched_priv\"
   ```

3. If you wish to keep your old fairshare resource usage data, use the `pbsfs` command to update the data.

   Display the usage data. Both the old and the new entities will appear. Entities defined before the upgrade will appear in the unknown group. Entities defined in the previous step will be in the correct department but have no usage.

   Display the usage data:

   ```
   pbsfs -p
   ```

   Assign the usage of the old entities to the new ones.

   Be aware that the usage may not reflect what you want. For example, if your previous entities were euser, and now you have egroup:euser, and you have a user who is in more than one group, you will need to divide the user’s usage so that it shows the usage by group that you want.

Old entity: Bob

```
% pbsfs -p

Bob    : group: 1   cgrp: 5   shares: 10  \
    Usage: 1005   Perc: 50%
```

The useful information is the name "Bob" and the usage of 1005

The newly defined entity is Math:Bob

```
Math:Bob   5       4       root       10
```

Assign Bob's usage to Math:Bob:

```
% pbsfs -s math:bob 1005
```

Remove the old entities:

```
% pbsfs -e
```

6.6.11 Start the New Server Without Defined Queues or Vnodes

When the new server starts it will have the default queue, workq, and server vnode already defined. You want to start the new server with empty configurations so that you can import your old settings.

1. Stop the new server:

   ```
   qterm
   ```

2. Remove the new server's default nodes file. Type this command in a single line:

   ```
   del "\Program Files\PBS Pro\
       home\server_priv\nodes"
   ```

3. Start the new server as a standalone, having it create a new database. Type this command in a single line:

```
"\Program Files\PBS Pro\
    exec\sbin\pbs_server" -N -t create
```

A message will appear saying "Create mode and server database exists, do you wish to continue?"

Type "**y**" to continue. The standalone server will exit after it creates the database.

Type Ctrl-C in order to get the terminal/command prompt back.

4 Start the new server as a Windows service:

```
net start pbs_server
```

5 Start the new scheduler as a Windows service:

```
net start pbs_sched
```

6.6.12 Set Server's License Location Attribute

Follow the steps in section 4.12 "Setting the pbs_license_file_location Attribute" on page 80.

6.6.13 Replicate Queues, Server and Vnodes Configuration

1 Give the new server the old server's configuration, but modified for the new PBS. Type this command in a single line:

```
"\Program Files\PBS Pro\exec\bin\qmgr" <
    "%WINDIR%\TEMP\PBS Pro Backup\
    server.new"
```

2 List the queues in the new server:

```
"\Program Files\PBS Pro\exec\bin\qstat -Q"
```

3 Enable the queues in the new server. For each queue where its enabled attribute is false:

```
"\Program Files\PBS Pro\exec\bin\qenable"
<queue name>
```

4 Replicate vnodes configuration, also modified for the new PBS. Type this command in a single line:

```
"\Program Files\PBS Pro\exec\bin\qmgr" <
    "%WINDIR%\TEMP\PBS Pro Backup\
    nodes.new"
```

The new version of PBS will write out its nodes file in a new format, but only when the server is shut down or a vnode is added or deleted. Therefore you will see the old format until this happens.

5 Verify that the original configurations were read in properly. Type this command in a single line:

```
"\Program Files\PBS Pro\exec\bin\qmgr" -c
    "print server"
    "\Program Files\PBS Pro\exec\bin\
    pbsnodes" -a
```

6.6.14 Start the Old Server

You must start the old server in order to move jobs to the new server. The old server must be started on alternate ports. Type the following commands without breaking the lines.

1 If you are upgrading from PBS Pro_5.3.3-wpl:

```
del "%WINDIR%\TEMP\PBS Pro Backup\
    home\server_priv\server.lock"
```

2 Tell PBS to use the `pbs.conf` file you saved in the backup directory, and to use the backup `exec` and `home` directories. Do not put any double quotes in either path:

```
set PBS_CONF_FILE=%WINDIR%\
    TEMP\PBS Pro Backup\pbs.conf

pbs-config-add "PBS_EXEC=%WINDIR%\
    TEMP\PBS Pro Backup\exec"
```

```
pbs-config-add "PBS_HOME=%WINDIR%\
    TEMP\PBS Pro Backup\home"
```

3 Verify that the old server is using the `pbs.conf` saved in the backup directory:

```
echo %PBS_CONF_FILE%
```

4 Verify that `pbs.conf` contains the `exec` and `home` locations in the backup directory:

```
type %PBS_CONF_FILE%
```

5 Start the old server daemon in the same command prompt window as above, and assign these alternate port values:

```
"%WINDIR%\TEMP\PBS Pro Backup\
    exec\sbin\pbs_server" -N -p 13001
    -M 13002 -R 13003 -S 13004 -g 13005
```

-p	Port number on which server listens for batch requests
-M	Port number on which server connects to MOM daemon
-R	Port number on which server queries status of MOM
-S	Port number on which server connects to scheduler daemon
-g	Port number on which server connects to PBS MOM Globus daemon
-d	Path of directory containing server's configuration
-N	Run in standalone mode

For more information see "Starting and Stopping PBS: Windows 2000 / XP" on page 421 and the `pbs_server(8B)` man page.

6.6.15 Verify Old Server is Running on Alternate Ports

Verify that the old `pbs_server` is running on the alternate ports by going to another `cmd` prompt window and running, typed as a single line:

```
"%WINDIR%\TEMP\PBS Pro Backup\exec\bin\
    qstat" @<old server host>:13001
```

6.6.16 Migrate User Passwords From the Old Server to the New Server

You will want to migrate user passwords to the new server if possible. Passwords can only be migrated if both the old and new servers' `single_signon_password_enable` attributes are `true`. The following commands should be given in a single line:

1. Find out whether the old server's `single_signon_password_enable` attribute is `true`:

```
"%WINDIR%\TEMP\PBS Pro Backup\
    exec\bin\qmgr" -c "list s
    single_signon_password_enable" <old
server host>:13001
```

2. Find out whether the new server's `single_signon_password_enable` attribute is `true`:

```
"\Program Files\PBS Pro\exec\bin\qmgr"
    <new server host>:15001

Qmgr: list s single_signon_password_enable
```

3. If both attributes are `true`, you can migrate user passwords from the old server to the new server:

```
"\Program Files\PBS Pro\exec\bin\
    pbs_migrate_users"
    <old server host>:13001
    <new server host>:15001
```

6.6.17 Move Existing Jobs to the New Server

You must move existing jobs from the old server to the new server. To do this, you run the new `qselect` and `qmove` commands, and give the new server's port number , 15001, in the destination. See the `qmove(1B)` information in the **PBS Professional User's Guide**.

There is one special case requiring an extra step. This is when the old server's `single_signon_password_enable` attribute is `false` and the new server's is `true`.

Give commands in a single line.

1 If the new server's `single_signon_password_enable` attribute is `true` and the old server's is `false`, temporarily set the new server's `single_signon_password_enable` to `false`:

```
"\Program Files\PBS Pro\exec\bin\qmgr"
    <new server host>:15001

Qmgr: set server
    single_signon_password_enable=false
```

2 You will need to verify later that all jobs have been moved. Print the list of jobs on the old server:

```
"%WINDIR%\TEMP\PBS Pro Backup\exec\
    bin\qstat" @<old server host>:13001
```

3 In another command prompt window, move each job in each queue.
This command may tie up the terminal. Create a file called `movejobs.bat`, containing the following lines. Replace `<old server host>` with the old server's host:

```
REM movejobs.bat
REM execute as follows:
REM
REM    movejobs <queue name>
    (e.g. movejobs workq)
REM
setlocal ENABLEDELAYEDEXPANSION
for /F "usebackq" %%j in (`"\Program Files\
```

```
PBS Pro\exec\bin\qselect" -q
%1@<old server host>:13001`)
do ("\Program Files\PBS Pro\
    exec\bin\qmove"
    %1@<new server host>:15001
    %%j@<old server host>:13001
)
```

Run `movejobs.bat` for each queue. Use the list of queue names saved when you disabled the queues.

For example, to move the jobs in `queue1` and `workq`, you would type:

```
cmd>movejobs queue1
cmd>movejobs workq
```

4 Verify that all jobs have been moved. Print the jobs on the new server:

```
"\Program Files\PBS Pro\exec\bin\qstat"
    @<new server host>:15001
```

5 Special case: If the old server's `single_signon_password_enable` attribute is false and the new server's was true (but is temporarily false), you must do the following three steps:

a. Apply a bad password hold to all the jobs on the new server. Create a file called `pholdjobs.bat`, containing the following lines. Replace `<new server host>` with the new server's host:

```
REM pholdjobs.bat
REM execute as follows:
REM
REM    pholdjobs
REM
setlocal ENABLEDELAYEDEXPANSION
for /F "usebackq" %%j in
(`"\Program Files\PBS Pro\exec\bin\
```

```
qselect" -q @<new server host>:15001`)
do (
"\Program Files\PBS Pro\exec\bin\qhold"
-h p %%j@<new server host>:15001
)
```

Run `pholdjobs.bat` by typing:

```
pholdjobs
```

b. Change the new server's attribute back to `true`.

```
"\Program Files\PBS Pro\exec\bin\qmgr"
<new server host>:15001
Qmgr:  set server
single_signon_password_enable=true
```

c. Each user with jobs on the new server must specify a password via `pbs_password`. See the **PBS Professional User's Guide**. Each user will type **`pbs_password`** and be prompted for a password.

6.6.18 Shut Down Old Server

1 Shut down the old server daemon:

```
"%WINDIR%\TEMP\PBS Pro Backup\
    exec\bin\qterm"   -t quick
    <old server host>:13001
```

6.6.19 Update New `sched_config`

Update the new scheduler's configuration file, in `\Program Files\PBS Pro\home\sched_priv\sched_config`, with any modifications that were made to the old `%WINDIR%\TEMP\PBS Pro Backup\home\sched_priv\sched_config.`

1 If you copied over your old scheduler log filter value, make sure that it has had 1024 added to it. If the value is less than 1024, add 1024 to it. For example, if the old log filter line is:

```
log_filter:256
```

change it to:

```
log_filter:1280
```

2 If it exists, replace the `strict_fifo` option with `strict_ordering`. If you do not, a warning will be printed in the log when the scheduler starts.

6.6.20 Start MOMs on Execution Hosts

1 On each execution host, start the MOM daemon as a Windows service:

```
net start pbs_mom
```

6.6.21 Optionally Start MOM on New Server's Host

If your old configuration had a MOM running on the server's host, and you wish to replicate the configuration, you can start a MOM on that machine.

1 Start the MOM daemon on the new server's host as a Windows service:

```
net start pbs_mom
```

6.6.22 Verify Communication Between Server and MOMs

1 Run `pbsnodes -a` on the server's host to see if it can communicate with the execution hosts in your complex. If a host is down, go to the problem host and restart the MOM:

```
net stop pbs_mom
net start pbs_mom
```

6.6.23 Enable Scheduling in the New Server

You must set the new server's `scheduling` attribute to `true` so that the scheduler will start jobs.

1 Enable scheduling in the new server:

```
"\Program Files\PBS Pro\exec\bin\
    qmgr" -c "set server scheduling=1"
```

6.6.24 Optionally Change PBS Service Account "pbsadmin" to Non-domain Administrator Account

If you want to run PBS Professional from a PBS service account which is not a domain administrator account, follow these steps:

1 Stop all PBS services:

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

2 Add the PBS service account "pbsadmin" to the local Administrators group:

```
net localgroup Administrators <domain
name>\pbsadmin /add
```

3 Go to Active Directory, and make pbsadmin be only a member of the "Domain Users" group. Add the account to "Domain Users" on the local computer, and make it the primary group. Then remove pbsadmin's membership in the "Domain Admins" group.

4 Delegate explicit domain read privilege to the pbsadmin account. See section 3.7.2.4 "Delegating Read Access" on page 24.

5 Restart the PBS services:

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

Chapter 7
Configuring the Server

The next three chapters will walk you through the process of configuring the Server, the MOMs and the scheduling policy. 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.

7.1 The qmgr Command

The PBS manager command, `qmgr`, provides a command-line interface to the PBS Server. The qmgr command can be used by anyone to list or print attributes. Operator privilege is required to be able to set or unset vnode, queue or server attributes. Manager privilege is required to create or delete queues or vnodes. The qmgr command will not display attributes which are unset, i.e. are at their default value.

Most of a vnode's attributes may be set using qmgr. However, some **must** be set on the individual execution host in local vnode definition files, NOT by using qmgr. Those that must be set on the execution host this way are

sharing
ncpus
mem
vmem

An example of the way to do this (in this case, changing the "sharing" attribute for a vnode named V10) uses the script “change_sharing”. See section 8.2.1 “Creation of Site-defined MOM Configuration Files” on page 259.

```
# cat change_sharing
$configversion 2
V10:  sharing = ignore_excl
# . /etc/pbs.conf
# $PBS_EXEC/sbin/pbs_mom -s insert ignore_excl
      change_sharing
# pkill -HUP pbs_mom
```

Do **not** set sharing, ncpus, mem, or vmem on a vnode via qmgr.

The `qmgr` command usage is:

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

The available options, and description of each, follows.

Option	Action
`-a`	Abort `qmgr` on any syntax errors or any requests rejected by a Server.
`-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.
`--version`	The qmgr command returns its PBS version information and exits. This option can only be used alone.

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.

Any attribute value set via `qmgr` containing commas, whitespace or the hashmark must be enclosed in double quotes. For example:

```
     Qmgr: set node Vnode1 comment="Node will be taken
offline Friday at 1:00 for memory upgrade."
     Qmgr: active node vnode1,vnode2,vnode3
```

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 the 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 (OP is the operation to be performed on the attribute and its value):

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

Where command is the sub-command to perform on an object. The commands are listed in the table below.

The object of the command can be explicitly named, as in"

`qmgr -c "print queue <queue name>"`

or can be specified before using the command, by making the object(s) active, for example:

`qmgr -c "active Vnode1"`

Only vnodes and queues can be created or deleted using qmgr.

You can specify the default server in a command by using "@default" instead of @<server name>. If you don't name a specific object, all objects of that type at the server will be affected.

For example, to print out all of the queue information for the default server:

`qmgr -c "print queue @default"`

Under Windows, use double quotes when specifying arguments to PBS commands, including `qmgr`.

Command	Explanation
`active`	Sets the objects that will be operated on in following commands. These objects remain active until the active command is used. Disregarded when an object is specified in a qmgr command.
`create`	Creates a new object; applies to queues and vnodes.
`delete`	Destroys an existing object; applies to queues and vnodes.
`help`	Prints command-specific help and usage information
`list`	Lists the current attributes and associated values of the object.
`print`	Prints settable queue and Server attributes in a format that will be usable as input to the `qmgr` command.
`set`	Defines or alters attribute values of the object.
`unset`	Clears the value of the attributes of the object. Note: this form does not 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: `[name][@server][,name[@server]...]` 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 affected.

Variable	qmgr Variable/Syntax Description
attr	Specifies the name of an attribute of the object which is to be set or modified. The attributes of objects are described 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: 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. Used to append a string to a string array, for example "s s managers+=<manager name>"
-=	Decrease the value of the attribute by the amount specified. Used to remove a string from a string array, for example "s s managers-=<manager name>"
value	The value to assign to an attribute. If value includes white space, commas, square brackets or other special characters, such as "#", the value string must be enclosed in quote marks (" ").

A few examples of the qmgr command follow. Commands can be abbreviated. The underlined letters are there to show which abbreviations can be used in place of complete words.

```
qmgr
Qmgr: create node mars
Qmgr: set node mars resources_available.ncpus=2
Qmgr: create node venus
Qmgr: set node mars resources_available.inner = true
Qmgr: set node mars resources_available.haslife= true
Qmgr: delete node mars
Qmgr: d n venus
```

7.1.1 qmgr Help System

The qmgr built-in help function, invoked using the "help" sub-command, is illustrated by the next example which shows that requesting usage information on qmgr's set command produces 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
```

7.2 Default Configuration

Server management consists of configuring the Server attributes, defining vnodes, 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 complex. (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 attributes 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 resources_default.ncpus = 1
set server scheduler_iteration = 600
set server resv_enable = True
set server node_fail_requeue = 310
set server max_array_size = 10000
set server default_chunk.ncpus=1
```

7.2.1 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 vnodes, modifying existing objects, and changing attributes that affect policy). The "Operator" level is more restrictive. Operators cannot create new objects nor modify any attribute that changes scheduling policy. See "operators" on page 189. 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 vnodes, and set/alter/unset the following attributes:

Table 2: Attributes Requiring Manager Privilege to Set or Alter

Server	Queue	Vnode
acl_hosts	alt_route	comment
acl_host_enable	from_route_queue	Mom
acl_resv_groups	require_cred	no_multinode_jobs

Table 2: Attributes Requiring Manager Privilege to Set or Alter

Server	Queue	Vnode
acl_resv_group_enable	require_cred_enable	pnames
acl_resv_hosts	route_destinations	queue
acl_resv_host_enable		resv_enable
acl_resv_users		
acl_resv_user_enable		
acl_roots		
acl_users		
acl_user_enable		
default_node		
flatuid		
mail_from		
managers		
operators		
query_other_jobs		
require_cred		
require_cred_enable		
resv_enable		

For details on setting these levels of privilege, see the `managers` and `operators` Server attributes, discussed in “Server Configuration Attributes” on page 182; for security-related aspects of PBS privilege, see section 11.7.7 “External Security” on page 427.)

7.3 The Server’s Nodes File

The server creates a file of the nodes managed by PBS. This nodes file is written only by the Server. On startup each MOM sends a time-stamped list of her known vnodes to the Server. The Server updates its information based on that message. If the time stamp on

the vnode list is newer than what the Server recorded before in the nodes file, the Server will create any vnodes which were not already defined. If the time stamp in the MOM's message is not newer, then the Server will not create any missing vnodes and will log an error for any vnodes reported by MOM but not already known.

Whenever new vnodes are created, the Server sends a message to each MOM with the list of MOMs and each vnode managed by the MOMs. The Server will only delete vnodes when they are explicitly deleted via qmgr.

This is different from the nodes file created for each job. See section 11.9.1 "The PBS_NODEFILE" on page 432.

7.4 Hard and Soft Limits

Hard limits cannot be exceeded. Soft limits can be exceeded, but make the user's jobs eligible for preemption. Hard and soft limits can be set for the number of jobs a user can run, or usage of a particular resource. Hard and soft limits can also be set for a group, both for number of jobs running and amount of resources used. Soft limits are only used with preemption.

Example of setting user run limits:

```
s q <queue_name> max_user_run=5
s q <queue_name> max_user_run_soft=4
```

Once a user has exceeded their soft limit, their jobs are eligible for preemption. In this example, a soft limit means that when user A has reached a `max_user_run_soft of 4`, their 5th job will still run, but their 6th will not. However, all of user A's jobs are now eligible to be preempted by another user who is under their limits. If it is necessary in order to run the other user's jobs, one of user A's jobs will be preempted, then another, until user A is no longer over their soft limit.

Hard and soft resource limits work the same way. When a user exceeds a resource soft limit, that user's jobs are eligible for preemption.

Example of setting user resource limits:

```
s q <queue_name> max_user_res.mem=200gb
s q <queue_name> max_user_res_soft.mem=100gb
```

The user will not be allowed to start jobs which would exceed the hard resource limit. So if a user's first job only uses 100GB of memory, that job will run. If the user then submits a second job that requests 200GB of memory, that job will not start while the first one is running. If a job is submitted that would exceed that limit by itself, that job stays queued indefinitely.

Note that `max_user_run_soft` and `max_user_res_soft` can only be set at the server and queue levels.

For more information on soft limits, see the `pbs_server_attributes(7B)` and `pbs_queue_attributes(7B)` man pages. See also the discussion of scheduling parameters using soft limits in "Enabling Preemptive Scheduling" on page 351.

7.5 Server Configuration Attributes

This section explains all the available Server configuration attributes and gives the default values for each. These attributes are set via the `qmgr` command.

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 subdomain and domain names. See also `acl_host_enable`.
Format: "[+|-]hostname.domain[,...]"
Default value: all hosts
```
Qmgr: set server acl_hosts=*.domain.com
Qmgr: set server acl_hosts="+*.domain.com,-*"
Qmgr: set server acl_hosts+=<host-
name.domain.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 reservations 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 subdomain and domain names. Requires Manager privilege to set or alter. See also `acl_resv_enable`.
Format: "[+|-]hostname.domain[,...]"
Default value: all hosts
To put all hosts in the domain on the list of those that can request reservations:
`Qmgr: set server acl_resv_hosts=*.domain.com`
To put a host on the list of hosts not allowed to request reservations:
`Qmgr: set server acl_resv_hosts+=-host.domain.com`
To add to list of allowed hosts:
`Qmgr: set server acl_resv_hosts+=host.domain.com`
To remove from list of allowed hosts:
`Qmgr: set server acl_resv_hosts-=host.domain.com`

acl_resv_group_enable
: If true directs the Server to use the reservation group ACL `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
: If true, directs the Server to use the `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
: A single list of users allowed or denied the ability to make reservation requests of this Server. Requires Manager privilege to set or alter. See also `acl_resv_user_enable`. Manager privilege overrides user access restrictions. The order of the elements in the list is important. The list is searched, starting at the beginning, for a

match. The first match encountered in the list is accepted and terminates processing. Therefore, to allow all users except for some, the list of denied users should be put at the front of the list, followed by the set of allowed users. When usernames are added to the list, they are appended to the end of the list.
Format: "[+|-]user[@host][,...]"
Default value: all users allowed
To set list of allowed users:
```
Qmgr: set server acl_resv_users="-bob,-
tom,joe,+"
```
To add to list of allowed users:
```
Qmgr: set server acl_resv_users+=nancy@terra
```
To remove from list of allowed users:
```
Qmgr: set server acl_resv_users-=joe
```
To remove from list of disallowed users:
```
Qmgr: set server acl_resv_users-=-joe
```
To add to list of disallowed users:
```
Qmgr: set server acl_resv_users+=-mary
```

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 — A single 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_user_enable`. Manager privilege overrides user access restrictions. The order of the elements in the list is important. The list is searched, starting at the beginning, for a match. The first match encountered in the list is accepted and terminates processing. Therefore, to allow all users except for some, the list of denied users should be put at the front of the list, followed by the set of allowed users. When usernames are added to the list, they are appended to the end of the list.
Format: "[+|-]user[@host][,...]"
Default value: all users allowed
To set list of allowed users:
```
Qmgr: set server acl_users="-bob,-tom,joe,+"
```
To add to list of allowed users:
```
Qmgr: set server acl_users+=nancy@terra
```
To remove from list of allowed users:
```
Qmgr: set server acl_users-=joe
```
To add to list of disallowed users:
```
Qmgr: set server acl_users+=-mary
```

acl_roots — List of superusers who may submit to and execute jobs at this Server. If the job execution ID is zero (0), then the job owner,

root@host, must be listed in this access control list or the job is rejected. See `acl_users` for syntax.
Format: "[+|-]user[@host][,...]"
Default value: no root jobs allowed
`Qmgr: set server acl_roots=root@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_chunk — Defines default elements of chunks for all jobs on this server. All jobs will inherit default chunk elements for elements not set at submission time. Jobs moved to this server from another server will lose their old defaults and inherit these.
Format: resource specification format, e.g. "default_chunk.resource=value,default_chunk.resource=value, ..."
Qmgr: **`set server default_chunk.mem=100mb,default_chunk.ncpus=1`**

It is strongly advised not to set "default_chunk.ncpus=1" to zero. The attribute may be set to a higher value if appropriate.

default_qdel_arguments
String containing argument to qdel. Argument is "-Wsuppress_mail=<N>". Settable by the administrator. Overridden by arguments given on the command line. Default: none
Example of setting value:
`Qmgr: set server default_qdel_arguments = "-Wsuppress_email = 3"`

default_qsub_arguments
String containing any valid arguments to qsub. Settable by the administrator. Overridden by arguments given on the command line and in script directives. Default: none
Example of setting value:
`Qmgr: set server default_qsub_arguments = "-m n -r n"`

default_queue — The queue which is the target queue when a request does not specify a queue name.
Format: a queue name.
Default value: workq
`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 11.7.5 “User Authorization” on page 426 for usage and important caveats.
Format: boolean
Default value: false = disabled
`Qmgr: set server flatuid=True`

job_sort_formula — Formula for computing job priorities in the finest-granularity class given in section 9.7 “Job Priorities in PBS Professional” on page 340. If the attribute job_sort_formula is set, the scheduler will compute job priorities according to the formula. If it is unset, the scheduler computes this class of job priorities according to fairshare, if fairshare is enabled. If neither is defined, the scheduler uses job_sort_key. When the scheduler sorts jobs according to the formula, it computes a priority for each job, where that priority is the value produced by the formula. Jobs with a higher value get higher priority. To set the job_sort_formula attribute, use the qmgr command:

Qmgr> s s job_sort_formula = "<formula>"

The formula can be made up of any number of expressions, where expressions contain terms which are added, subtracted or multiplied. You cannot use division. Multiplication takes precedence over addition or subtraction. You cannot use two operators in a row. For example, "A+-B" is disallowed.

Terms can be:
Constants expressed as NUM or NUM.NUM: [0-9]+'.'[0-9]+
The following attribute values:
queue_priority: value of priority attribute for queue in which job resides
job_priority: value of the job's priority attribute
fair_share_perc: percentage of fairshare tree for this job's entity
The following resources: (the amount requested, not used)
ncpus
mem
walltime
cput
Custom numeric job-wide resources: these must be

alphanumeric with a leading alphabetic: [a-zA-Z][a-zA-Z0-9_]* This will represent the amount requested, not the amount used. They must be of type long, float, or size.

Default: unset.
Can be set by Manager or Operator.

log_events
A bit string which specifies the type of events which are logged; see also section 11.17 "Use and Maintenance of Logfiles" on page 480.
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, as well as the address where email about important events and warnings will be sent. On Windows, must be a fully qualified mail address.
Format: string
Default value: adm
```
Qmgr: set server mail_from=boss@domain.com
```

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.domain.com
```

max_array_size
The maximum number of subjobs (separate indices) that are allowed in an array job. Format: integer. Default value:10000.

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 UNIX 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 7.4 "Hard and Soft Limits" on page 181.)
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.

This limit cannot be applied selectively to primetime or non-primetime. Use a cron script to turn this limit on and off for that.

max_group_run
max_group_run_soft

The maximum number of jobs owned by a UNIX group that are allowed to be running from this server at one time.This limit can be specified as either a *hard* or *soft* limit. (See also section 7.4 “Hard and Soft Limits” on page 181.)
Format: integer
Default value: none

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

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 7.4 “Hard and Soft Limits” on page 181.)
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 6 CPUs as a maximum that any single user may consume. The second line in the example illustrates setting a *soft* limit of 3 CPUs on the same resource.

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 7.4 “Hard and Soft Limits” on page 181.)
Format: integer
Default value: none

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

node_fail_requeue

This server attribute controls how long the server will wait

before requeueing or deleting a job when it loses contact with the primary execution host. (If the job is running on more than one execution host and the primary execution host loses contact with a non-primary execution host, the node_fail_requeue attribute does not apply. In this case the job is immediately requeued or deleted.)

See section 7.5.1 "Node Fail Requeue" on page 195.

Requires either Manager or Operator privilege to set.
Format: integer
Default value: 310 (seconds)

```
Qmgr: set server node_fail_requeue=200
```

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 9.6.12 "Node Grouping" on page 339.
Format: boolean
Default value: disabled

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

node_group_key — Specifies the resource to use for node grouping. Must be a string or string_array. Requires Manager privilege to set or alter. See also node_group_enable, and section 9.6.12 "Node Grouping" on page 339.
Format: string
Default value: disabled

```
Qmgr: set server \
   node_group_key=resource[,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.
Format: "user@host.sub.domain[,user@host.sub.domain...]"
Default value: root on the local host.

```
Qmgr: set server \
   operators+=user1@sol.domain.com
Qmgr: set server operators=user1@*.domain.com
Qmgr: set server operators=user1@*
```

pbs_license_file_location

Hostname of license server, or local pathname to the actual license file(s), which is associated with a license server. String. Set by PBS Manager. Readable by all. Default value: empty string, mean-

ing no server to contact. section 5.4.3.1 “Setting the License File Location in pbs_license_file_location” on page 88.

The ALTAIR_LM_LICENSE_FILE environment variable is set by the server to the same value as this attribute.

To set pbs_license_file_location to the hostname of the license server:

```
qmgr> set server
pbs_license_file_location=<port1>@<host1>:
<port2>@<host2>:…:<portN>@<hostN>
```

where <host1>, <host2>, …, <hostN> can be IP addresses.

To set pbs_license_file_location to a local path:

```
qmgr> set server
pbs_license_file_location=<path_to_local_l
icense_file>[[:<path_to_local_license_file
2>]:…:<path_to_local_license_fileN>]]
```

To unset pbs_license_file_location:

```
Qmgr> unset server \
    pbs_license_file_location
```

pbs_license_linger_time

The number of seconds to keep an unused CPU license, when the number of licenses is above the value given by pbs_license_min. Time. Set by PBS Manager. Readable by all. Default: 3600 seconds. See section 5.4.3.4 “Setting pbs_license_linger_time” on page 92.

To set pbs_license_linger_time:

```
Qmgr> set server \
    pbs_license_linger_time=<Z>
```

To unset pbs_license_linger_time:

```
Qmgr> unset server \
    pbs_license_linger_time
```

pbs_license_max

Maximum number of licenses to be checked out at any time, i.e maximum # of CPU licenses to keep in the PBS local license pool. Sets a cap on the number of CPUs that can be licensed at one time. Long. Set by PBS Manager. Readable by all. Default: maximum value for an integer. section 5.4.3.3 "Setting pbs_license_max" on page 91.

To set `pbs_license_max`:

```
qmgr> set server pbs_license_max=<Y>
```

To unset `pbs_license_max`:

```
Qmgr> unset server pbs_license_max
```

pbs_license_min

Minimum number of CPUs to permanently keep licensed, i.e. the minimum # of CPU licenses to keep in the PBS local license pool. This is the minimum number of licenses to keep checked out. Long. Set by PBS Manager. Readable by all. Default: zero. section 5.4.3.2 "Setting pbs_license_min" on page 90.

This is for specifying the minimum # of CPU licenses that must be checked-out at any given time. That is, The default value is 0.

To set `pbs_license_min`:

```
Qmgr> set server pbs_license_min=<X>
```

To unset `pbs_license_min`:

```
Qmgr> unset server pbs_license_min)
```

query_other_jobs

The setting of this attribute 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: true (users may query or select jobs owned by other users)
```
Qmgr: set server query_other_jobs=false
```

resources_available

List of resources and amounts available to jobs on 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. The job inherits this list when there is no queue default. The values for `resources_default` are not derived from any other values; they are either set or not set. See also section 7.10 “Resource Defaults” on page 231.
Format: “resources_default.resource_name=value[,...]
Default value: for ncpus, the default value is 1
```
Qmgr: set server resources_default.mem=8mb
Qmgr: set server resources_default.ncpus=1
Qmgr: s s
resources_default.place="pack:shared"
```

resources_max — Maximum amount of each resource which can be requested by a single job on this Server if there is not a `resources_max` valued defined for the queue in which the job resides. See section 7.10 “Resource Defaults” on page 231.
Format: “resources_max.resource_name=value[,...]
Default value: infinite usage
```
Qmgr: set server resources_max.mem=1gb
Qmgr: set server resources_max.ncpus=32
```

resv_enable — This attribute 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 attribute is set True the Server will accept, for the Scheduler’s subsequent consideration, any reservation submission not otherwise rejected due 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
```

rpp_highwater
: The maximum number of RPP packets that can be in transit at any time. Acceptable values: Greater than or equal to one. Integer. Default: 64. Settable by Manager. Visible to all.
`Qmgr: set server rpp_highwater=100`

rpp_retry
: The maximum number of times the RPP network library will try to send a UDP packet again before giving up. The number of retries is added to the original try , so if rpp_retry is set to 2, the total number of tries will be 3. Integer. Acceptable values: Greater than or equal to zero. Default: 10. Settable by Manager. Visible to all.
`Qmgr: set server rpp_retry=12`

scheduler_iteration
: The time, in seconds, between iterations of attempts by the Scheduler 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 attribute to `true` results in an immediate call to the Scheduler. The PBS installation script sets `scheduling` to True. However, a call to `pbs_server -t create` sets `scheduling` to false.
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.
Qmgr: **set server scheduling=true**

single_signon_password_enable
: If enabled, this option allows users to specify their passwords only once, and PBS will remember them for future job executions. An unset value is treated as `false`. See discussion of use, and caveats, in section section 7.15 "Password Management for Windows" on page 240.
The feature can be enabled (set to True) only if no jobs exist, or if all jobs are of type "p" hold (bad password).
Format: boolean. It can be disabled only if there are no jobs currently in the system.
Default: false (UNIX), true (Windows)

```
Qmgr: set server single_signon_password_enable=true
```

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

FLicenses — Shows the number of floating PBS licenses currently available for allocation to unlicensed CPUs. One license is required for each virtual CPU. The scheduler uses this is the attribute to determine the number of licenses available.

license_count — Count of available licenses. Snapshot taken every 5 minutes. `license_count`= Avail_Global:<X> Avail_Local:<Y> Used:<Z> High_Use:<W>

Avail_Global is the number of PBS CPU licenses still kept by the Altair License Server (checked-in).

Avail_Local is the number of PBS CPU licenses in the internal PBS license pool (checked-out).

Used is the number of PBS CPU licenses currently in use.

High_Use is the highest number of CPU licenses checked-out and used at any given time while the current instance of the PBS server is running.

“Avail_Global” + “Avail_Local” + “Used” is the total number of CPU licenses configured for the PBS complex.

Integer. Set by Server. Readable by all. Default: zero.

pbs_version — The release version number of the Server.

resources_assigned — The total amount of certain resources allocated to running jobs. The resources allocated to a job from vnodes will not be released until certain allocated resources such as cpusets have been freed by all MOMs running the job.

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

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 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 shut down when the last currently running job completes.

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

7.5.1 Node Fail Requeue

This server attribute controls how long the server will wait before requeueing or deleting a job when it loses contact with the primary execution host. (If the job is running on more than one execution host and the primary execution host loses contact with a non-primary execution host, the node_fail_requeue attribute does not apply. In this case the job is immediately requeued or deleted.)

Whether a job is requeued or deleted is controlled by its `rerunnable` attribute. If a job's `rerunnable` attribute is set to "y", then the job is requeued. If the job's `rerunnable` attribute is set to "n", the job is deleted. See the "`-r y|n`" option to the `qsub` command in the **PBS Professional User's Guide**.) If a job is deleted, mail is sent to the owner of the job.

The server waits for the specified number of seconds, then attempts to contact the primary execution host, then kills and requeues the job if it cannot contact the host. If the value is zero or is unset, the job is neither killed nor requeued, but allowed to continue running. If the value is negative, it is treated as if it were set to 1 second.

This attribute's value is the delay between the time the server determines that the primary execution host cannot be contacted and the time it requeues the job, and does not include the time it takes to determine that the host is out of contact. When the server loses contact with an execution host, all jobs for which this is the primary execution host are requeued or killed at the same time.

When a job is thus requeued, it retains its original place in its original queue with its former priority. This usually means that it is the next job to be started. Exceptions are when another higher-priority job was submitted after the requeued job started, or when this job's owner is over their fairshare limit.

The number of seconds selected should be long enough to exceed any transient non-vnode failures, but short enough to requeue the job in a timely fashion.

Once a job is requeued or aborted, the resources allocated to the job cannot be made available until they are actually (a) freed or (b) made shareable to other jobs.

Manager or Operator privilege is required to set this attribute.
Format: integer
Default value: 310 (seconds)
Qmgr: **set server node_fail_requeue=200**

7.6 Queues Within PBS Professional

Once you have the Server attributes 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.

7.6.1 Execution and Routing 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. 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 7.6.4 "Attributes of Execution Queues Only" on page 203
section 7.6.5 "Attributes for Route Queues Only" on page 204
section 7.12 "Selective Routing of Jobs into Queues" on page 235
section 7.16.6 "Failover and Route Queues" on page 254
section 13.4 "Complex Multi-level Route Queues" on page 526.

7.6.2 Creating Queues

To create an execution queue:

```
#
# Create and define queue exec_queue
#
qmgr
Qmgr:
create queue exec_queue
set queue exec_queue queue_type = Execution
set queue exec_queue enabled = true
set queue exec_queue started = true
```

Now we will create a routing queue, which will send jobs to our execution queue:

```
qmgr
Qmgr:
create queue routing_queue
set queue routing_queue queue_type = Route
```

```
set queue routing_queue route_destinations = exec_queue
```

Note:
1. Destination queues must be created before being used as the routing queue's route_destinations.
2. Routing queue's route_destinations must be set before enabling and starting the routing queue.

```
set queue routing_queue enabled = true
set queue routing_queue started = true
```

Note:
If we want the destination queue to accept jobs only from a routing queue, we set its from_route_only attribute to true:

```
set queue exec_queue from_route_only = True
```

7.6.3 Queue Configuration Attributes

Queue configuration attributes 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 attributes require the user to have full Administrator privilege before they can be modified. The following attributes apply to both queue types:

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: unset
```
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: unset
```
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 — A single list of users allowed or denied the ability to enqueue jobs in this queue. Requires Manager privilege to set or alter. See also `acl_user_enable`. Manager privilege overrides user access restrictions. The order of the elements in the list is important. The list is searched, starting at the beginning, for a match. The first match encountered in the list is accepted and terminates processing. Therefore, to allow all users except for some, the list of denied users should be put at the front of the list, followed by the set of allowed users. When usernames are added to the list, they are appended to the end of the list.
Format: "[+|-]user[@host][,...]"
Default value: all users allowed
To set list of allowed users:
```
Qmgr: set queue QNAME acl_users="-bob,-tom,joe,+"
```
To add to list of allowed users:
```
Qmgr: set queue QNAME acl_users+=nancy@terra
```
To remove from list of allowed users:
```
Qmgr: set queue QNAME acl_users-=joe
```

To add to list of disallowed users:
```
Qmgr: set queue QNAME acl_users+=-mary
```

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_array_size — The maximum number of subjobs that a job array in that queue can have. Job arrays with more than this number will be rejected at qsub time.
Format: integer.
Default: 10000.
```
Qmgr: set queue QNAME max_array_size = 5000
```

max_group_res
max_group_res_soft — The maximum amount of the specified resource that all members of the same UNIX 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 7.4 “Hard and Soft Limits” on page 181.)
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_group_run
max_group_run_soft — The maximum number of jobs owned by a UNIX 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 7.4

"Hard and Soft Limits" on page 181.)
Format: integer
Default value: none

```
Qmgr: set queue QUEUE max_group_run=10
Qmgr: set queue QUEUE max_group_run_soft=7
```

max_queuable — The maximum number of jobs allowed to reside in the queue at any given time. Once this limit is reached, 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 in submitting to this 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 7.4 "Hard and Soft Limits" on page 181.)
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
```

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 from this queue. This limit can be specified as either a *hard* or *soft* limit. (See also section 7.4 "Hard and Soft Limits" on page 181.)
Format: integer
Default value: none

```
Qmgr: set queue QUEUE max_user_run=6
Qmgr: set queue QUEUE max_user_run_soft=3
```

node_group_key — Specifies the resource to use for node grouping. Must be a string or string_array. Overrides server's `node_group_key`.
Format: string. Default value: disabled. Example:

```
Qmgr: set queue Q \
   node_group_key=RESOURCE[,RESOURCE ...]
```

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
Default value: 0
```
Qmgr: set queue QNAME priority=123
```

queue_type — The type of the queue: execution or route. This attribute 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
```

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 the queue’s `resources_default` is not set, the default limit for a job is determined by the first of the following attributes which is set: Server’s `resources_default`, queue’s `resources_max`, Server’s `resources_max`. An unset resource is viewed as having a value of zero. See also section 7.10 “Resource Defaults” on page 231.
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
Qmgr: set queue QNAME
resources_default.place="pack:shared"
```

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 7.10 “Resource Defaults” on page 231.
Format: “resources_max.*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 7.10 “Resource Defaults” on page 231.
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 — When true, jobs may be scheduled for execution from this queue. When false, the queue is considered *stopped* and jobs will not be executed from this queue.
Format: boolean
Default value: unset
```
Qmgr: set queue QNAME started=true
```

7.6.4 Attributes of 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
```

default_chunk — Defines default elements of chunks for all jobs on this queue. All jobs will inherit default chunk elements for elements not set at submission time, if server and queue resources_default do not apply. See the pbs_resources(7B) man page. Jobs moved to this queue from another queue will lose their old defaults and inherit these.
Format: resource specification format, e.g. "default_chunk.resource=value,default_chunk.resource=value, ..."
Qmgr: **`set queue QNAME default_chunk.mem=100mb`**

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
```

7.6.5 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 7.12 "Selective Routing of Jobs into Queues" on page 235.
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 net-

work 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`

7.6.6 Read-only Attributes of Queues

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

hasnodes	If true, indicates that the queue has vnodes 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	Amount of resources allocated to jobs running in this queue.

7.6.7 Queue Status

When you use the qstat command to find the status of a queue, it is reported in the "State" field. The field will show two letters. One is either E (enabled) or D (disabled.) The other is R (running, same as started) or S (stopped.)

7.7 Vnodes: Virtual Nodes

A virtual node, or *vnode*, is an abstract object representing a set of resources which form a usable part of a machine. This could be an entire host, or a nodeboard or a blade. A single host can be made up of multiple vnodes. Each vnode can be managed and scheduled independently. PBS views hosts as being composed of one or more vnodes. Commands such as

```
Qmgr: create node VNODE
```

have not changed, and operate on vnodes despite referring to nodes. However, only the natural vnode on a multi-vnode host should be created this way. See the `pbs_node_attributes(7B)` man page.

On Windows, there is a one-to-one correspondence between MOMs and vnodes.

7.7.1 Where Jobs Run

Where jobs will be run is determined by an interaction between the Scheduler and the Server. This interaction is affected by the list of hosts known to the server, and the system configuration onto which you are deploying PBS. Without this list of vnodes, 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. By default, when the Scheduler seeks a vnode meeting the requirements of a job, it will select the first available vnode in the list that meets those requirements. Thus the order of vnodes in the nodes file has a direct impact on vnode selection for jobs. (This default behavior can be overridden by the various vnode-sorting options available in the Scheduler. For details, see the discussion of `node_sort_key` in section 9.3 “Scheduler Configuration Parameters” on page 315.)

Use the `qmgr` command to create or delete vnodes. See section 7.7.3 “Creating or Modifying Vnodes” on page 207. Only use the `qmgr` command to create or delete vnodes.

Vnodes can have attributes and resources associated with them. Attributes are name=value pairs, and resources use name.resource=value pairs. A user’s job can specify that the vnode(s) used for the job have a certain set of attributes or resources. See section 7.9 “PBS Resources” on page 217.

7.7.2 Natural Vnodes

A natural vnode does not correspond to any actual hardware. It is used to define any placement set information that is invariant for a given host. See section 9.6 “Placement Sets and Task Placement” on page 326. It is defined as follows:

`name`	The name of the natural vnode is, by convention, the MOM contact name, which is usually the hostname. The MOM contact name is the vnode's Mom attribute. See the pbs_node_attributes(7B) man page.
`pnames attribute`	An attribute, "pnames", with value set to the list of resource names that define the placement sets' types for this machine.

`sharing attribute`	An attribute, "sharing" is set to the value "ignore_excl".

The order of the pnames attribute follows placement set organization. If name X appears to the left of name Y in this attribute's value, an entity of type X may be assumed to be smaller (that is, be capable of containing fewer vnodes) than one of type Y. No such guarantee is made for specific instances of the types.

Natural vnodes must have their schedulable resources (ncpus, mem, vmem) set to zero to prevent them from having jobs scheduled on them.

Here is an example of the vnode definition for a natural vnode:

```
altix03:  pnames = cbrick, router
altix03:  sharing = ignore_excl
altix03:  resources_available.ncpus = 0
altix03:  resources_available.mem = 0
altix03:  resources_available.vmem = 0
```

On a multi-vnoded machine which has a natural vnode, anything set in the `mom_resources` line in `PBS_HOME/sched_priv/sched_config` is shared by all of that machine's vnodes.

7.7.3 Creating or Modifying Vnodes

After `pbs_server` is started, the vnode list may be created via the `qmgr` command. First start up pbs_mom, then use qmgr to add the vnode. For example, to add a new vnode, use the "create" sub-command of `qmgr`:

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

where the attributes and their associated possible values are shown in the table below. Vnode attributes cannot be used as vnode names. On a multi-vnode system, only the natural vnode should be created this way. Vnode attributes are listed in section 7.8 "Vnode Configuration Attributes" on page 210.

Important: All comma-separated attribute-value strings must be enclosed in quotes.

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

```
qmgr
Qmgr: create node mars resources_available.ncpus=2
Qmgr: create node venus
```

Modify vnodes: Once a vnode has been created, its attributes and/or boolean resources can be modified using the following `qmgr` syntax:

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

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

```
qmgr
Qmgr: set node mars resources_available.inner=true
Qmgr: set node mars resources_available.haslife=true
```

Delete vnodes: 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
```

7.7.3.1 Caveats

Most of a vnode's attributes may be set using qmgr. However, some **must** be set on the individual execution host in local vnode definition files, NOT by using qmgr. Those that must be set on the execution host this way are

sharing
ncpus
mem
vmem

An example of the way to do this (in this case, changing the "sharing" attribute for a vnode named V10) uses the script "change_sharing". See section 8.2.1 "Creation of Site-defined MOM Configuration Files" on page 259.

```
# cat change_sharing
$configversion 2
V10:  sharing = ignore_excl
# . /etc/pbs.conf
# $PBS_EXEC/sbin/pbs_mom -s insert ignore_excl
```

```
        change_sharing
# pkill -HUP pbs_mom
```

Do **not** set sharing, ncpus, mem, or vmem on a vnode via qmgr.

It is not a good idea to try to use qmgr to create the vnodes for an Altix, other than the natural vnode. You do need to create the natural vnode via qmgr. It is possible to use qmgr to create a vnode with any name. The "[x]" naming does not imply any special significance; it just an internal convention for naming vnodes on an Altix. The fact that you can create a vnode with a weird name does not mean however that the MOM on the host knows about that vnode. If the MOM does not know about the vnode, the vnode will be considered "stale" and not usable. Be default, MOM only knows about the natural vnode, the one whose name is the same as the host.

7.7.4 Virtual Nodes on Blue Gene

On the IBM Blue Gene, each vnode is a basic allocation unit, where a set of those units makes up a partition. A vnode could be 1 base partition or 1/16 of a base partition (nodecard). For example, one partition can be made up of 1 vnode containing 1024 CPUs (1 BP), and a smaller partition can be made up of 4 vnodes containing 64 CPUs each (1 nodecard).

Each vnode has a unique name prefixed by the local hostname and enclosed in brackets. If the vnode is representing a base partition, then it is named after the base partition ID (i.e. bluegene[BP_ID]); if the vnode is representing a nodecard, then it is named "bluegene[<BP_ID>#<QUARTER_CARD_NO>#<NODECARD_ID>]". For example, "bluegene[R001]" is a vnode representing the midplane "R001", and "bluegene[R101#3#J216]" is a vnode representing nodecard "J216" found in quadrant 3 of the base partition "R001".

Each vnode reports the number of cpus and the amount of memory available (this will not be explicitly requested by users).

Each vnode has its "sharing" attribute set to "force_excl".

Each vnode will also have its "resource_available.arch" set to "bluegene".

Each vnode has a list of Blue Gene partitions to which the vnode's compute nodes are assigned. A new resource keyword of string type called "partition" is used to enumerate the partitions.

7.7.4.1 The Natural Vnode on Blue Gene

The Blue Gene natural vnode must have values of zero for resources_available for ncpus, mem, vmem and ncpus, e.g.

```
resources_available.ncpus=0
resources_available.mem=0
resources_available.vmem=0
```

7.8 Vnode Configuration Attributes

A vnode has the following configuration attributes:

comment — General comment; can be set by a PBS Manager. If this attribute is not explicitly set, the PBS Server will use it to display vnode status, specifically why the vnode 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"`

lictype — **Deprecated**. No longer used.

max_running — The maximum number of jobs allowed to be run on this vnode 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 vnode 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 vnode at one time.
Format: integer
`Qmgr: set node MyNode max_group_run=8`

Mom — Hostname of host on which MOM daemon will run. Can be explicitly set only via `qmgr`, and only at vnode creation. Defaults to value of vnode resource (vnode name.)

no_multinode_jobs — If this attribute is set true, jobs requesting more than one vnode

will not be run on this vnode. This attribute 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 vnode.
Format: boolean
```
Qmgr: set node MyNode no_multinode_jobs=true
```

Port — Port number on which MOM will listen. Integer. Can be explicitly set only via `qmgr`, and only at vnode creation. On multi-vnode machine, can only be set on natural vnode.

priority — The priority of this vnode against other vnodes of the same type on this Server. (A larger value is higher priority than a smaller value.) May be used in conjunction with `node_sort_key`.
Format: integer
Default value: 0
```
Qmgr: set node MyNode priority=123
```

queue — Name of an execution queue (if any) associated with a vnode. If this attribute is set, only jobs from the named queue will be run on the associated vnode, and jobs in that queue will only be run on the vnode or vnodes associated with that queue. Note: a vnode can be associated with at most one queue by this method. Note that if a vnode is associated with a queue, it will no longer be considered for advance reservations, nor for node grouping.
Format: queue specification
```
Qmgr: set node MyNode queue=MyQueue
```

resources_available — List of resources available on vnode. Any valid PBS resources can be specified.
Format: resource list
```
Qmgr:set node MyNode resources_available.ncpus=2
Qmgr:set node MyNode resources_available.RES=xyz
```

resv_enable — Whether or not the vnode can be used for advance reservation requests. The vnode is available for advance reservations, except when it is configured for cycle harvesting. Any reservations already assigned to this vnode will not be removed if

this attribute is subsequently set to false. Requires manager privilege to set or alter. Format: True/False. Default value: True. Default value is False if the vnode is marked for cycle harvesting.

sharing — Defines whether more than one job at a time can use this vnode's resources. Either a) the vnode is allocated exclusively to one job, or b) the vnode's unused resources are available to other jobs.
Allowable values: default_shared | default_excl | ignore_excl | force_excl
This attribute can be set via the vnode definition entries in MOM's config file.
Example: `vnodename: sharing=force_excl`
Default value: default_shared.

A vnode's behavior is determined by a combination of its sharing attribute and a job's placement directive. The behavior is defined as follows:

Table 3: Vnode Sharing by Attribute and Placement

vnode's sharing attribute	Place Statement Contents		
	unset	place=shared	place=excl
unset	shared	shared	excl
sharing=default_shared	shared	shared	excl
sharing=default_excl	excl	shared	excl
sharing=ignore_excl	shared	shared	shared
sharing=force_excl	excl	excl	excl

The administrator may want to require that each vnode in the system be used exclusively by whatever job is running on it. The administrator should then set "sharing=force_excl". This will override any job "place=shared" setting. Similarly, "sharing=ignore_excl" will override any job "place=excl" setting.

If there is a multi-vnoded system which has a pool of applica-

tion licenses available for use, these will be associated with a resource defined on the natural vnode (i.e., the vnode whose name is the same as the host). The natural vnode's sharing attribute should be set to "ignore_excl". The pool of licenses will be shared among different jobs. Note that this case does not override a job's "excl" setting. The individual license obtained by the job will be held exclusively. See section 10.7 "Application Licenses" on page 388.

state Shows or sets the state of the vnode. Format: string.
`Qmgr: set node MyNode state=offline`

Table 4: Node States

State	Set By	Description
free	Server Manager Operator	Node is up and has available CPU(s). Server will mark a vnode "free" on first successful ping after vnode was "down". Manager/Operator should only use this to clear an "offline" state.
offline	Manager Operator	Node is not usable. Jobs running on this vnode will continue to run. Used by Manager/Operator to mark a vnode not to be used for jobs.
down	Server	Node is not usable. Existing communication lost between Server and MOM.
job-busy	Server	Node is up and all CPUs are allocated to jobs.
job-exclusive	Server	Node is up and has been allocated exclusively to a single job.

Table 4: Node States

State	Set By	Description
busy	Server	Node is up and has load average greater than `$max_load`. When the loadave is above max_load, that node is marked "busy". The scheduler won't place jobs on a node marked "busy". When the loadave drops below ideal_load, the "busy" mark is removed. Consult your OS documentation to determine values that make sense.
stale	Server	MOM managing vnode is not reporting any information. Server can still communicate with MOM.
state-unknown, down	Server	Node is not usable. Since Server's latest start, no communication with this vnode. May be network or hardware problem, or no MOM on vnode.

A vnode has the following read-only attributes:

pcpus Shows the number of physical CPUs on the vnode. On a multi-vnoded machine, this resource will appear only on the first vnode.

license **Deprecated**. Indicates the vnode "license state" as a single character, according to the following table:

`u`	No jobs are running on this node
`f`	At least one job has been allocated to this vnode

ntype No longer used to distinguish between vnode uses. The "time-shared" and "cluster" node types are **deprecated**.

pbs_version	PBS version for the vnode's MOM . Available only to Manager/Operator.
resources_assigned	List of resources in use on vnode. Format: resource list
reservations	List of reservations pending on the vnode. Format: reservation specification
jobs	List of jobs executing on the vnode. A job is listed in the vnode's `jobs` attribute until the vnode's resources allocated to the job are freed.

If the following vnode resources 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
```

7.8.1 Node Comments

Nodes have a "comment" attribute which can be used to display information about that vnode. If the comment attribute has not been explicitly set by the PBS Manager and the vnode is down, it will be used by the PBS Server to display the reason the vnode was marked down. If the Manager has explicitly set the attribute, 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"
```

Once set, vnode comments can be viewed via `pbsnodes`, `xpbsmon` (vnode detail page), and `qmgr`. (For details see "The pbsnodes Command" on page 498 and "The xpbsmon GUI Command" on page 518.)

7.8.2 Associating Vnodes with Multiple Queues

You can use resources to associate a vnode with more than one queue. The scheduler will use the resource for scheduling just as it does with any resource. In order to map a vnode to more than one queue, you must define a custom resource. Define the custom resource and add it to the scheduler's sched_priv/sched_config file as follows.

Add to $PBS_HOME/server_priv/resourcedef:

```
Qlist type=string_array flag=h
```

Change $PBS_HOME/sched_priv/sched_config to add "Qlist", e.g.,

```
resources: "ncpus, mem, arch, host, vnode, Qlist"
```

Now, as an example, assume you have 3 queues: MathQ, PhysicsQ, and ChemQ, and you have 4 vnodes: vn[1], vn[2], vn[3], vn[4]. To achieve the following mapping:

```
MathQ -->  vn[1], vn[2]
PhysicsQ -->vn[2], vn[3], vn[4]
ChemQ --> vn[1], vn[2], vn[3]
```

Which is the same as:

```
vn[1] <-- MathQ, ChemQ
vn[2] <-- MathQ, PhysicsQ, ChemQ
vn[3] <-- PhysicsQ, ChemQ
vn[4] <-- PhysicsQ
```

Set the following via qmgr:

Add queue to vnode mappings:

```
Qmgr: s n vn[1] resources_available.Qlist="MathQ,ChemQ"
Qmgr: s n vn[2] resources_available.Qlist="MathQ,PhysicsQ,ChemQ"
Qmgr: s n vn[3] resources_available.Qlist="PhysicsQ,ChemQ"
Qmgr: s n vn[4] resources_available.Qlist="PhysicsQ"
```

Force jobs to request the correct Q values:

```
Qmgr: s q MathQ resources_default.Qlist=MathQ
Qmgr: s q MathQ resources_min.Qlist=MathQ
Qmgr: s q MathQ resources_max.Qlist=MathQ
Qmgr: s q MathQ default_chunk.Qlist=MathQ
```

```
Qmgr: s q PhysicsQ resources_default.Qlist=PhysicsQ
Qmgr: s q PhysicsQ resources_min.Qlist=PhysicsQ
Qmgr: s q PhysicsQ resources_max.Qlist=PhysicsQ
Qmgr: s q PhysicsQ default_chunk.Qlist=PhysicsQ

Qmgr: s q ChemQ resources_default.Qlist=ChemQ
Qmgr: s q ChemQ resources_min.Qlist=ChemQ
Qmgr: s q ChemQ resources_max.Qlist=ChemQ
Qmgr: s q ChemQ default_chunk.Qlist=ChemQ
```

If you use the vnode's queue attribute, the vnode can be associated only with the queue named in the attribute.

7.9 PBS Resources

Resources can be available on the server and on vnodes. Jobs can request resources. Resources are allocated to jobs, and some resources such as memory are consumed by jobs. The scheduler matches requested resources with available resources, according to rules defined by the administrator. PBS can enforce limits on resource usage by jobs.

PBS provides built-in resources, and in addition, allows the administrator to define custom resources. The administrator can specify which resources are available on a given vnode, as well as at the queue or server level (e.g. floating licenses.) Vnodes can share resources. The administrator can also specify default arguments for `qsub`. These can include resources. See the `qsub(1B)` man page and "Server Configuration Attributes" on page 182.

Resources made available by defining them via resources_available at the queue or server level are only used as job-wide resources. These resources (e.g. `walltime, server_dyn_res`) are requested using -l RESOURCE=VALUE. Resources made available at the host (vnode) level are only used as chunk resources, and can only be requested within chunks using -l select=RESOURCE=VALUE. Resources such as mem and ncpus can only be used at the vnode level in a new-style resource request.

Resources are allocated to jobs both by explicitly requesting them and by applying specified defaults. Jobs explicitly request resources either at the vnode level in chunks defined in a selection statement, or in job-wide resource requests. See the PBS Profes-

sional User's Guide and the pbs_resources(7B) manual page, "PBS Resources" on page 217 and section 4.3.1 "Rules for Submitting Jobs" on page 31 in the **PBS Professional User's Guide**.

Boolean resources default to "False".

A "consumable" resource is one that is reduced by being used, for example, ncpus, licenses, or mem. A "non-consumable" resource is not reduced through use, for example, walltime or a boolean resource.

Resources are tracked in server, queue, vnode and job attributes. Servers, queues and vnodes have two attributes, resources_available.RESOURCE and resources_assigned.RESOURCE. The resources_available.RESOURCE attribute tracks the total amount of the resource available at that server, queue or vnode, without regard to how much is in use. The resources_assigned.RESOURCE attribute tracks how much of that resource has been assigned to jobs at that server, queue or vnode. Jobs have an attribute called resources_used.RESOURCE which tracks the amount of that resource used by that job.

7.9.1 Job Resource Limits

Jobs are assigned limits on the amount of resources they can use. These limits apply to how much the job can use on each vnode (per-chunk limit) and to how much the whole job can use (job-wide limit). Limits are derived from both requested resources and applied default resources.

Each chunk's per-chunk limits determine how much of any resource can be used in that chunk. Per-chunk resource usage limits are the amount of per-chunk resources requested, both from explicit requests and from defaults.

Job resource limits set a limit for per-job resource usage. Job resource limits are derived in this order from:

explicitly requested job-wide resources (e.g. -l resource=value)
the select specification (e.g. -l select =...)
the queue's default_resources.RES
the server's default_resources.RES
the queue's resources_max.RES
the server's resources_max.RES

The server's default_chunk.RES does **not** affect job-wide limits.

The resources requested for chunks in the select specification are summed, and this sum is used for a job-wide limit. Job resource limits from sums of all chunks override those from job-wide defaults and resource requests.

Various limit checks are applied to jobs. If a job's job resource limit exceeds queue or server restrictions, it will not be put in the queue or accepted by the server. If, while running, a job exceeds its limit for a consumable or time-based resource, it will be terminated.

For a job, enforcement of resource limits is per-MOM, not per-vnode. So if a job requests 3 chunks each of which has 1MB of memory, and all chunks are placed on one host, the limit for that job for memory for that MOM is 3MB. Therefore one chunk can be using 2 MB and the other two using 0.5MB and the job can continue to run.

7.9.2 Unset Resources

When job resource requests are being matched with available resources, a numerical resource that is unset on a host is treated as if it were zero, but an unset resource on the server or queue is treated as if it were infinite. An unset string cannot be matched. An unset Boolean resource is treated as if it is set to "False".

The resources ompthreads, mpiprocs, and nodes are ignored for unset resource matching.

The following table shows how a resource request will or won't match an unset resource.

Table 5: Matching Requests to Unset Resources

Resource Type	Unset Resource	Matching Request Value
boolean	False	False
float	0.0	0.0
long	0	0
size	0	0
string	""	Never matches
string array	""	Never matches
time	0, 0:0, 0:0.0, 0:0:0	0, 0:0, 0:0.0, 0:0:0

To preserve backward compatibility, you can set the server's

resource_unset_infinite attribute with a list of resources that will behave as if they are infinite when they are unset. See *resource_unset_infinite* in section 9.3 “Scheduler Configuration Parameters” on page 315.

Note that jobs may be placed on different vnodes from those where they would have run in earlier versions. This is because a job’s resource request will no longer match the same resources on the server, queues and vnodes.

7.9.3 Deleting Custom Resources

If the administrator deletes a resource definition from $PBS_HOME/server_priv/resourcedef and restarts the server, any and all jobs which requested that resource will be purged from the server when it is restarted. Therefore removing any custom resource definition should be done with extreme care.

7.9.4 Vnodes and Shared Resources

Node-level resources can be “ shared” across vnodes. This means that a resource is managed by one vnode, but available for use at others. This is called an *indirect* resource. Any vnode-level dynamic resources (i.e. those listed in the PBS_HOME/sched_priv/sched_config “mom_resources” line) will be treated as “ shared” resources. The MOM manages the sharing. The resource to be shared is defined as usual on the managing vnode. The built-in resource ncpus cannot be shared. Static resources can be made indirect.

To set a static value:

```
Qmgr: s n managing_vnode resources_available.RES
   =<value>
```

To set a dynamic value, in MOM config:

```
managing_vnode:RES=<value>
managing_vnode:“RES=!path-to-command”
```

To set a “ shared” resource RES on a borrowing vnode, use either

```
Qmgr: s n borrowing_vnode resources_available.RES
   =@managing_vnode
```

or in MOM config, for static or dynamic:

```
borrowing_vnode:RES=@managing_vnode
```

Example: to make a static host-level license dyna-license on hostA indirect at vnodes hostA0 and hostA1:

```
Qmgr: set node hostA0 \
```

```
     resources_available.dyna-license=@hostA
Qmgr: set node hostA1 \
     resources_available.dyna-license=@hostA
```

For example, to set the resource `string_res` to "round" on the natural vnode of altix03 and make it indirect at altix03[0] and altix03[1]:

```
Qmgr: set node altix03 resources_available.string_res=round
Qmgr: s n altix03[0] resources_available.string_res=@altix03
Qmgr: s n altix03[1] resources_available.string_res=@altix03

pbsnodes -va

altix03
     ...
     string_res=round
     ...

altix03[0]
     ...
     string_res=@altix03
     ...

altix03[1]
     ...
     string_res=@altix03
     ...
```

If you had set the resource `string_res` individually on altix03[0] and altix03[1]:

```
Qmgr: s n altix03[0] resources_available.string_res=round
Qmgr: s n altix03[1] resources_available.string_res=square

pbsnodes -va

altix03
     ...
       <--------string_res not set on natural vnode
     ...
```

```
altix03[0]
     ...
     string_res=round
     ...

altix03[1]
     ...
     string_res=square
     ...
```

7.9.4.1 Defining Resources for the Altix

On an Altix where you are running pbs_mom.cpuset, you can manage the resources at each vnode. For dynamic host-level resources, the resource is shared across all the vnodes on the machine, and MOM manages the sharing. For static host-level resources, you can either define the resource to be shared or not. Shared resources are usually set on the natural vnode and then made indirect at any other vnodes on which you want the resource available. For resources that are not shared, you can set the value at each vnode. Note that you do not want the scheduler to try to run jobs on the natural vnode. To prevent this, make sure that the values of mem, vmem and ncpus are set to zero on the natural vnode.

If any of the following resources has been explicitly set to a non-zero value on the natural vnode, set `resources_available.ncpus`, `resources_available.mem` and `resources_available.vmem` to zero on each natural vnode:

```
Qmgr: set node <natural vnode name>  \
     resources_available.ncpus=0
Qmgr: set node <natural vnode name>  \
     resources_available.mem=0
Qmgr: set node <natural vnode name>  \
     resources_available.vmem=0
```

7.9.5 Matching Jobs to Resources

For all resources except boolean and string and string array resources, if a resource is unset (not defined) at a vnode, a resource request will behave as if that resource is zero. If a resource is unset at the server or queue level, the resource request will behave as if that resource is infinite. An unset string or string resource cannot be matched.

For boolean resources, if a resource is unset (undefined) at a server, queue, or vnode, the resource request will behave as if that resource is set to "false". It will match a resource request for that boolean with a value of "false", but not "true".

7.9.6 String Arrays: Multi-valued String resources

The resource of type string_array is a comma-delimited set of strings. Each vnode can have its resource RES be a different set of strings. A job can only request one string per resource in its resource request. The job is placed on a vnode where its requested string is one of the multiple strings set on a vnode.

Example:

Define a new resource
`"foo_arr type=string_array flag=h"`

Setting via qmgr:

```
Qmgr> set node n4 \
   resources_available.foo_arr="f1, f3, f5"
```

Vnode n4 has 3 values of `foo_arr`: f1, f3, and f5

```
Qmgr> set node n4 resources_available.foo_arr+=f7
```

Vnode n4 now has 4 values of `foo_arr`: f1, f3, f5 and f7

Submission:

```
   qsub –l select=1:ncpus=1:foo_arr=f3
```

A string array resource with one value works exactly like a string resource. A string array uses the same flags as other non-consumable resources. The default value for a job's multi-valued string resource, listed in resource_default.RES, can only be one string.

For string_array resources on a queue, `resources_min` and `resources_max` must be set to the same set of values. A job must request one of the values in the set to be allowed into the queue. For example, if we set resources_min.strarr and resources_max.strarr to "blue,red,black", jobs can request –l strarr=blue, -l strarr=red, or –l strarr=black to be allowed into the queue.

7.9.7 Resource Types

The resource values are specified using the following data types:

boolean	Boolean-valued resource. Should be defined only at the vnode level. Non-consumable. Can only be requested inside a select

statement, i.e. in a chunk. Name of resource is a string. Allowable values (case insensitive): True|T|Y|1|False|F|N|0

A boolean resource named "RESOURCE" is defined in PBS_HOME/server_priv/resourcedef by putting in a line of the form:

RESOURCE type=boolean flag=h

float — Float. Allowable values: [+-] 0-9 [[0-9] ...][.][[0-9] ...]

long — Long integer. Allowable values: 0-9 [[0-9] ...]

size — Number of bytes (default) or words. It is expressed in the form `integer[suffix]`. The suffix is a multiplier defined in the following table. The size of a word is the word size on the execution host.

b or w	bytes or words.
kb or kw	Kilo (2^{10}, 1024) bytes or words.
mb or mw	Mega (2^{20}, 1,048,576) bytes or words.
gb or gw	Giga (2^{30}, 1,073,741,824) bytes or words.
tb or tw	Tera (2^{40}, or 1024 gigabytes) bytes or words.
pb or pw	Peta (2^{50}, or 1,048,576 gigabytes) bytes or words.

string — String. Non-consumable. Allowable values: Any printable character, including the space character., except the tab or other white space and the ampersand (“&”) character. The first character must be alphanumeric or underscore. Only one of the two types of quote characters, " or ', may appear in any given value.

Values:[_a-zA-Z0-9][[-_a-zA-Z0-9 ! " # $ % ´ ( ) * + , - . / : ; < = > ? @ [\ ] ^ _ ‘ { | } ~] ...]

string_array — Comma-separated list of strings. Strings in string arrays may not contain commas. Non-consumable. Resource request will

succeed if request matches one of the values. Resource request can contain only one string.

time specifies a maximum time period the resource can be used. Time is expressed in seconds as an integer, or in the form:

```
[[hours:]minutes:]seconds[.milliseconds]
```

Different resources are available on different systems, often depending on the architecture of the computer itself. For example, on the NEC SX-8, there is no virtual memory, but there is "whole process address space". So for the NEC SX-8, `mem=vmem`. The table below lists the available resources that can be requested by PBS jobs on any system.

7.9.8 Resource Flags

FLAGS is a set of characters which indicate whether and how the Server should accumulate the requested amounts of the resource in the attribute `resources_assigned` when the job is run. This allows the server to keep track of how much of the resource has been used, and how much is available.

For example, when defining a static consumable host-level resource, such as a node-locked license, you would use the "n" and "h" flags. However, when defining a dynamic resource such as a floating license, no flag would be used.

The value of `flag` is a concatenation of one or more of the following letters:

h Indicates a host-level resource. Used alone, means that the resource is not consumable. Required for any resource that will be used inside a select statement.
Example: for a boolean resource named "green":
green type=boolean flag=h

n The amount is consumable at the host level, for all vnodes assigned to the job. Must be consumable or time-based. (Cannot be used with boolean or string resources.) The "h" flag must also be used.

f The amount is consumable at the host level for only the first vnode allocated to the job (vnode with first task.) Must be con-

sumable or time-based. (Cannot be used with boolean or string resources.) The “h” flag must also be used.

(no flags) Indicates a queue-level or server-level resource that is not consumable.

q The amount is consumable at the Queue and Server level. Must be consumable or time-based.

Table 6: When to Use Flags

Resource	Server	Queue	Host
Static, consumable	flags = q	flags = q	flags = nh or fh
Static, not consumable	no flags	no flags	flags = h
Dynamic	(server_dyn_res line in sched_config) no flags	(cannot be used)	(MOM config and mom_resources line in sched_config) flags = h

7.9.9 Built-in Resources

Table 7: Built-in Resources

Resource	Description
arch	System architecture. For use inside chunks only. One architecture can be defined for a vnode. One architecture can be requested per vnode. Allowable values and effect on job placement are site-dependent. Type: string. See "Specifying Architectures" on page 229.
cput	Amount of CPU time used by the job for all processes on all vnodes. Establishes a job resource limit. Non-consumable. Type: time.
file	Size of any single file that may be created by the job. Type: size.
host	Name of execution host. For use inside chunks only. Automatically set to the short form of the hostname in the Mom attribute. Cannot be changed. Site-dependent. Type: string.
mem	Amount of physical memory i.e. workingset allocated to the job, either job-wide or vnode-level. Consumable. Type: size.
mpiprocs	Number of MPI processes for this chunk. Defaults to 1 if ncpus > 0, 0 otherwise. For use inside chunks only. Type: integer. The number of lines in PBS_NODEFILE is the sum of the values of mpiprocs for all chunks requested by the job. For each chunk with mpiprocs=P, the host name for that chunk is written to the PBS_NODEFILE P times.
ncpus	Number of processors requested. Cannot be shared across vnodes. Consumable. Type: integer.
nice	Nice value under which the job is to be run. Host-dependent. Type: integer.
nodect	**Deprecated**. Number of chunks in resource request from selection directive, or number of vnodes requested from node specification. Otherwise defaults to value of 1. Read-only. Type: integer.

Table 7: Built-in Resources

Resource	Description
ompthreads	Number of OpenMP threads for this chunk. Defaults to ncpus if not specified. For use inside chunks only. Type: integer. For the MPI process with rank 0, the environment variables NCPUS and OMP_NUM_THREADS are set to the value of ompthreads. For other MPI processes, behavior is dependent on MPI implementation.
pcput	Amount of CPU time allocated to any single process in the job. Establishes a job resource limit. Non-consumable. Type: time.
pmem	Amount of physical memory (workingset) for use by any single process of the job. Establishes a job resource limit. Consumable. Type: size
pvmem	Amount of virtual memory for use by the job. Establishes a job resource limit. Not consumable. Type: size.
software	Site-specific software specification. For use only in job-wide resource requests. Allowable values and effect on job placement are site-dependent. Type: string.
vmem	Amount of virtual memory for use by all concurrent processes in the job. Establishes a job resource limit, or when used within a chunk, establishes a per-chunk limit. Consumable. Type: size.
vnode	Name of virtual node (vnode) on which to execute. For use inside chunks only. Site-dependent. Type: string. See the pbs_node_attributes(7B) man page.
walltime	Actual elapsed time during which the job can run. Establishes a job resource limit. Non-consumable. Type: time. Default: 5 years.

Every consumable resource such as `mem` has four associated values, each of which is used in several places in PBS:

Table 8: Values Associated with Consumable Resources

Value	Node	Queue	Server	Accounting Log	Job	Scheduler
`resources_available`	X	X	X			X

Table 8: Values Associated with Consumable Resources

Value	Node	Queue	Server	Accounting Log	Job	Scheduler
`resources_assigned`	X	X	X	X		
`resources_used`				X	X	X
`Resource_List`					X	X

The Vnode, Server, and Queue values are usually displayed via `pbsnodes` and `qmgr`; the Accounting values appear in the PBS accounting file; and the job values are usually viewed via `qstat`. The Scheduler values implicitly appear in the Scheduler's configuration file.

The `resources_assigned` values are reported differently for Vnodes (or Queues, or the Server) versus in the Accounting records. The value of `resources_assigned` reported for Vnodes (or Queues, or the Server) is the amount directly requested by jobs in the job's `Resource_List` (without regard to "excl"). The value of the job's `resource_assigned` (note the singular "resource") reported in the Accounting records is the actual amount assigned to the job by PBS (taking "excl" into account). The job's `resource_assigned` is not a job attribute. All allocated consumable resources will be included in the "resource_assigned" entries, one resource per entry. Consumable resources include ncpus, mem and vmem by default, and any custom resource defined with the -n or -f flags. A resource will not be listed if the job does not request it directly or inherit it by default from queue or server settings. For example, if a job requests one CPU on an Altix that has four CPUs per blade/vnode and that vnode is allocated exclusively to the job, even though the job requested one CPU, it is assigned all 4 CPUs.

7.9.9.1 Specifying Architectures

The `resources_available.arch` resource is the value reported by MOM unless explicitly set by the Administrator. The values for `arch` are:

Table 9: Values for resources_available.arch

OS	Resource Label
AIX 4, AIX 5	aix4

Table 9: Values for resources_available.arch

OS	Resource Label
HP-UX 10	hpux10
HP-UX 11	hpux11
IRIX	irix6
IRIX with cpusets	irix6cpuset
Linux	linux
Linux with cpusets	linux_cpuset
NEC	super-ux
Solaris	solaris7
Tru64	digitalunix
Unicos	unicos
Unicos MK2	unicosmk2
Unicos SMP	unicossmp

7.9.10 Setting Chunk Defaults

It is possible to set defaults on queues and the Server for resources used within a chunk. For example, the administrator could set the default for ncpus for chunks at the server. This means that if a job requests a certain chunk in which only mem and arch are defined, the default for ncpus will be added to that chunk.

Set the defaults for the server:

```
qmgr
Qmgr: set server  default_chunk.ncpus=1
Qmgr: set server  default_chunk.mem=1gb
```

Set the defaults for queue small:

```
qmgr
Qmgr: set queue small default_chunk.ncpus=1
Qmgr: set queue small default_chunk.mem=512mb
```

7.9.11 Defining New Resources

It is possible for the PBS Manager to define new resources within PBS Professional. 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 371.

7.10 Resource Defaults

The administrator can specify default resources on the server and queue. These resources can be job-wide, which is the same as adding -l RESOURCE to the job's resource request, or they can be chunk resources, which is the same as adding :RESOURCE=VALUE to a chunk. Job-wide resources are specified via resources_default on the server or queue, and chunk resources are specified via default_chunk on the server or queue. The administrator can also specify default resources to be added to any `qsub` arguments. In addition, the administrator can specify default placement of jobs.

For example, to set the default architecture on the server:

```
Qmgr: set server resources_default.arch=linux
```

To set default values for chunks, see section 7.9.10 "Setting Chunk Defaults" on page 230.

To set the default job placement for a queue:

```
Qmgr: set queue QUEUE resources_default.place=free
```

See the PBS Professional User's Guide for detailed information about how -l `place` is used.

To set the default rerunnable option in a job's resource request:

```
Qmgr: set server default_qsub_arguments="-r y"
```

Or to set a default boolean in a job's resource request so that jobs don't run on Red:

```
Qmgr: set server default_qsub_arguments="-l Red=false"
```

To set default placement involving a colon:

```
Qmgr: set server resources_default.place="pack:shared"
```

7.10.1 Jobs and Default Resources

Jobs get default resources, job-wide or per- chunk, with the following order of precedence.

Table 10: Order in which default resources are assigned to jobs

Order of assignment	Default value	Affects Chunks?	Job-wide?
1	Default qsub arguments	If specified	If specified
2	Queue's default_chunk	Yes	No
3	Server's default_chunk	Yes	No
4	Queue's resources_default	No	Yes
5	Server's resources_default	No	Yes
6	Queue's resources_max	No	Yes
7	Server's resources_max	No	Yes

See the `qmgr(8B)` man page for how to set these defaults.

For each chunk in the job's selection statement, first queue chunk defaults are applied, then server chunk defaults are applied. If the chunk does not contain a resource defined in the defaults, the default is added. The chunk defaults are called "default_chunk.RESOURCE".

For example, if the queue in which the job is enqueued has the following defaults defined:

```
default_chunk.ncpus=1
default_chunk.mem=2gb
```

a job submitted with this selection statement:

```
select=2:ncpus=4+1:mem=9gb
```

will have this specification after the default_chunk elements are applied:

```
select=2:ncpus=4:mem=2gb+1:ncpus=1:mem=9gb.
```

In the above, `mem=2gb` and `ncpus=1` are inherited from `default_chunk.`

The job-wide resource request is checked against queue resource defaults, then against server resource defaults. If a default resource is defined which is not specified in the resource request, it is added to the resource request.

7.10.1.1 Moving Jobs Between Queues

If the job is moved from the current queue to a new queue, any default resources in the job's resource list inherited from the queue are removed. This includes a select specification and place directive generated by the rules for conversion from the old syntax. If a job's resource is unset (undefined) and there exists a default value at the new queue or server, that default value is applied to the job's resource list. If either select or place is missing from the job's new resource list, it will be automatically generated, using any newly inherited default values.

Example: Given the following set of queue and server default values:

Server
`resources_default.ncpus=1`
Queue QA
`resources_default.ncpus=2`
`default_chunk.mem=2gb`
Queue QB
`default_chunk.mem=1gb`
no default for ncpus

The following illustrate the equivalent select specification for jobs submitted into queue QA and then moved to (or submitted directly to) queue QB:

`qsub -l ncpus=1 -lmem=4gb`
In QA: `select=1:ncpus=1:mem=4gb`
- No defaults need be applied

In QB: `select=1:ncpus=1:mem=4gb`
- No defaults need be applied

`qsub -l ncpus=1`
In QA: `select=1:ncpus=1:mem=2gb`
- Picks up 2gb from queue default chunk and 1 ncpus from qsub

In QB: `select=1:ncpus=1:mem=1gb`
- Picks up 1gb from queue default chunk and 1 ncpus from qsub

`qsub -lmem=4gb`

In QA: `select=1:ncpus=2:mem=4gb`
- Picks up 2 ncpus from queue level job-wide resource default and 4gb mem from qsub

In QB: `select=1:ncpus=1:mem=4gb`
- Picks up 1 ncpus from server level job-wide default and 4gb mem from qsub

`qsub -l nodes=4`

In QA: `select=4:ncpus=1:mem=2gb`
- Picks up a queue level default memory chunk of 2gb. (This is not 4:ncpus=2 because in prior versions, "nodes=x" implied 1 CPU per node unless otherwise explicitly stated.)

In QB: `select=4:ncpus=1:mem=1gb`
(In prior versions, "nodes=x" implied 1 CPU per node unless otherwise explicitly stated, so the ncpus=1 is not inherited from the server default.)

`qsub -l mem=16gb -l nodes=4`

In QA: `select=4:ncpus=1:mem=4gb`
(This is not 4:ncpus=2 because in prior versions, "nodes=x" implied 1 CPU per node unless otherwise explicitly stated.)

In QB: `select=4:ncpus=1:mem=4gb`
(In prior versions, "nodes=x" implied 1 CPU per node unless otherwise explicitly stated, so the ncpus=1 is not inherited from the server default.)

7.11 Server and Queue Resource Min/Max Attributes

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 and is not assigned that resource through default qsub arguments, 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.

If a job is submitted without a request for a specific resource, and that resource is specified in the server or queue resources_max, the job may inherit that value for that resource. Whether the job inherits the value in resources_max is determined by the order of inheritance given in section 7.10.1 "Jobs and Default Resources" on page 232.

7.12 Selective Routing of Jobs into Queues

You may want to route jobs to various queues on a Server, or even between Servers, based on the resource requirements of the jobs. The queue attributes *resources_min* and *resources_max* discussed allow this selective routing. The queue's resources_min/max can only be used with job-wide resources. You cannot use custom host-level resources with queue resources_min/max. This would include any custom resources created with flag=h. That is, you cannot use a custom resource defined with flag=h.

Jobs can only be routed based on resources outside of the select specification, or based on sums of nodal resources.

If you want to use a boolean resource to route jobs w/resources_min/max you will have to define it at the server or queue level (without flag=h.) It will have to be requested with "-l select=x -l <boolean resource>=True". A server or queue-level resource cannot be used to direct a job to an execution node.

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` attribute 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, its 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.

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, and it is not given the resource through defaults, (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`. 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` attribute at a queue level only applies to jobs in that queue.

The check for admission of a job to a queue has the following sequence:

1. Clear the job's current defaults (from both existing queue and server)
2. Set new defaults based on named destination queue
3. Test limits against queue min/max and server min/max
4. Clear the job's new defaults
5. Reset the defaults based on the actual queue in which the job resides

If a queue resource 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 become 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.

If the job is to be moved into a different queue, then the default values are again cleared and reset based on that destination queue. This happens as the job is enqueued.

If a resource is not set on job submission, it is not checked against the queue's min/max. If no default was set, it won't be included in the Resource_List. The resources_min/max are only checked against equivalent entries in the job's Resource_List. Only consumable resources (those with flag=n or q) are taken from the select specification and turned into separate entries in the Resource_List.

7.12.1 Checks Performed When Jobs are Admitted Into Queues

When a job is being considered for a queue because it was submitted or it was qmoved, the following checks are performed:

Step 1 Any current defaults, either from the server or the current queue, are cleared.

Step 2 New defaults, based on the potential destination queue, are set.

Step 3 The job's limits are tested against the queue and server minima/maxima.

Step 4 The new defaults are cleared.

Step 5 Final defaults are set based on which queue the job was actually enqueued in.

7.13 Overview of Advance Reservations

An *Advance Reservation* is a set of resources with availability limited to a specific user (or group of users), a specific start time, and a specified duration. Users submit reservation requests, and then PBS either confirms or rejects the reservation. Once the reservation is confirmed, the queue that was created to support this reservation will be enabled, allowing jobs to be submitted to it. The queue will have a user level access control list set to the user who submitted the reservation and any other users the owner specified. The queue will accept jobs in the same manner as normal queues. When the reservation start time is reached, the queue will be started. Once the reservation is complete, any jobs remaining in the queue or still running will be deleted, and the reservation removed. When a reservation is requested and confirmed, it means that a check was made to see if the reservation would conflict with currently running jobs, other confirmed reservations, and dedicated time. A reservation request that fails this check is denied. If there are insufficient resources, the scheduler won't run a reservation job. For example, if the reservation is for one hour, but a job is submitted with a walltime of 2 hours, the job will not be started.

Leave enough time between reservations for the reservations and jobs in them to clean up. A job consumes reservations even while it is in the "E" or exiting state. This can take longer when large files are being staged. If the job is still running when the reservation ends, it may take up to two minutes to be cleaned up. The reservation itself cannot finish

cleaning up until its jobs are cleaned up. This will delay the start time of jobs in the next reservation unless there is enough time between the reservations for cleanup.

Example: To submit a reservation for 1 chunk, with 2 CPUs and 100MB of memory at 3:30pm for 30 minutes and named MyResv:

```
pbs_rsub -N MyResv -R 1530 -D 30:00
   -l select=1ncpus=2:mem=100mb
```

And you would see:

```
R123.myhost UNCONFIRMED
```

Hosts/vnodes that have been configured to accept jobs only from a specific queue (vnode-queue restrictions) cannot be used for advance reservations.

To delete an advance reservation, use the `pbs_rdel` command, not the `qmgr` command.

For additional information on configuring your system to use the advance reservation feature, see the various `acl_resv_*` Server configuration attributes in section 7.5 "Server Configuration Attributes" on page 182.

7.13.1 Advance Reservations and FLEX Licensing

Reservation jobs won't run if PBS runs out of FLEX licenses. Set the server's `pbs_license_min` attribute to the total number of CPUs, including virtual CPUs, in the PBS complex. See section 5.9.1.3 "Licensing and Advance Reservations" on page 103 and section 5.4.3 "Setting Server Licensing Attributes" on page 88.

7.14 SGI Weightless CPU Support

Submitting a job and requesting `-l ncpus=0` is legal. In a non-cpuset SGI IRIX 6.x environment, the job's kernel scheduling priority will be set "weightless". There will be no allocation at the Server, Queue, or Vnode 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
```

7.15 Password Management for Windows

PBS Professional will allow users to specify two kinds of passwords: a per-user/per-server password, or a per-job password. The PBS administrator must choose which method is to be used. (Discussion of the difference between these two methods is given below; detailed usage instructions for both are given in the **PBS Professional User's Guide**.)

This feature is intended for Windows environments. It should *not* be enabled in UNIX since this feature requires the PBS_DES_CRED feature, which is not enabled in the normal binary UNIX version of PBS Professional. Setting this attribute to "true" in UNIX may cause users to be unable to submit jobs.

The per-user/per-server password was introduced as part of the single signon password scheme. The purpose is to allow a user to specify a password only once and have PBS remember this password to run the user's current and future jobs. A per-user/per-server password is specified by using the command:

```
pbs_password
```

The user must run this command before submitting jobs to the Server. The Server must have the single_signon_password_enable attribute set to "true".

Alternatively, one can configure PBS to use the current per-job password scheme. To do this, the Server configuration attribute single_signon_password_enable must be set to "false", and jobs must be submitted using:

```
qsub -Wpwd
```

You cannot mix the two schemes; PBS will not allow submission of jobs using -Wpwd when single_signon_password_enable is set to "true".

Important: If you wish to migrate from an older version of PBS Professional on Windows to the current version, be sure to review Chapter 5 of this document, as well as the discussion of `pbs_migrate_users` in Chapter 11.

7.15.1 Single Signon and the qmove Command

A job can be moved (via the `qmove` command) from a Server at hostA to a Server at hostB. If the Server on hostB has `single_signon_password_enable` set to `true`, then the user at hostB must have an associated per-user/per-server password. This requires that the user run `pbs_password` at least once on hostB.

7.15.2 Single Signon and Invalid Passwords

If a job's originating Server has `single_signon_password_enable` set to `true`, and the job fails to run due to a bad password, the Server will place a hold on the job of type "`p`" (bad password hold), update the job's comment with the reason for the hold, and email the user with possible remedy actions. The user (or a manager) can release this hold type via:

```
qrls -h p <jobid>
```

7.15.3 Single Signon and Peer Scheduling

In a peer scheduling environment, the Scheduler may move jobs from complex A to complex B. If the Server in complex B has `single_signon_password_enable` attribute set to `true`, then users with jobs on complex A must make sure they have per-user/per-server passwords on complex B. This is done by issuing a `pbs_password` command on complex B.

7.16 Configuring PBS Redundancy and Failover

The redundancy-failover feature of PBS Professional provides the capability for a backup Server to assume the workload of a failed Server, thus eliminating the one single point of failure in PBS Professional. If the Primary Server fails due to a hardware or software error, the Secondary Server will take over the workload 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 Professional 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.

The server attribute values for `pbs_license_file_location`, `pbs_license_min`, `pbs_license_max`, and `pbs_license_linger_time` are set through the primary server. Since these values are saved in PBS_HOME/server_priv/serverdb, and PBS_HOME is in a shared location, the secondary server can use these licensing parameters. No additional licensing steps are needed for the secondary server to work properly.

7.16.1 Failover Requirements

The following requirements must be met to provide a reliable failover service:

1. The Primary and Secondary Servers must be run on different hosts. Only one Secondary Server is permitted.

2. The Primary and Secondary Server hosts must be the same architecture, i.e. binary compatible, including word length, byte order and padding within the structures.

3. Both the Primary and Secondary Server host must be able to communicate over the network with all execution hosts where a `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. The directory must be readable and writable by root on UNIX, or have Full Control permissions for the local "Administrators" group on the local host on Windows.

   When selecting the failover device, consider both the hardware and the available file systems, as the solution needs to support concurrent read and write access from two hosts. The best solution is a high availability file server device connected to both the Primary and Secondary Server hosts, used in conjunction with a file system that supports both multiple export/mounting and simultaneous read/write access from two or more hosts

(such as SGI CXFS, IBM GPFS, or Red Hat GFS).

To avoid introducing a single point of failure, use an NFS file server with the file system exported to and *hard mounted* by both the Primary and Secondary Server hosts. Make sure that neither server host is the machine on which the PBS_HOME file system resides.

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 and secondary server hosts.

5. The /etc/hosts files on the two servers must be set up so that each can find the other and all the hosts in the complex.

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

6. A MOM, `pbs_mom`, may run on either the Primary or the Secondary hosts, or both, however, this is not recommended. 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 the `pbs.conf` file. The `PBS_MOM_HOME` entry specifies a directory which has the following contents:

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

Directory Contents	Description
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)

Windows:

Note: In the table below, references to “access to Admin-account” refer to access to the local Administrators group on the local host.
.

Directory Contents	Description
auxiliary	Directory with full access to *Admin-account* and read-only access to Everyone
checkpoint	Directory with full access only to *Admin-account*
mom_logs	Directory with full access to *Admin-account* and read-only access to Everyone
mom_priv	Directory with full access to *Admin-account* and read-only access to Everyone
mom_priv/jobs	Subdirectory with full access to *Admin-account* and read-only access to Everyone
mom_priv/config	File with full access-only to *Admin-account*
pbs_environment	File with full access to *Admin-account* 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 the pbs.conf file,

pbs_mom will use that directory for its "home" instead of PBS_HOME.

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

7.16.2 Failover Configuration for UNIX/Linux

The steps below outline the process for general failover setup, and should be sufficient for configuration under UNIX. To configure PBS Professional for failover operation, follow these steps:

1. Select two systems of the same architecture to be the Primary and Secondary Server systems. They should be binary compatible.

2. Configure a file system (or at least a directory) that is read/write accessible by root (UNIX) from both systems. If an NFS file system is used, it must be "hard mounted" (UNIX) and root or Administrator must have access to read and write as "root" or as "Administrators" on both systems. Beware of dependencies on remote file systems: PBS depends on the paths in $PBS_CONF being available when its startup script is executed, PBS will hang if a remote file access hangs, and normal privileges don't necessarily carry over for access to remote file 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").

   The NFS lock daemon, lockd, must be running for the file system on the primary and secondary hosts.

3. Install PBS Professional on both systems, specifying the shared file system location for the PBS_HOME directory. DO NOT START ANY PBS DAEMONS.

4. Modify /etc/pbs.conf file on both systems, as follows:

5. Change PBS_SERVER on both systems to the short form of the Primary Server's hostname. The value must be a valid hostname. Example:

```
PBS_SERVER=servername
```

6. Add the following entries to both pbs.conf files; they must have the same value in both files:

```
PBS_PRIMARY=primaryname.domain.com
PBS_SECONDARY=secondaryname.domain.com
```

where "**primaryname.domain.com**" is the fully qualified host name of the Primary Server's host, and "**secondaryname.domain.com**" is the fully qualified host name of the Secondary Server's host. It is important that these entries be correct and distinct as they are used by the Servers to determine their status at startup.

These entries must also be added to the pbs.conf file on any system on which the PBS commands are installed, and on all execution hosts in the complex.

A sample /etc/pbs.conf file for each server:

Primary:

```
PBS_START_SERVER=1
PBS_START_MOM=0
PBS_START_SCHED=1
PBS_SERVER=primaryname.domain.com
PBS_PRIMARY=primaryname.domain.com
PBS_SECONDARY=secondaryname.domain.com
```

Secondary:

```
PBS_START_SERVER=1
PBS_START_MOM=0
PBS_START_SCHED=0
PBS_SERVER=primaryname.domain.com
PBS_PRIMARY=primaryname.domain.com
PBS_SECONDARY=secondaryname.domain.com
```

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

8. On the Secondary host, modify the pbs.conf file to not start the Scheduler by setting

   PBS_START_SCHED=0

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

9. It is not recommended to run pbs_mom on both the Primary and Secondary Servers hosts. If you do run a pbs_mom on both the Primary and Secondary Server hosts, make sure that /etc/pbs.conf on each host has a PBS_MOM_HOME defined. This will be local to that host. You will need to replicate the PBS_MOM_HOME directory structure at the place specified by PBS_MOM_HOME.

10. 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:

    pbs_server -F <delay>

11. The 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.

12. Start up the primary and secondary servers in any order.

13. Once the Primary Server is started, use the qmgr command to

set or modify the Server's "mail_from" attribute 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.

14. If you have acl_hosts and acl_host_enable set on the server, you must add the failover host to the list. Use the qmgr command:

```
Qmgr: s server acl_hosts+=<secondary server>
```

7.16.3 Failover Configuration for Windows

Under Windows, configure Server failover from the console of the hosts or through VNC. Setting up the Server failover feature from a Remote Desktop environment will cause problems. In particular starting of the Server in either the primary host or secondary host would lead to the error:

error 1056: Service already running

even though PBS_HOME\server_priv\server.lock and PBS_HOME\server_priv\server.lock.secondary files are non-existent.

The following illustrates how PBS can be set up on Windows with the Server failover capability using the network share facility. That is, the primary and secondary Server/Scheduler will share a PBS_HOME directory that is located on a network share file system on a remote host. In this scenario a primary pbs_server is run on hostA, a secondary Server is run on hostB, and the shared PBS_HOME is set up on hostC using Windows network share facility.

Important: Note that hostC must be set up on a Windows 2000 Server, Windows 2000 Advanced Server, or Windows Server 2003 platform.

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

2. Next 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. Now configure a shared PBS_HOME by doing the following:

a. Go to hostC; create a folder named e.g., C:\pbs_home.

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" permissions for the local Administrators group on the local computer.

4. 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=\\hostC\pbs_home"
```

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 \\hostC\pbs_home"
```

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

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

6. 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"
```

7. Set up the secondary Server so that it will only start the scheduler when it takes over from the Primary, and not when it is rebooted.

On the secondary Server modify the `pbs.conf` file to start the scheduler by running:

```
pbs-config-add “PBS_START_SCHED=1”
```

Then go to the Control Panel->Administrative Tools->Services, and bring up the PBS_SCHED service dialog, select General tab, and specify "Manual" for Startup type.

In this way, when the secondary host is rebooted, the scheduler won't automatically start up. Instead, the server can bring it up manually when it takes over for the primary Server. If the secondary server is told to take over and the primary host is still down, then the secondary server will start the scheduler via "net start pbs_sched".

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. If you have `acl_hosts` and `acl_host_enable` set on the server, you must add the failover host to the list. Use the `qmgr` command:

```
Qmgr: s server acl_hosts+=<secondary server>
```

10. 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->Ser-`

vices, 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.

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

    Qmgr: **set server managers="<*account that installed PBS*>@*,pbsadmin@*"**

Important: Set up of the Server failover feature in Windows may encounter problems if performed from a Remote Desktop environment. In particular, starting the Server on either the primary host or secondary host would lead to the error:

```
error 1056  Service already running
```

even though PBS_HOME\server_priv\server.lock and PBS_HOME\server_priv\server.lock.secondary files are non-existent. To avoid this, configure Server failover from the console of the hosts or through VNC.

Important: Under certain conditions under Windows, the primary Server fails to take over from the secondary even after it is returned into the network. The workaround, should this occur, is to reboot the primary Server machine.

7.16.4 Failover: Normal Operation

The Primary Server and the Secondary Server may be started by hand, or 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 to 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 -F as a start parameter given by the `Start-> Control Panel-> Administrator Tools-> Services->` `PBS_SERVER` dialog.

The primary and the secondary server use different lock files:
primary: server.lock
secondary: server.lock.secondary.

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 On UNIX, 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
```

Check that all daemons required by your NFS are running.

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 Secondary must register itself in order to take over should the Primary fail. The Primary will reply with information to allow the Secondary to use the license file location should it become active. The Primary and Secondary use the same license information, which is set through the Primary.

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 attribute `mail_from`. The Secondary will inform the `pbs_mom` on the configured vnodes 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 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`. If the secondary Server is started manually, it will not start its own scheduler. Since that is a manual operation, it is assumed that the manual operation will also start the Scheduler.

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`" attribute 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.

7.16.5 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 (standby 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 shut down and restart the Secondary Server while it is active, and wish to keep it 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. The default delay is 30 seconds.

7.16.6 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
```

7.16.7 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 9.17.3 "Peer Scheduling and Failover Configuration" on page 365.

7.17 Recording Server Configuration

If 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 vnodes and all the vnode properties. The commands could be read back into `qmgr` via standard input:

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

7.18 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 vnodes may be defined with the same vnode name. One may be a Globus vnode (MOM), and the other a non-Globus vnode. 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.

Important: Globus support is not currently available on Windows.

7.19 Configuring the Server for FLEX Licensing

The PBS server must be configured for FLEX licensing. You must set the location where PBS will look for the license server, by setting the server attribute pbs_license_file_location. The other server licensing attributes have defaults, but you may wish to set them as well. See section 5.4 “Configuring PBS for Licensing” on page 85.

You may also wish to have redundant license servers. See section 5.5 “Redundant License Servers” on page 93

Chapter 8
Configuring MOM

The installation process creates a basic MOM configuration file which contains the minimum necessary in order to run PBS jobs. This chapter describes the MOM configuration files, and explains all the options available to customize the PBS installation to your site.

The organization of this chapter has changed. Information specific to configuring machines such as the Altix is presented in section 8.9 "Configuring MOM for Machines with cpusets" on page 291.

8.1 Introduction

The pbs_mom command starts the PBS job monitoring and execution daemon, called MOM. The pbs_mom daemon starts jobs on the execution host, monitors and reports resource usage, enforces resource usage limits, and notifies the server when the job is finished. The MOM also runs any prologue scripts before the job runs, and runs any epilogue scripts after the job runs.

The MOM performs any communication with job tasks and with other MOMs. The MOM on the first vnode on which a job is running manages communication with the MOMs on the remaining vnodes on which the job runs.

The MOM manages one or more vnodes. PBS may treat a host such as an Altix as a set of virtual nodes, in which case one MOM manages all of the host's vnodes. See section 7.7 “Vnodes: Virtual Nodes” on page 205.

The MOM's error log file is in PBS_HOME/mom_logs. The MOM writes an error message in its log file when it encounters any error. If it cannot write to its log file, it writes to standard error.

The executable for pbs_mom is in PBS_EXEC/sbin, and can be run only by root.

For information on starting and stopping MOM, see section 11.4.4 “Manually Starting MOM” on page 407.

8.1.1 Single- vs. Multi-vnoded Systems

The following section contains information that applies to all PBS MOMs. The PBS MOM pbs_mom.cpuset has extensions to take manage multi-vnoded systems such as the Altix. These systems can be subdivided into more than one virtual node, or vnode. PBS manages each vnode as if it were a host. While the information in this section is true for all MOMs, any information that is specific to multi-vnoded systems is in section 8.9 “Configuring MOM for Machines with cpusets” on page 291.

8.2 MOM Configuration Files

The behavior of each MOM is controlled through its configuration files. MOM reads the configuration files at startup and reinitialization. On UNIX, this is when `pbs_mom` receives a SIGHUP signal or is started or restarted, and on Windows, when MOM is started or restarted.

MOM's configuration information can be contained in configuration files of three types: default, PBS reserved, and site-defined. The default configuration file is usually PBS_HOME/mom_priv/config. PBS reserved configuration files are created by PBS and are prefixed with "PBS". Site-defined configuration files are those created by the site administrator.

Any PBS reserved MOM configuration files are only created when PBS is started, not when the MOM is started. Therefore, if you make changes to the hardware or a change occurs in the number of CPUs or amount of memory that is available to PBS, such as a non-PBS process releasing a cpuset, you should restart PBS in order to re-create the PBS reserved MOM configuration files.

When MOM is started, it will open its default configuration file, `mom_priv/config`, in the path specified in `pbs.conf`, if the 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 section 11.4.4 "Manually Starting MOM" on page 407.

The files are processed in this order:

The default configuration file
PBS reserved configuration files
Site-defined configuration files

Within each category, the files are processed in lexicographic order.

The contents of a file that is read later will override the contents of a file that is read earlier.

8.2.1 Creation of Site-defined MOM Configuration Files

To change the cpuset flags, create a file "update_flags" containing only

```
cpuset_create_flags CPUSET_CPU_EXCLUSIVE
```

then use the `pbs_mom -s insert <script> <filename>` option to create the script:

```
pbs_mom -s insert update_script update_flags
```

The script `update_script` is the new site-defined configuration file. Its contents will override previously-read `cpuset_create_flags` settings.

Configuration files can be listed, added, deleted and displayed using the `-s` option. An attempt to create or remove a file with the "PBS" prefix will result in an error. See section 11.4.4 "Manually Starting MOM" on page 407 for information about pbs_mom options.

MOM's configuration files can use the syntax shown below in section 8.2.2 "Syntax and Contents of Default Configuration File" on page 260, or the syntax for describing vnodes shown in section 8.9.1.2 "Syntax of Version 2 PBS Reserved Configuration Files" on page 292.

8.2.1.1 Location of MOM's Configuration Files

The default configuration file is in PBS_HOME/mom_priv/.

PBS places PBS reserved and site-defined configuration files in an area that is private to each installed instance of PBS. This area may change with future releases. Do not attempt to manipulate these files directly. This area is relative to the default PBS_HOME. Note that the `-d` option changes where MOM looks for PBS_HOME, and using this option will prevent MOM from finding any but the default configuration file. If you use the -d option, MOM will look in the wrong place for any PBS reserved and site-defined files.

Do not directly create PBS reserved or site-defined configuration files; instead, use the `pbs_mom -s` option. See section 11.4.4 “Manually Starting MOM” on page 407 for information on `pbs_mom`.

The `-c` option will change which default configuration file MOM reads.

Site-defined configuration files can be moved from one installed instance of PBS to another. Do not move PBS reserved configuration files. To move a set of site-defined configuration files from one installed instance of PBS to another:

1.Use the -s list directive with the "source" instance of PBS to enumerate the site-defined files.

2.Use the -s show directive with each site-defined file of the "source" instance of PBS to save a copy of that file.

3.Use the -s insert directive with each file at the "target" instance of PBS to create a copy of each site-defined configuration file.

8.2.2 Syntax and Contents of Default Configuration File

Configuration files with this syntax list local resources and initialization values for MOM. Local resources are either static, listed by name and value, or externally-provided, listed by name and command path. Local static resources are for use only by the scheduler. They do not appear in a `pbsnodes -a` query. See the -c option. Do not change the syntax of the default configuration file.

Each configuration item is listed on a single line, with its parts separated by white space. Comments begin with a hashmark ("#").

The default configuration file must be secure. It must be owned by a user ID and group ID both less than 10 and must not be world-writable.

8.2.2.1 Externally-provided Resources

Externally-provided resources use a shell escape to run a command. These resources are described with a name and value, where the first character of the value is an exclamation mark ("!"). The remainder of the value is the path and command to execute.

Parameters in the command beginning with a percent sign ("%") can be replaced when the command is executed. For example, this line in a configuration file describes a resource named "escape":

```
escape   !echo %xxx %yyy
```

If a query for the "escape" resource is sent with no parameter replacements, the command executed is "echo %xxx %yyy". If one parameter replacement is sent, "escape[xxx=hi there]", the command executed is "echo hi there %yyy". If two parameter replacements are sent, "escape[xxx=hi][yyy=there]", the command executed is "echo hi there". If a parameter replacement is sent with no matching token in the command line, "escape[zzz=snafu]", an error is reported.

8.2.2.2 Initialization Values

Initialization value directives have names beginning with a dollar sign ("$"). They are listed here:

```
$action <default_action> <timeout> <new_action>
```

Replaces the `default_action` for an event with the site-specified `new_action`. `timeout` is the time allowed for `new_action` to run. The `default_action` can be one of:

Table 11: How $action is Used

default_action	Result
checkpoint	Run `new_action` in place of the periodic job checkpoint, after which the job continues to run.
checkpoint_abort	Run `new_action` to checkpoint the job, after which the job is terminated.

Table 11: How $action is Used

default_action	Result
multinodebusy	Used with cycle harvesting and multi-vnode jobs. Changes default behavior when a vnode becomes busy. Instead of allowing the job to run, the job is requeued. The `new_action` is requeue.
restart	Runs `new_action` in place of restart.
terminate	Runs `new_action` in place of SIGTERM or SIGKILL when MOM terminates a job.

`$checkpoint_path <path>`

MOM will write checkpoint files in the directory given by `path`. This path can be absolute or relative to `PBS_HOME/mom_priv`.

`$clienthost <hostname>`

`hostname` is added to the list of hosts which will be allowed to connect to MOM as long as they are using a privileged port. For example, this will allow the hosts "fred" and "wilma" to connect to MOM:

```
$clienthost       fred
$clienthost       wilma
```

The following hostnames are added to $clienthost automatically. The server and the localhost are automatically added to $clienthost. If configured, the secondary server is also added to $clienthost. The server sends each MOM a list of the hosts in the nodes file, and these are added internally to $clienthost. None of these hostnames need to be listed in the configuration file.

The hosts in the nodes file make up a "sisterhood" of machines. Any one of the sisterhood will accept connections from within the sisterhood. The sisterhood must all use the same port number.

`$cputmult <factor>`

This sets a factor used to adjust CPU time used by each job. This allows adjustment of time charged and limits enforced where jobs run on a system with different CPU performance. If MOM's system is faster than the reference system, set this factor to a decimal value greater than 1.0. For example:

```
$cputmult 1.5
```

If MOM's system is slower, set this factor to a value between 1.0 and 0.0. For example:

```
$cputmult 0.75
```

`$dce_refresh_delta <delta>`

Defines the number of seconds between successive refreshings of a job's DCE login context. For example:

```
$dce_refresh_delta 18000
```

`$enforce <limit>`

MOM will enforce the given limit. Some limits have associated values, and appear in the configuration file like this:

```
$enforce variable_name value
```

See section 8.8 "Resource Limit Enforcement" on page 283.

`$enforce mem` MOM will enforce each job's memory limit. See section 8.8 "Resource Limit Enforcement" on page 283.

`$enforce cpuaverage`

MOM will enforce `ncpus` when the average CPU usage over a job's lifetime usage is greater than the job's limit. See section 8.8.2.1 "Average CPU Usage Enforcement" on page 287.

`$enforce average_trialperiod <seconds>`

Modifies cpuaverage. Minimum number of seconds of job walltime before enforcement begins. Default: 120. Integer.
See section 8.8.2.1 "Average CPU Usage Enforcement" on page 287.

`$enforce average_percent_over <percentage>`

Modifies cpuaverage. Gives percentage by which a job may exceed its ncpus limit. Default: 50. Integer. See section 8.8.2.1 "Average CPU Usage Enforcement" on page 287.

`$enforce average_cpufactor <factor>`
: Modifies cpuaverage. The ncpus limit is multiplied by factor to produce actual limit. Default: 1.025. Float. See section 8.8.2.1 “Average CPU Usage Enforcement” on page 287.

`$enforce cpuburst`
: MOM will enforce the ncpus limit when CPU burst usage exceeds the job's limit. See section 8.8.2.2 “CPU Burst Usage Enforcement” on page 288.

`$enforce delta_percent_over <percentage>`
: Modifies cpuburst. Gives percentage over limit to be allowed. Default: 50. Integer. See section 8.8.2.2 “CPU Burst Usage Enforcement” on page 288.

`$enforce delta_cpufactor <factor>`
: Modifies cpuburst. The ncpus limit is multiplied by factor to produce actual limit. Default: 1.5. Float. See section 8.8.2.2 “CPU Burst Usage Enforcement” on page 288.

`$enforce delta_weightup <factor>`
: Modifies cpuburst. Weighting factor for smoothing burst usage when average is increasing. Default: 0.4. Float. See section 8.8.2.2 “CPU Burst Usage Enforcement” on page 288.

`$enforce delta_weightdown <factor>`
: Modifies cpuburst. Weighting factor for smoothing burst usage when average is decreasing. Default: 0.4. Float. See section 8.8.2.2 “CPU Burst Usage Enforcement” on page 288.

`$ideal_load <load>`
: Defines the load below which the host is not considered to be busy. Used with the $max_load directive. No default. Float. Example:

```
$ideal_load 1.8
```

  Use of $ideal_load adds a static resource called "ideal_load", which is only internally visible.

`$kbd_idle <idle_wait> <min_use> <poll_interval>`

Declares that the host will be used for batch jobs during periods when the keyboard and mouse are not in use.

The host must be idle for a minimum of idle_wait seconds before being considered available for batch jobs. No default. Integer.

The host must be in use for a minimum of min_use seconds before it becomes unavailable for batch jobs. Default: 10. Integer.

Mom checks for activity every poll_interval seconds. Default: 1. Integer.

Example:

```
$kbd_idle 1800 10 5
```

`$logevent <mask>`

Sets the mask that determines which event types are logged by pbs_mom. To include all debug events, use 0xffffffff.

Table 12: Log Events

Name	Hex Val	Message Category
PBSE_ERROR	0001	Internal errors
PBSE_SYSTEM	0002	System errors
PBSE_ADMIN	0004	Administrative events
PBSE_JOB	0008	Job-related events
PBSE_JOB_USAGE	0010	Job accounting info
PBSE_SECURITY	0020	Security violations
PBSE_SCHED	0040	Scheduler events
PBSE_DEBUG	0080	Common debug messages
PBSE_DEBUG2	0100	Uncommon debug messages
PBSE_RESV	0200	Reservation-related info

Table 12: Log Events

Name	Hex Val	Message Category
PBSE_DEBUG3	0400	Rare debug messages

`$max_check_poll <seconds>`
Maximum time between polling cycles, in seconds. Must be greater than zero. Integer.

`$min_check_poll <seconds>`
Minimum time between polling cycles, in seconds. Must be greater than zero and less than $max_check_poll. Integer.

`$max_load <load> [suspend]`
Defines the load above which the host is considered to be busy. Used with the `$ideal_load` directive. No default. Float. Example:

```
$max_load 3.5 suspend
```

Use of `$max_load` adds a static resource to the vnode called "max_load", which is only internally visible.

The optional "suspend" directive tells PBS to suspend jobs running on the node if the load average exceeds the max_load number, regardless of the source of the load (PBS and/or logged-in users). Without this directive, PBS will not suspend jobs due to load.

`$prologalarm <timeout>`
Defines the maximum number of seconds the prologue and epilogue may run before timing out. Default: 30. Integer. Example:

```
$prologalarm 30
```

`$restart_background <true|false>`
Controls how MOM runs a restart script after checkpointing a job.
When this option is set to true, MOM forks a child which runs the restart script. The child returns when all restarts for all the local tasks of the job are done. MOM does not block on the

restart. When this option is set to false, MOM runs the restart script and waits for the result. Boolean. Default: false.

`$restart_transmogrify <true|false>`
: Controls how MOM runs a restart script after checkpointing a job. When this option is set to true, MOM runs the restart script, replacing the session ID of the original task's top process with the session ID of the script.

  When this option is set to false, MOM runs the restart script and waits for the result. The restart script must restore the original session ID for all the processes of each task so that MOM can continue to track the job.

  When this option is set to false and the restart uses an external command, the configuration parameter restart_background is ignored and treated as if it were set to true, preventing MOM from blocking on the restart.

  Boolean. Default: false

`$restrict_user <value>`
: Controls whether users not submitting jobs have access to this machine. If value is "on", restrictions are applied. The interval between when PBS applies restrictions can be at most 10 seconds. See `$restrict_user_exceptions` and `$restrict_user_maxsysid`. Boolean. Default: off.

`$restrict_user_exceptions <user_list>`
: Comma-separated list of users who are exempt from access restrictions applied by `$restrict_user`. Leading spaces within each entry are allowed. Maximum number of names in list is 10.

`$restrict_user_maxsysid <value>`
: Any user with a numeric user ID less than or equal to value is exempt from restrictions applied by `$restrict_user`.

  If `$restrict_user` is on and no value exists for `$restrict_user_maxsysid`, PBS looks in `/etc/login.defs for` SYSTEM_UID_MAX for the value. If

there is no maximum ID, it looks for SYSTEM_MIN_UID, and uses that value minus 1. Otherwise the default is used.

Integer. Default: 999.

`$restricted <hostname>`

The hostname is added to the list of hosts which will be allowed to connect to MOM from a non-privileged port. Hostnames can be wildcarded. For example, to allow queries from any host from the domain "xyz.com":

```
$restricted      *.xyz.com
```

Queries from the hosts in the $restricted list are only allowed access to information internal to the host managed by this MOM, such as load average, memory available, etc. They may not run shell commands. No machines are added automatically to this list.

`$suspendsig <suspend_signal> [resume_signal]`

Alternate signal suspend_signal is used to suspend jobs instead of SIGSTOP. Optional resume_signal is used to resume jobs instead of SIGCONT.

`$tmpdir <directory>`

Location where each job's scratch directory will be created. Default: /tmp. For example:

```
$tmpdir /memfs
```

`$usecp <hostname:source_prefix> <destination_prefix>`

MOM will use /bin/cp (or xcopy on Windows) to stage in/out files or deliver output when the source and destination are both on the local host. In addition, the administrator can use $usecp to list other source locations that can be directly copied to/from local destinations. Both source_prefix and destination_prefix are absolute pathnames of directories, not files. For example:

```
$usecp      *.example.com:/home/ /home/
```

This says any file available remotely under the /home/ directory is also directly available to the MOM in the /home/ directory.

```
$usecp       HostA:/users/work/myproj/  \
        /sharedwork/proj_results/
```

This says that a staging or output reference to a file under /users/ work/myproj/ on HostA should instead refer to a file under /sharedwork/proj_results/ on the MOM.

`$wallmult <factor>`

Each job's walltime usage is multiplied by this factor. For example:

```
$wallmult 1.5
```

8.2.2.3 Static MOM Resources

Local static resources are for use only by the scheduler. They do not appear in a `pbsnodes -a` query. Static resources local to the MOM are described one resource to a line, with a name and value separated by white space. For example, tape drives of different types could be specified by:

tape3480 4

tape3420 2

tapedat 1

tape8mm 1

`memreserved <megabytes>`

The amount of per-vnode memory reserved for system overhead. This much memory is deducted from the value of resources_available.mem for each vnode managed by this MOM. Default is 0MB. For example,

```
memreserved 16
```

8.2.2.4 Windows Notes

If the argument to a MOM option is a pathname containing a space, enclose it in double quotes as in the following:

hostn !"\Program Files\PBS Pro\exec\bin\hostn" *host*

8.3 Configuring MOM's Polling Cycle

MOM's polling cycle is set by `$min_check_poll` and `$max_check_poll`. The interval between each poll starts at `$min_check_poll` and increases with each cycle until it reaches `$max_check_poll`, after which it remains the same. The amount by which the cycle increases is 1/20 of the difference between `$max_check_poll` and `$min_check_poll`.

MOM polls for resource usage for cput, walltime, mem and ncpus. See section 8.8 "Resource Limit Enforcement" on page 283. Job-wide limits are enforced by MOM using polling. See section 8.8.1 "Job Memory Limit Enforcement on UNIX" on page 284. MOM can enforce cpuaverage and cpuburst resource usage; see section 8.8.2.1 "Average CPU Usage Enforcement" on page 287 and section 8.8.2.2 "CPU Burst Usage Enforcement" on page 288.

MOM enforces the $restrict_user access restrictions on a polling cycle which can be set to a maximum of 10 seconds. See section 8.7 "Restricting User Access to Execution Hosts" on page 282.

Cycle harvesting has its own polling interval. See the information for $kbd_idle in section 8.2.2.2 "Initialization Values" on page 261.

8.4 Configuring MOM Resources

8.4.1 Static MOM Resources

Configure static vnode-level resources using `qmgr`.
Example:

```
Qmgr: set node VNODE resources_available.RES = <value>
```

While it is possible to configure static resources in the MOM configuration file, it is not recommended. Qmgr is preferred because (1) the change takes effect immediately, as opposed to having to send a HUP signal to MOM; and (2) all such static resources can be centrally managed and viewed via `qmgr`. For more information on creating site-specific resources, see Chapter 9, "Customizing PBS Resources" on page 371.

That being said, to specify static resource names and values in the MOM configuration file, you can add a list of resource name/value pairs, one pair per line, separated by white space.

8.4.2 Dynamic MOM Resources

Configure dynamic vnode-level resources by adding shell escapes to the MOM configuration file, *PBS_HOME*/mom_priv/config . The primary use of this feature is to add site-specific resources, such as software application licenses. The form is:

RESOURCE_NAME !*path-to-command*

The RESOURCE_NAME specified should be the same as the corresponding entry in the Server's *PBS_HOME*/server_priv/resourcedef file. See Chapter 9, "Customizing PBS Resources" on page 371 and section 10.7 "Application Licenses" on page 388.

8.5 Configuring MOM for Site-Specific Actions

8.5.1 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 parameter 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. If the *SCRIPT_PATH* specifies a full path, be sure to include the drive letter so that PBS can locate the file. For example, C:\winnt\temp\terminate.bat. The script

must be writable by no one but an Administrator-type account.

ARGS are optional arguments to the script. Values for *ARGS* may be: 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 ensure 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 timeout 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.

8.5.2 Site-Specific Job Checkpoint and Restart

The PBS Professional site-specific job checkpoint facility allows an Administrator to replace the built-in checkpoint facilities of PBS Professional with a site-defined external command. This is most useful on computer systems that do not have OS-level checkpointing. This feature is used by setting these MOM configuration parameters.

```
$action checkpoint        TIME_OUT !SCRIPT_PATH ARGS [...]
$action checkpoint_abort  TIME_OUT !SCRIPT_PATH ARGS [...]
$action restart           TIME_OUT !SCRIPT_PATH [ARGS ...]
```

The `checkpoint` parameter specifies that the script in `SCRIPT_PATH` is run, and the job is left running. This script is called once for each of the job's tasks, and is supplied by the site. The script must take care of everything necessary to checkpoint the job and restart it.

The `checkpoint_abort` parameter specifies that the script in `SCRIPT_PATH` is run, but the job is terminated. This script is called once for each of the job's tasks, and is supplied by the site. The script must handle everything necessary to checkpoint the job and restart it.

The `restart` parameter specifies the script to be used to restart the job. This script is called once for each of the job's tasks, and is supplied by the site. When the job is restarted, it will be running on the same machine as before, with the same priority.

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 performed.

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

PBS uses the following MOM configuration parameters to control how restart scripts are run. See “$restart_background <true|false>” on page 266 and “$restart_transmogrify <true|false>” on page 267.

```
$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.

8.5.3 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 vnodes 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.

Note: checkpointing is not supported for job arrays. On systems that support checkpointing, subjobs are not checkpointed; instead they run to completion.

8.6 Configuring Idle Workstation Cycle Harvesting

"Harvesting" of idle workstation cycles is a method of expanding the available computing resources of your site by automatically including in your complex unused workstations that otherwise would have sat idle. This is particularly useful for sites that have a significant number of workstations 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. Detection of "usage" can be configured to be based upon system load average or by keystroke activity (as discussed in

the following two sections below). Furthermore, cycle harvesting can be configured for all jobs, single-vnode jobs only, and/or with special treatment for multi-vnode (parallel) jobs. See section 8.6.4 “Cycle Harvesting: Serial vs Parallel Jobs” on page 280 for details.

8.6.1 Cycle Harvesting Based on Load Average

Cycle harvesting based on load average is load balancing based on load average. You set each workstation’s max_load and ideal_load. When max_load is exceeded, the node is marked as "state=busy". It will show up this way in pbsnodes and the scheduler will not place jobs on busy nodes. When the load drops below ideal_load, the state changes back to "free".

To set up cycle harvesting of idle workstations based on load average, perform the following steps:

Step 1 If PBS is not already installed on the target execution workstations, do so now, selecting the execution-only install option. (See Chapter 4 of this manual for details.)

Step 2 Edit the `PBS_HOME/mom_priv/config` configuration file on each target execution workstation, adding the two load-specific configuration parameters with values appropriate to your site.

```
$max_load 5
$ideal_load 3
```

Then HUP the MOM:

```
kill -HUP <pbs_mom PID>
```

Step 3 Edit the `PBS_HOME/sched_priv/sched_config` configuration file to direct the Scheduler to perform scheduling based on `load_balancing`.

```
load_balancing:  true   ALL
```

It is also recommended to remove the `ncpus` entry from the Scheduler `resources` parameter, in order to allow more jobs to run than there are CPUs available on the workstation.

Then HUP the scheduler:

```
kill -HUP <pbs_sched PID>
```

8.6.2 Cycle Harvesting Based on Keyboard/Mouse Activity

If a system is configured for keyboard/mouse-based 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 vnode 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".

Important: Jobs on workstations that become *busy* will not be migrated; they will remain on the workstation until they complete execution, are rerun, or are deleted.

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 8.6.3.
FreeBSD	unsupported	pbs_idled	See section 8.6.3.
HP-UX 10 and 11	supported	device	See below
IRIX	supported	pbs_idled	See section 8.6.3.
Linux	supported	device	See below
Solaris	supported	device	See below
Tru64	supported	pbs_idled	See section 8.6.3.
Windows XP Pro	supported	other	See below
Windows 2003 Server	supported	other	See below
Windows 2000 Pro	supported	other	See below

System	Status	Method	Reference
Windows 2000 Server	supported	other	See below

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_wait [ min_use [ poll_interval ] ]
```

These three parameters, representing time specified in seconds, control the transitions between *free* and *busy* states. Definitions follow.

idle_wait	time (in seconds) that the workstation keyboard and mouse must be idle before the workstation becomes available to PBS.
min_use	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.
poll_interval	frequency of checking the state of the keyboard and mouse.

After changing each MOM's configuration file, HUP the MOM:

```
kill -HUP <pbs_mom PID>
```

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 *min_use* is 10 seconds, the default for *poll_interval* is 1 second. There is no default for *idle_wait*; setting it to non-zero is required to activate the cycle harvesting feature.

Elaborating on the above example will help clarify the role of the various times. Let's 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 (idle poll period), `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 either *free* (if any CPUs are available) or *job-busy* (if all CPUs are in use.) 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.

8.6.3 Cycle Harvesting on Machines with X-Windows

On some systems cycle harvesting is simple to implement as the console, keyboard, and mouse device access times are updated by the operating system periodically. The PBS MOM process takes note of that and marks the vnode busy if any of the input devices are in use. On other systems, however, this data is not available. (See table in section 8.6.2 above.) 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 process called `pbs_idled`. This program runs in the background and monitors X and reports to the `pbs_mom` whether the vnode is idle or not.

Because of X-Windows security, running `pbs_idled` requires more modification than 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 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.

8.6.4 Cycle Harvesting: Serial vs Parallel Jobs

Given local usage policy constraints, and the possible performance impact of running certain applications on desktop systems, a site may need to limit the usage of cycle harvesting to a subset of jobs. The most common restriction is on the use of multi-vnode jobs.

A site may wish to enable cycle harvesting, but only for single-vnode 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 210.

When a job is running on a workstation configured for cycle harvesting, and that vnode becomes "busy", the job is suspended. However, suspending a multi-vnode parallel job may have undesirable side effects because of the inter-process communications. Thus the default action for a job which uses multiple vnodes when one or more of the vnodes becomes busy, is to leave the job running.

It is possible, however, to specify that the job should be requeued (and subsequently re-scheduled to run elsewhere) when any of the vnodes on which the job is running becomes *busy*. To enable this action, the Administrator must add the following parameter 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 vnode becomes busy.

8.6.5 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`) within the job script, and such a command is running when PBS decides to suspend the job on the vnode, then the file transfer will be suspended as well.

However, if the job has PBS file staging parameters (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 "`Exiting`" state, after the `epilogue` is run), and is not subject to suspension. (For more information on PBS file staging, see the **PBS Professional User's Guide**.)

8.6.6 Cycle Harvesting on Windows

Under Windows, when a machine becomes “busy” because the keyboard is being used, the effect on the job is different. Instead of being suspended, the job has its priority lowered from Normal to Low. For example, you submit a job and it begins to run on a workstation, and the CPU loading on that machine goes to 100%. Then you move the mouse: you’ll see that the CPU loading is still 100%. This is because the job has lower priority, but is not suspended. If you use qstat, you’ll see that the job’s state is “U”, because PBS has marked the job as “suspended”. Local activity on the machine will have higher priority.

8.7 Restricting User Access to Execution Hosts

PBS provides a facility to prevent users from using machines controlled by PBS except by submitting jobs. You can turn this feature on using the $restrict_user MOM directive. This uses the $restrict_user_exceptions and $restrict_user_maxsysid directives. This can be set up vnode by vnode so that a user requesting exclusive access to a set of vnodes will be guaranteed that no other user will be able to use the nodes assigned to his job, or a user requesting non-exclusive access to a set of nodes will be guaranteed that no access will be allowed to the nodes except through PBS. Also, a privileged user can be allowed access to the complex such that they can login to a vnode without having a job active, or an abusive user can be denied access to the complex nodes. The administrator can find out when users try to circumvent a policy of using PBS to access nodes. In addition, you can ensure that application timings will be reproducible on a complex controlled by PBS. The log level for messages concerning restricting users is PBSE_SYSTEM (0002).

For a vnode with access restriction turned on:

Any user not running a job who logs in or otherwise starts a process on that vnode will have his processes terminated.

A user who has logged into a vnode where he owns a job will have his login terminated when the job is finished.

When MOM detects that a user that is not exempt from access restriction is using the system, that user's processes are killed and a log message is output:

```
01/16/2006 22:50:16;0002;pbs_mom;Svr;restrict_user; \
killed uid 1001 pid 13397(bash)
```

with logging level PBSE_SYSTEM.

You can set up a list of users who are exempted from the restriction via the $restrict_user_exceptions directive. This list can contain up to 10 user names.

Examples:

Turn access restriction on for a given node:

```
$restrict_user on
```

Limit the users affected to those with a user ID greater than 500:

```
$restrict_user_maxsysid 500
```

Exempt specific users from the restriction:

```
$restrict_user_exceptions userA, userB, userC
```

8.8 Resource Limit Enforcement

You may wish to prevent jobs from swapping memory. To prevent this, you can set limits on the amount of memory a job can use. Then the job must request an amount of memory equal to or smaller than the amount of physical memory available.

PBS measures and enforces memory limits in two ways: on each host, by setting OS-level limits (using the limit system calls), and by periodically summing the usage recorded in the /proc entries. Note: enforcement is (1) site optional (one must add "$enforce mem" to the MOM's config file), and (2) only happens if the job requests a limit (via "mem=..." in the qsub parameters).

Job resource limits can be enforced for single-vnode jobs, or for multi-vnode jobs using LAM or a PBS-aware MPI. See the following table for an overview. Memory limits are handled differently depending on the operating system; see "Job Memory Limit Enforcement on UNIX" on page 284. The ncpus limit can be adjusted in several ways; for a discussion see "Job NCPUS Limit Enforcement" on page 286.

Table 13: Resource Limit Enforcement

Limit	What determines when limit is enforced	Scope of limit	Enforcement method
file size	automatically	per-process	setrlimit()

Table 13: Resource Limit Enforcement

Limit	What determines when limit is enforced	Scope of limit	Enforcement method
pvmem	automatically	per-process	setrlimit()
pmem	automatically	per-process	setrlimit()
pcput	automatically	per-process	setrlimit()
cput	automatically	job-wide	MOM poll
walltime	automatically	job-wide	MOM poll
mem	if `$enforce mem` in MOM's config	job-wide	MOM poll
ncpus	if `$enforce cpuaverage`, `$enforce cpuburst`, or both, in MOM's config. See "Job NCPUS Limit Enforcement" on page 286.	job-wide	MOM poll

8.8.1 Job Memory Limit Enforcement on UNIX

Enforcement of `mem` resource usage is available on all UNIX platforms, but not Windows. To enforce `mem` resource usage, put `$enforce mem` into MOM's config file. Enforcement is off by default.

The `mem` resource can be enforced at both the job level and the vnode level. The job level will be the smaller of a job-wide resource request and the sum of that for all chunks. The vnode level is the sum for all chunks on that node.

Job-wide limits are enforced by MOM polling the working set size of all processes in the job's session. Jobs that exceed their specified amount of physical memory are killed. A job may exceed its limit for the period between two polling cycles. See "Configuring MOM's Polling Cycle" on page 270.

Per-process limits are enforced by the operating system kernel. PBS calls the kernel call setrlimit() to set the limit for the top process (the shell) and any process started by the shell inherits those limits.

If a user submits a job with a job limit, but not per-process limits (qsub -l cput=10:00) then PBS sets the per-process limit to the same value. If a user submits a job with both job and per-process limits, then the per-process limit is set to the lesser of the two values.

Example: a job is submitted with qsub -lcput=10:00

a) There are two CPU-intensive processes which use 5:01 each.
The job will be killed by PBS for exceeding the cput limit.
5:01 + 5:01 is greater than 10:00.

b) There is one CPU-intensive process which uses 10:01.
It is very likely that the kernel will detect it first.

c) There is one process that uses 0:02 and another that uses 10:00.
PBS may or may not catch it before the kernel does depending on exactly when the polling takes place.

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.

The following table shows which OS resource limits can be used by each operating system.

Table 14: RLIMIT Usage in PBS Professional

OS	file	mem/pmem	vmem/pvmem	cput/pcput
AIX	RLIMIT_FSIZE	RLIMIT_RSS	RLIMIT_DATA RLIMIT_STACK	RLIMIT_CPU
HP-UX	RLIMIT_FSIZE	RLIMIT_RSS	RLIMIT_AS	RLIMIT_CPU
IRIX	RLIMIT_FSIZE	RLIMIT_RSS	RLIMIT_VMEM	RLIMIT_CPU
Linux	RLIMIT_FSIZE	RLIMIT_RSS	RLIMIT_AS	RLIMIT_CPU
MacOS	RLIMIT_FSIZE	RLIMIT_RSS	RLIMIT_DATA RLIMIT_STACK	RLIMIT_CPU
SunOS	RLIMIT_FSIZE	RLIMIT_DATA RLIMIT_STACK	RLIMIT_VMEM	RLIMIT_CPU

Table 14: RLIMIT Usage in PBS Professional

OS	file	mem/pmem	vmem/pvmem	cput/pcput
Super-UX	RLIMIT_FSIZE	RLIMIT_UMEM RLIMIT_DATA RLIMIT_STACK	ignored	RLIMIT_CPU
Tru64	RLIMIT_FSIZE	RLIMIT_RSS	RLIMIT_VMEM	RLIMIT_CPU

For mem/pmem, the limit is set to the smaller of the two. For vmem/pvmem, the limit is set to the smaller of the two. Note that RLIMIT_RSS, RLIMIT_UMEM, and RLIMIT_VMEM are not standardized (i.e. do not appear in the The Open Group Base Specifications Issue 6).

8.8.1.1 Sun Solaris-specific Memory Enforcement

Solaris does not support `RLIMIT_RSS`, but instead has `RLIMIT_DATA` and `RLIMIT_STACK`, which are hard limits. On Solaris or another Open Group standards-compliant OS, a `malloc`() call that exceeds the limit will return `NULL`. This behavior is different from other operating systems and may result in the program (such as a user's application) receiving a `SIGSEGV` signal.

8.8.1.2 Memory Enforcement on cpusets

There should be no need to do so: either the vnode containing the memory in question has been allocated exclusively (in which case no other job will also be allocated this vnode, hence this memory) or the vnode is shareable (in which case using mem_exclusive would prevent two CPU sets from sharing the memory). Essentially, PBS enforces the equivalent of mem_exclusive by itself.

8.8.2 Job NCPUS Limit Enforcement

Enforcement of the `ncpus` limit (number of CPUs used) is available on all platforms. The `ncpus` limit can be enforced using average CPU usage, burst CPU usage, or both. By default, enforcement of the ncpus limit is off. See "$enforce <limit>" on page 263.

8.8.2.1 Average CPU Usage Enforcement

To enforce average CPU usage, put "`$enforce cpuaverage`" in MOM's config file. You can set the values of three variables to control how the average is enforced. These are shown in the following table.

Table 15: Variables Used in Average CPU Usage

Variable	Type	Description	Default
cpuaverage	Boolean	If present (=true), MOM enforces ncpus when the average CPU usage over the job's lifetime usage is greater than the specified limit.	false
average_trialperiod	integer	Modifies cpuaverage. Minimum job walltime before enforcement begins. Seconds.	120
average_percent_over	integer	Modifies cpuaverage. Percentage by which the job may exceed ncpus limit.	50
average_cpufactor	float	Modifies cpuaverage. ncpus limit is multiplied by this factor to produce actual limit.	1.025

Enforcement of cpuaverage is based on the polled sum of CPU time for all processes in the job. The limit is checked each poll period. Enforcement begins after the job has had `average_trialperiod` seconds of walltime. Then, the job is killed if the following is true:

(cput / walltime) > (ncpus * average_cpufactor + average_percent_over / 100)

8.8.2.2 CPU Burst Usage Enforcement

To enforce burst CPU usage, put "`$enforce cpuburst`" in MOM's config file. You can set the values of four variables to control how the burst usage is enforced. These are shown in the following table.

Table 16: Variables Used in CPU Burst

Variable	Type	Description	Default
cpuburst	Boolean	If present (=true), MOM enforces ncpus when CPU burst usage exceeds specified limit.	false
delta_percent_over	integer	Modifies cpuburst. Percentage over limit to be allowed.	50
delta_cpufactor	float	Modifies cpuburst. ncpus limit is multiplied by this factor to produce actual limit.	1.5
delta_weightup	float	Modifies cpuburst. Weighting factor for smoothing burst usage when average is increasing.	0.4
delta_weightdown	float	Modifies cpuburst. Weighting factor for smoothing burst usage when average is decreasing.	0.1

MOM calculates an integer value called `cpupercent` each polling cycle. This is a moving weighted average of CPU usage for the cycle, given as 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 that during the period, the job used an average of three CPUs.

```
new_percent = change_in_cpu_time*100 / 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 `new_percent` is higher than the old `cpupercent` value. `delta_weight_down` is used if `new_percent` is lower than the old `cpupercent` value. `delta_weight_[up|down]` controls the speed with which `cpu-`

percent changes. If delta_weight_[up|down] is 0.0, the value for cpupercent does not change over time. If it is 1.0, cpupercent will take the value of new_percent for the poll period. In this case cpupercent changes quickly.

However, cpupercent is controlled so that it stays at the greater of the average over the entire run or ncpus*100.

max_poll_period is the maximum time between samples, set in MOM's config file by $max_check_poll, with a default of 120 seconds.

The job is killed if the following is true:

new_cpupercent > ((ncpus * 100 * delta_cpufactor) + delta_percent_over)

The following entries in MOM's config file turns 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
```

Cpuburst and cpuaverage information show up in MOM's log file, whether or not they has been configured in mom_config. This is so a site can test different parameters for cpuburst/cpuaverage before enabling enforcement. You can see the effect of any change to the parameters on your job mix before "going live".

8.8.2.3 SGI IRIX 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.

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.

When PBS kills a process, PBS will send the signal to all the processes in the job about which it knows. It will signal bottom up, i.e. the child process is signalled before its parent, so that the last process signalled is the top level shell.

For a multiple vnode job, the above order is repeated on each vnode. The order in which the vnodes perform the above sequence is indeterminate due to network latency and other activity on the various MOMs. For a job running on vnodes A, B, and C, you cannot predict if the processes on A will be killed before the processes on C or B.

When a process (and therefore its session) are attached to a job via pbs_attach, the MOM logs the the pid, sid, job ID and task ID of the attached process. The log message is of the form:

```
pid XXX sid YYYY cmd SSSSS attached as task XNNNN
```

PBS can only kill those processes about which it knows. If the MPI is not integrated with PBS, PBS will not know about the processes started via MPI and will not kill them.

8.9 Configuring MOM for Machines with cpusets

There is an enhanced PBS MOM called pbs_mom.cpuset which is designed to manage a machine with cpusets. Using cpusets on the Altix requires the SGI ProPack library. See SGI's documentation for more information. The standard PBS MOM can also manage a machine with cpusets, but PBS and the jobs it manages will not create or otherwise make use of them.

8.9.0.1 Vnodes and cpusets

A cpuset is a list of CPUs and memory nodes managed by the OS. Processes executing within a cpuset are typically confined to use only the resources defined by the set. An Altix using `pbs_mom.cpuset` will present multiple vnodes to its server; these in turn are visible when using commands such as `pbsnodes`. Each of these vnodes is being managed by the one instance of `pbs_mom.cpuset`. An IRIX machine using `pbs_mom.cpuset` will present a single vnode.

8.9.1 Configuration Files for Multi-vnoded Machines

PBS uses three kinds of configuration files: the default configuration file described in "Syntax and Contents of Default Configuration File" on page 260, PBS reserved configuration files, which are created by PBS, and site-defined configuration files, described in "Syntax of Version 2 PBS Reserved Configuration Files" on page 292.

The default configuration file lists MOM resources and initialization values. To change this file, you edit it directly.

Site-defined configuration files are used to make site-specific changes in vnode configuration. Instead of editing these directly, you create a local file and give it as an argument to the `pbs_mom -s insert` option, and PBS creates a new configuration file for you. See "Creation of Site-defined MOM Configuration Files" on page 259. Their syntax is

called “version 2” in order to differentiate it from the syntax of the default configuration files. You can also remove a site-defined configuration file using the `pbs_mom -s remove` option.

PBS reserved files contain vnode configuration information. These are created by PBS. Any attempt to operate on them will result in an error.

You can list and view the PBS reserved configuration files and the site-defined configuration files using the `pbs_mom -s list` and `pbs_mom -s show` options.

Do not mix the configuration files or the syntax. Each type must use its own syntax, and contain its own type of information.

8.9.1.1 Creation of PBS Reserved Configuration Files

Any PBS reserved MOM configuration files are only created when PBS is started via the pbs start/stop script, not when the MOM is started with the pbs_mom command. Therefore, if you make changes to the hardware or a change occurs in the number of CPUs or amount of memory that is available to PBS, such as a non-PBS process releasing a cpuset, you should restart PBS, by typing “<path-to-script>/pbs start”, in order to re-create the PBS reserved MOM configuration files. The MOM daemon will normally be started by the PBS start/stop script.

8.9.1.2 Syntax of Version 2 PBS Reserved Configuration Files

These configuration files contain the configuration information for vnodes, including the resources available on those vnodes. They do not contain initialization values for MOM. The resources described in these configuration files can be set via `qmgr` and can be viewed using `pbsnodes -av`.

PBS reserved configuration files and site-defined configuration files use this syntax. Do not use this syntax for the default configuration file, and do not use the default configuration file’s syntax to describe vnode information. For information about vnodes, see section 7.7 “Vnodes: Virtual Nodes” on page 205.

Any configuration file containing vnode-specific assignments must begin with this line:

```
$configversion 2
```

The format a file containing vnode information is:

```
<ID> : <ATTRNAME> = <ATTRVAL>
```

where

<ID> sequence of characters not including a colon (":")

<ATTRNAME>sequence of characters beginning with alphabetics or numerics, which can contain underscore ("_") and dash ("-")

<ATTRVAL> sequence of characters not including an equal sign ("=")

The colon and equal sign may be surrounded by white space.

A vnode's ID is an identifier that will be unique across all vnodes known to a given pbs_server and will be stable across reinitializations or invocations of pbs_mom. ID stability is important when a vnode's CPUs or memory might change over time and PBS is expected to adapt to such changes by resuming suspended jobs on the same vnodes to which they were originally assigned. Vnodes for which this is not a consideration may simply use IDs of the form "0", "1", etc. concatenated with some identifier that ensures uniqueness across the vnodes served by the pbs_server. Vnode attributes cannot be used as vnode names. Vnode attributes are listed in section 7.8 "Vnode Configuration Attributes" on page 210.

8.10 Configuring MOM on an Altix

The configuration information for the Altix in this book is in three sections. The information common to all MOMs applies to the Altix; see section 8.2 "MOM Configuration Files" on page 258. The information common to ProPack 2, 3, 4 and 5 also applies; see "Static Resources for Altix Running ProPack 2 or Greater" on page 298 and "Initialization Values for Altix Running ProPack 2 or Greater" on page 299. Last, there are separate sections specific to ProPack 2/3 and to ProPack 4/5.

To verify which CPUs are included in a cpuset created by PBS, on ProPack 4/5, use:

```
cpuset -d <set name> | egrep cpus
```

This will work either from within a job or not.

The alt_id returned by MOM has the form `cpuset=<name>`. <name> is the name of the cpuset, which is the $PBS_JOBID.

A cpusetted machine can have a "boot cpuset" defined by the administrator. A boot cpuset contains one or more CPUs and memory boards and is used to restrict the default placement of system processes, including login. If defined, the boot cpuset will contain CPU 0.

By default, the PBS MOM will not use the boot cpuset. The CPUSET_CPU_EXCLUSIVE flag prevents CPU 0 from being used by the MOM in the creation of job cpusets. This flag is set by default.

The MOM excludes from its use all CPUs in sets not belonging to PBS. The way to reserve some for other uses is to create a boot CPU set.

In order to use `pbs_mom.cpuset` on an Altix, you will need a vnode definitions file, which contains all the information about the machine's vnodes and their resources. This is used by PBS for scheduling jobs. Each Altix may have a different topology, depending on how it is wired. The PBS startup script creates the vnode definitions file for ProPack 4 and greater if it detects that pbs_mom.cpuset has been copied to pbs_mom.

The cpuset hierarchy has changed for version 8.0 and later. There are no directories under /PBSPro for shared or suspended cpusets.

On a suspend request, the cpuset MOM will move the processes to the global cpuset, then restore them later upon restart.

When PBS Professional creates job cpusets, it does not set the CPU or memory exclusive flags. PBS manages the exclusivity on these cpusets.

8.10.1 Configuring MOM for an Altix Running ProPack 4/5

On an Altix running ProPack 4/5, the vnode definitions file is generated automatically by PBS. The MOM includes routers automatically when she generates the file. There is a script which can be modified to produce different vnode definitions. The script is $PBS_EXEC/lib/init.d/sgigenvnodelist.awk. This script is designed to be modified by the PBS administrator. It is an alternative to using pbs_mom -s to insert changed vnode definitions.

8.10.2 Configuring MOM for an Altix Running ProPack 2/3

8.10.2.1 CPU 0 Allocation with cpusets for an Altix Running ProPack 2/3

MOM does not use the CPUs on any nodeboard containing either CPU 0 or a CPU which was in use at startup.

8.10.2.2 Vnode Definitions File

If you wish to use cpusets you must have a vnode definitions file. The vnode definitions file is not automatically generated for ProPack 2/3. This file must be generated for your system; you can generate it, or you can contact support for help. See "Technical Support" on page ii.

The format of the file is described in section 8.9.1.2 "Syntax of Version 2 PBS Reserved Configuration Files" on page 292. An example file would look like this:

First, a preamble of the form

```
$configversion 2

AltixHostName:  pnames = <placement set types list>
AltixHostName:  sharing = ignore_excl
AltixHostName:  resources_available.ncpus = 0
AltixHostName:  resources_available.mem = 0
AltixHostName:  resources_available.vmem = 0
```

where <placement set types list> is a list of the placement set type names that will be referred to in subsequent resource definitions.

For each vnode (e.g. C-brick, blade)

```
AltixHostVnodeName:  sharing = default_excl
AltixHostVnodeName:  resources_available.ncpus = \
<number>
AltixHostVnodeName:  cpus = <CPU list>
AltixHostVnodeName:  mems = <number>
AltixHostVnodeName:  resources_available.mem = \
<memory amount>
AltixHostVnodeName:  resources_available.<pstype> = \
<psname>
```

where for this vnode,

<number> is a non-negative integer (number of CPUs or memory board number

<CPU list> is the list of CPUs (may be a comma- or dash-separated list)

<memory amount> is the amount of memory (in KB), suffixed by the string "kb"

for each placement set type of which this vnode is a member, there is a line of the form
<name>:resources_available.<pstype> = <psname>

<name> is the vnode's name, <pstype> is the placement set type, and <psname> is the uniquely-named placement set.

See SGI's documentation on generating topology information and SGI's topology(1) man page.

8.10.2.3 Generating Vnode Definitions File for ProPack 2/3

If the Altix is running ProPack 2 or 3, generate a vnode definitions file for it. Support can help you create a preliminary file. See section "Technical Support" on page ii.

1 Create the preliminary file `prelim_defs` with the help of the technical support group.

2 Add the definition of the natural vnode to `prelim_defs`. See section 7.7.2 "Natural Vnodes" on page 206.

3 Set the amount of memory on each vnode via `prelim_defs`.

3a Find the number of pages per node:

hinv -v -c memory

This will give you a list of nodes and pages per node:

```
Node   Pages
 0     248836
 1     250880
 2     250880
 3     250880
 4     250880
 5     250880
 6     504831
 7     504831
 8     504832
 9     504832
10     504832
```

11 503671

3b Look in `/proc/meminfo` for the value of MemTotal. Use this value for main memory size:

```
cat /proc/meminfo

MemTotal:     72058142 kB
```

3c Calculate the amount of memory per vnode:

(main mem / total # pages) * (pages / node) = mem/vnode

If we use 72058142kB as the main memory size for our example, then for Vnode0 in the example above, we would have:

(72058142kB / 4531065 total pages) * (248836) = 3957272kB

3d Set the amount of memory on each vnode. For each vnode, add a line of this form to `prelim_defs`:

```
<vnodename> resources_available.mem = \
    <MEM>
```

4 Define the placement sets you want via the `pnames` attribute. Add a line of this form to `prelim_defs`:

```
<natural vnode name> \
    pnames=<RESOURCE[,RESOURCE ...]
```

See section 9.6.9.2 "Examples of Configuring Placement Sets on an Altix" on page 334.

5 Use `pbs_mom -s insert` to create `scriptname` from `prelim_defs` and add it to the configuration files. See the section "-s script_options" on page 411 for `pbs_mom`.

```
pbs_mom -s insert <scriptname> \
    <prelim_defs>
```

6 Have MOM re-read her configuration files:

```
pkill -HUP pbs_mom
```

8.10.3 Altix-Specific Configuration Parameters in Default MOM Configuration File

8.10.3.1 Static Resources for Altix Running ProPack 4 or 5

`cpuset_create_flags <flags>`
CPUSET_CPU_EXCLUSIVE | 0
Default: CPUSET_CPU_EXCLUSIVE

8.10.3.2 Static Resources for Altix Running ProPack 2 or 3

`cpuset_create_flags <flags>`
CPUSET_CPU_EXCLUSIVE
CPUSET_MEMORY_LOCAL
CPUSET_MEMORY_EXCLUSIVE
CPUSET_MEMORY_MANDATORY|
CPUSET_POLICY_KILL|
CPUSET_EVENT_NOTIFY
CPUSET_KERNEL_AVOID
See SGI's documentation on cpusetCreate(3x).

Default: CPUSET_CPU_EXCLUSIVE|
CPUSET_MEMORY_LOCAL|
CPUSET_MEMORY_EXCLUSIVE|
CPUSET_MEMORY_MANDATORY|
CPUSET_POLICY_KILL|
CPUSET_EVENT_NOTIFY

8.10.3.3 Static Resources for Altix Running ProPack 2 or Greater

`cpuset_destroy_delay <delay>`
MOM will wait `delay` seconds before destroying a cpuset of a just-completed job. This allows processes time to finish.
Default: 0. Integer. For example,
cpuset_destroy_delay 10

8.10.3.4 Initialization Values for Altix Running ProPack 2 or Greater

`pbs_accounting_workload_mgmt <value>`
: Controls whether CSA accounting is enabled. The name does not start with a dollar sign. If set to "1", "on", or "true", CSA accounting is enabled. If set to "0", "off", or "false", CSA accounting is disabled. Values are case-insensitive. Default: "true"; enabled.

8.10.3.5 Switching From Standard MOM to Cpusetted MOM on Altix

If you switch from the standard MOM to the cpusetted MOM, you'll need to create a modified vnode definitions file with any changes that you made previously via qmgr. Use the `pbs_mom -s insert` command to add it. You'll also need to unset any ncpus, mem, vmem and sharing values you've added with qmgr. For example,

```
Qmgr: u n a450-2 resources_available.ncpus
```

Then stop and start the mom to get the changes to take effect.

8.10.3.6 Switching From Cpusetted MOM to Standard MOM on Altix

If you switch from the cpusetted MOM to the standard MOM on the Altix, you'll need to remove any vnode definition files you added that contain information dependent on the automatically-generated ones.

Remove your own vnode definitions files. List them:

```
pbs_mom -s list
```

Remove each file you added:

```
pbs_mom -s remove <scriptname>
```

Add new configuration files with any information you need:

```
pbs_mom -s insert <new scriptname>
```

Then stop and start the mom to get the changes to take effect.

8.10.4 Configuring MOM for Comprehensive System Accounting

8.10.4.1 Requirements for CSA

Using CSA requires the version of pbs_mom.cpuset that is built with CSA enabled. CSA can be used on SGI Altix machines running SGI's ProPack 2.4 or greater, and having library (not system) call interfaces to the kernel's job container and CSA facilities. Both the Linux job container facility and CSA support must either be built into the kernel or available as loadable modules.

For information on getting Linux job container software configured and functioning, go to http://www.ciemat.es/informatica/gsc/perfdoc/007-4413-003/sgi_html/index.html and see "Linux Resource Administration Guide", subsection "Linux Kernel Jobs".

See the Release Notes for information on which versions of ProPack provide support for CSA with PBS.

If CSA is enabled, the PBS user can request the kernel to write user job accounting data to accounting records. These records can then be used to produce reports for the user. If workload management is enabled, the kernel will write workload management accounting records associated with the PBS job to the system-wide process accounting file. The default for this file is /var/csa/day/pacct.

There are two pbs_mom daemons for the Altix, one for cpusets and the standard daemon. The downloadable CSA-enabled PBS binaries for the Altix are built so that job container and CSA facilities are available in the kernel, so that both CSA user job accounting and CSA workload management accounting are available in both of the pbs_mom daemons.

In order for CSA user job accounting and workload management accounting requests to be acted on by the kernel, the administrator needs to make sure that the parameters CSA_START and WKMG_START in the /etc/csa.conf configuration file are set to "on" and that the system reflects this. You can check this by running the command:

```
csaswitch -c status
```

To set CSA_START to "on", use the command:

```
csaswitch -c on -n csa
```

To set WKMG_START to "on", use:

```
csaswitch -c on -n wkmg
```

Alternatively, you can use the CSA startup script /etc/init.d/csa with the desired argument (on/off) - see the system's manpage for csaswitch and how it is used in the /etc/init.d/csa startup script.

8.10.4.2 Configuration for CSA

If MOM is configured for CSA support, MOM can issue CSA workload management record requests to the kernel. To configure MOM for CSA support, modify $PBS_HOME/mom_priv/config, by adding a line for the parameter `pbs_accounting_workload_mgmt.` Set this parameter to "on"/"true"/"1" to enable CSA support, and "off"/"false"/"0" to disable it. If the parameter is absent, CSA support is enabled by default.

After modifying the MOM config file, either restart pbs_mom or send it SIGHUP.

For information on SGI Job Containers, see "SGI Job Container / Limits Support" on page 465.

8.10.5 Troubleshooting ProPack 4/5 cpusets

The ProPack4/5 cpuset-enabled mom may occasionally encounter errors during startup from which it cannot recover without help. If pbs_mom was started without the -p flag, one may see

```
"/PBSPro hierarchy cleanup failed in <dir> -
restart pbs_mom with '-p'"
```

where <dir> is one of /PBSPro, /PBSPro/shared, or /PBSPro/suspended. If this occurs, try restarting pbs_mom with the -p flag. If this succeeds, no further action will be necessary to fix this problem. However, it is possible that if pbs_mom is started with the -p flag, one may then see any of these messages:

```
"cpuset_query for / failed - manual intervention
is needed"
"/PBSPro query failed - manual intervention is needed"
"/PBSPro cpuset_getmems failed - manual intervention
is needed"
```

In this case, there is likely to be something wrong with the PBSPro cpuset hierarchy. First, use the `cpuset(1)` utility to test it:

```
# cpuset -s /PBSPro -r | while read set
 do
   cpuset -d $set > /dev/null
 done
```

If cpuset detects no problems, no output is expected. If a problem is seen, expect output of the form

```
cpuset </badset> query failed
/badset: Unknown error
```

In this case, try to remove the offending cpuset by hand, using the cpuset(1) utility,

```
# cpuset -x badset
cpuset <badset> removed.
```

This may fail because the named cpuset contains other cpusets, because tasks are still running attached to the named set, or other unanticipated reasons. If the set has subsets,

```
# cpuset -x nonempty
cpuset <nonempty> remove failed
/nonempty: Device or resource busy
```

first remove any CPU sets it contains:

```
# cpuset -s nonempty -r
/nonempty
/nonempty/subset
...

# cpuset -s nonempty -r | tac | while read set
do
   cpuset -x $set
done
...
cpuset </nonempty/subset> removed.
cpuset </nonempty> removed.
```

Note that output is previous output, reversed.

If the set has processes that are still attached,

```
# cpuset -x busy
cpuset <busy> remove failed
/busy: Device or resource busy
```

one can choose to either kill off the processes,

```
# kill `cpuset -p busy`
# cpuset -x busy
cpuset <busy> removed.
```

or wait for them to exit. In the latter case, be sure to restart pbs_mom using the -p flag to prevent it from terminating the running processes.

Finally, note that if removing a cpuset with cpuset -x should fail, one may also try to remove it with rmdir(1), provided one takes care to prepend the cpuset file system mount point first. For example,

```
# mount | egrep cpuset
cpuset on /dev/cpuset type cpuset (rw)
# find /dev/cpuset/nonempty -type d -print |
  tac | while read set
do
   rmdir $set
done
```

8.11 Configuring MOM for IRIX with cpusets

The pbs_mom for the irix6_cpuset architecture forks into two pbs_moms: one that services jobs, and one that gathers process information for every process that it tracks. It can fork an additional MOM for killing off stray or unauthorized processes. This MOM is turned off by default, but can be turned on by setting the "restrict_user" configuration file option to “on”.

If the cpuset MOM is used, PBS jobs can run on only one IRIX machine at a time. If the non-cpuset MOM is used, PBS jobs can run across multiple IRIX machines. However, the MOM will not be able to manage the cpusets.

On IRIX, the cpuset name is the first 8 characters of the job ID. If there is already a cpuset by that name, the last character in the name is replaced by a,b,c...z,A,...,Z until a unique name is found.

8.11.1 Small vs Multi-vnode Jobs and Shared vs. Exclusive cpusets

The irix6_cpuset pbs_mom classifies jobs as either small or multi-vnode. Small jobs use limited CPUs and memory, and run in shared cpusets, which are designated for small jobs. The definition of a small job is set using cpuset_small_ncpus and cpuset_small_mem in MOM's config file. These set the limits for how many CPUs and how much memory a small job can use. The default for small jobs is one CPU and the memory size of one nodeboard, which is system-dependent. The limit for the number of nodeboards used for shared cpusets is set in max_shared_nodes in MOM's config file. Once the last job using a shared cpuset exits or is suspended, the shared cpuset is cleared. There is no walltime associated with a shared cpuset.

Multi-vnode jobs use the resources of more than one nodeboard, and run in exclusive cpusets, by themselves. Furthermore, any job with the "ssinodes" attribute set will run in exclusive cpusets.

8.11.2 cpusets Used by PBS on IRIX

Mom will not use or remove any cpuset that is already in use when MOM starts up. This includes the boot cpuset, if it exists.

CPU 0 will only be allocated for a job if there is no boot cpuset and no other CPUs are available to satisfy a request. Use of CPU 0 for jobs can degrade performance, since the kernel uses this CPU heavily for system daemons.

8.11.3 IRIX-Specific Configuration Parameters in Default Configuration File

The irix6_cpuset MOM needs to have a uniform number of working CPUs in the nodeboards it manages. In MOM’s config file, set `minnodecpus` to the the minimum number of CPUs on a nodeboard. That way, if a CPU fails, that nodeboard will be removed from the scheduling pool.

8.11.3.1 Initialization Values for IRIX

$checkpoint_upgrade <value>

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. The <value> can be "1",

"true", "on", "0", "false", "off". Default: false. For details on use, see section 11.6.4 "Checkpointing Jobs Prior to SGI IRIX Upgrade" on page 424.

`$enforce complexmem`
: Specifies whether memory segments should be shared across jobs, as shown by getmemusage. If not set, shared segments count in their entirety against each job, as shown by ps. Only used with non-cpusetted IRIX.

8.11.3.2 Static Resources for IRIX

The following resources are IRIX-specific.

`alloc_nodes_greedy <0|1>`
: Determines whether MOM allocates nodeboards that are close together. A value of 1 means that MOM will allocate any nodeboard. Default: 1. For example,

```
alloc_nodes_greedy 0
```

`cpuset_create_flags <flags>`
: Lists the flags for when MOM does a cpusetCreate(3) for each job. flags is an or-ed list of flags. The flags are:
CPUSET_CPU_EXCLUSIVE
CPUSET_MEMORY_LOCAL
CPUSET_MEMORY_EXCLUSIVE
CPUSET_MEMORY_MANDATORY
CPUSET_MEMORY_KERNEL_AVOID
CPUSET_POLICY_KILL
CPUSET_POLICY_PAGE
CPUSET_POLICY_SHARE_WARN
CPUSET_POLICY_SHARE_FAIL

See SGI's documentation on cpusetCreate(3).

Default: CPUSET_CPU_EXCLUSIVE|
CPUSET_MEMORY_LOCAL|
CPUSET_MEMORY_EXCLUSIVE|
CPUSET_MEMORY_MANDATORY|
CPUSET_POLICY_KILL|
CPUSET_EVENT_NOTIFY

Note that the default flags must be overridden with a set that does NOT contain CPUSET_EVENT_NOTIFY.

`cpuset_destroy_delay <delay>`
MOM will wait delay seconds before issuing a cpusetDestroy(3) on the cpuset of a just-completed job. This allows processes time to finish. Default: 5. Integer. For example,

```
cpuset_destroy_delay 10
```

`cpuset_small_mem <mem>`
Defines the maximum amount of memory for a small job. Jobs requesting mem kilobytes of memory will be considered small, and will be assigned a shared cpuset. Default: the amount of memory on one nodeboard. For example,

```
cpuset_small_mem 1024
```

`cpuset_small_ncpus <num>`
Defines the maximum number of CPUs for a small job. Jobs requesting num or fewer will be considered small, and will be assigned a shared cpuset. Cannot exceed the number of CPUs on a nodeboard. Default: 1. For example,

```
cpuset_small_ncpus 2
```

`enforce <mem | !mem>`
Enforce or don't enforce each job's mem request. Default: enforced.

`enforce <pvmem | !pvmem>`
Enforce or don't enforce each job's pvmem request. Default: enforced.

`enforce <vmem | !vmem>`
Enforce or don't enforce each job's vmem request. Default: enforced.

`enforce <walltime | !walltime>`
Enforce or don't enforce each job's walltime request. Default: enforced.

`enforce <pcput | !pcput>`
Enforce or don't enforce each job's pcput request. Default: enforced.

`enforce <cput | !cput>`
Enforce or don't enforce each job's cput request. Default: enforced.

`enforce <cpupct | !cpupct>`
Enforce or don't enforce each job's cpupercent request. Default:

not enforced.

`enforce <file | !file>`
Enforce or don't enforce each job's file request. Default: enforced

`enforce <hammer | !hammer>`
Enforce or don't enforce the killing of processes of unauthorized users. Default: not enforced

`enforce <nokill | !nokill>`
Don't kill or kill the non-PBS processes if hammer code is enabled. Default: don't kill.

`enforce <cpusets | !cpusets>`
Enforce or don't enforce cpusets. Default: enforced.

`max_shared_nodes <vnodes>`
The maximum number of nodeboards that are allowed to be assigned to shared cpusets. Default: 2048. For example,

```
max_shared_nodes 64
```

`minnodemem <mem>`
Sets mem megabytes as the minimum amount of memory on a vnode to consider it for running jobs. MOM calculates that available memory for a job is (minnodemem - memreserved) MB. Default: smallest amount of memory found on any nodeboard. For example,

```
minnodemem 512
```

`minnodecpus <num>`
Sets num as the minimum number of working cpus on a vnode to consider it for running jobs. Default: smallest number of CPUs found on any nodeboard. Integer. For example,

```
minnodecpus 2
```

`schd_quantum <num>`
Sets num as the minimum number of nodeboards to be assigned to a job. Default: 1. Integer. For example,

```
schd_quantum 2
```

8.11.4 IRIX OS-Level Checkpoint With cpusets

MOM supports use of IRIX checkpointing features to allow the checkpointing and restart of jobs running within SGI cpusets. This requires SGI IRIX version 6.5.16 or later. See section 11.6 "Checkpoint / Restart Under PBS" on page 422.

8.11.5 Resource Reporting for cpusets

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 and system cpusets (e.g. `boot`). Be sure to `unset` any manual settings of `resources_available.{ncpus,mem}` in both the vnode and the Server to get this count automatically updated by MOM.

You may need to restrict PBS from using the entire system by reducing the number of cpus or the amount of memory available to jobs. You can do this by setting the value of resources_available.{mem,ncpus}. 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).

8.11.6 CPU 0 Allocation with cpusets

Some special vnode and CPU allocation rules are enforced by MOM on cpuset enabled systems. If `cpuset_create_flags` set during `cpusetCreate()` contains a flag for `CPUSET_CPU_EXCLUSIVE` then CPU 0 will not be allowed to be part of a cpuset. This is the default setting. (On an IRIX system, 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 vnode for system daemons. Usually, PBS with cpusets is used in conjunction with a boot cpuset which the system administrator creates which includes nodeboard 0. To use the default setting for `cpuset_create_flags` except that CPU 0 is to be used by PBS, the following can be added to MOM's config file (all on one line, without the "\"s):

```
cpuset_create_flags CPUSET_MEMORY_LOCAL|\
CPUSET_MEMORY_MANDATORY|\
CPUSET_MEMORY_EXCLUSIVE|\
CPUSET_POLICY_KILL|CPUSET_EVENT_NOTIFY
```

8.12 MOM Globus Configuration

For the optional Globus MOM, the same configuration mechanism applies as with the regular MOM except only three initiation value parameters are applicable: `$clienthost`, `$restricted`, `$logevent`. For details, see the description of these configuration parameters earlier in this chapter. Examples of different MOM configurations are included in Chapter 12 "Example Configurations" on page 521.

Chapter 9

Configuring the Scheduler

The Scheduler implements the local site policy determining 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.

9.1 Scheduling Policy

The scheduler runs just one scheduling policy, which you can define. You can define placement sets and user and group resource and job limits, etc. However, you cannot have two different scheduling policies on two different queues or partitions. Whatever is set in the scheduler's configuration file applies to all queues or partitions.

9.1.1 Default Scheduler Configuration

The 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. Specific system resources are checked to make sure they are available: `mem` (memory requested), `ncpus` (number of CPUs requested), `arch` (architecture requested), `host`, and `vnode`

(`cnode` on Blue Gene).

2. Queues are sorted into descending order using the queue `priority` attribute to determine the order in which jobs are to be considered. Jobs in the highest priority queue will be considered for execution before jobs from the next highest priority queue. If queues don't have different priority, queues are ordered randomly.

3. Jobs within queues of priority `preempt_queue_prio` (default 150) or higher will preempt jobs in lower priority queues.

4. The jobs within each queue are sorted into ascending order of requested CPU time (`cput`). The shortest job is placed first.

5. Jobs that have waited to run for the amount of time specified in `max_starve` are *starving*. `max_starve` defaults to 24 hours. Starving jobs are given higher priority.

6. Any queue whose name starts with "`ded`" is treated as a dedicated time queue (see discussion below). A sample dedicated time file (*`PBS_HOME`*`/sched_priv/dedicated_time`) is included in the installation.

7. Primetime is set to 6:00 AM - 5:30 PM. Any holiday is considered non-prime. Standard U.S. Federal holidays for the year 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 & vnode	`resources_available`	>= resources requested by job

Object	Attribute/Resource	Comparison
server, queue & vnode	`max_running`	>= number of jobs running
server, queue & vnode	`max_user_run`	>= number of jobs running for a user
server, queue & vnode	`max_group_run`	>= number of jobs running for a group
server & queue	`max_group_res`	>= usage of specified resource by group
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 and Soft Limits" on page 181) Not enabled by default.
server & queue	`max_user_run_soft`	>= maximum running jobs for a user (see "Hard and Soft Limits" on page 181) Not enabled by default.
server & queue	`max_group_res_soft`	>= usage of specified resource by group (see "Hard and Soft Limits" on page 181) Not enabled by default.
server & queue	`max_group_run_soft`	>= maximum running jobs for a group (see "Hard and Soft Limits" on page 181) Not enabled by default.
queue	`started`	= true
queue	`queue_type`	= execution
job	`job_state`	= queued / suspended

Object	Attribute/Resource	Comparison
node	`loadave`	Boolean in sched_config. Used with max_load and ideal_load. When the loadave is above max_load, that node is marked “busy”. The scheduler won’t place jobs on a node marked “busy”. When the loadave drops below ideal_load, the “busy” mark is removed. Consult your OS documentation to determine values that make sense. Default: not enabled.
node	`arch`	= type requested by job
node	`host`	= name requested by job

9.1.2 Jobs that Can Never Run

A job that can never run will sit in the queue until it becomes the most deserving job. Whenever this job is considered for being run, and backfilling is being used, the error message “resource request is impossible to solve: job will never run” is printed in the scheduler’s log file. The scheduler then examines the next job in line to be the most deserving job.

The scheduler only determines if a job will never run if backfilling is used. If backfilling is turned off, then the scheduler won't determine if a job will ever run or not. It just decides it can't run now.

9.2 New Scheduler Features

9.2.1 New Tunable Formula

The new server attribute “job_sort_formula” is used for sorting jobs according to a site-defined formula. See section 9.7.2 “Tunable Formula for Computing Job Priorities” on page 342.

9.2.2 Change to sched_config

The default job_sort_key of cput is commented out in the default sched_config file. It is left in as a usage example.

9.3 Scheduler Configuration Parameters

To tune the behavior of the 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 cannot 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 prime-time, non-primetime, 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 scheduling options, and the default values, are as follows.

backfill — Boolean. If this is set to "True", the scheduler will attempt to schedule smaller jobs around starving jobs and when using strict_ordering, as long as running the smaller jobs won't change the start time of the jobs they were scheduled around. The scheduler chooses jobs in the standard order, so other starving jobs will be considered first in the set to fit around the most starving job. For starving jobs, it only has an effect if the parameter "help_starving_jobs" is true. If backfill is "False", the scheduler will idle the system to run starving jobs. Can be used with strict_ordering.
Default: true all

backfill_prime — boolean: Directs the Scheduler not to run jobs which will overlap the boundary between primetime and non-primetime. This assures that jobs restricted to running in either primetime or non-primetime can start as soon as the time boundary happens. See also prime_spill, prime_exempt_anytime_queues.
Default: false all

by_queue
: boolean. If set to true, jobs are run first from the first queue until that queue is empty, then the next queue, and so on. If `sort_queues` is set to true, queues are ordered highest-priority first. If `by_queue` is set to false, all jobs are treated as if they are in one large queue. The `by_queue` attribute is overridden by the `round_robin` attribute when `round_robin` is set to true. See section 9.8 “How Queues are Ordered” on page 345.

  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 9.9 “Defining Dedicated Time” on page 346.
  Default: `ded`

fair_share
: boolean: This will enable the fairshare 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 9.15 “Using Fairshare” on page 354.
  Default: `false all`

fairshare_entity
: string: Specifies the “entity” for which fairshare usage data will be collected. Can be “euser”, “egroup”, “Account_Name”, or “queue”, or egroup:euser.)
  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 9.15.5 “Tracking Resource Usage” on page 358. A special case resource is the exact string “ncpus*walltime”. The number of cpus used is multiplied by the walltime in seconds used by the job to determine the usage.
  Default: “`cput`”.

half_life
: time: The half life for fairshare usage; after the amount of time specified, the fairshare usage will be halved. Requires that `fair_share` be enabled. See also section 9.15 "Using Fairshare" on page 354.
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 lower-priority 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. `job_sort_key` can be used to sort by either resources or by special case sorting routines. Multiple `job_sort_key` entries can be used, in which case the first entry will be the primary sort key, the second will be used to sort equivalent items from the first sort, etc. The `HIGH` option implies descending sorting, `LOW` implies ascending. See example for details.

  This attribute is overridden by the `job_sort_formula` attribute. If both are set, `job_sort_key` is ignored and an error message is printed.

  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 9.16 "Enabling Strict Priority" on page 361
`job_priority HIGH|LOW`	Sort jobs by the job `priority` attribute regardless of job owner. (The `priority` attribute can be set during job submission via the "`-p`" option to the `qsub` command, as discussed in the **PBS Professional User's Guide**.)
`preempt_priority HIGH`	Sort jobs by preemption priority. Recommended that this be used when soft user limits are used. Also recommended that this be the primary sort key.
`sort_priority HIGH|LOW`	**Deprecated**. See `job_priority`, above.

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 of single-host jobs across a complex. The load balancing takes into consideration the load on each host as well as all resources specified in the "resource" list. See `smp_cluster_dist`, and section 9.12 "Enabling Load Balancing" on page 350. Load balancing can result in overloaded CPUs.
Default: `false all`

load_balancing_rr — Deprecated. To duplicate this setting, enable `load_balancing` and set `smp_cluster_dist` to `round_robin`. See also section 9.12 "Enabling Load Balancing" on page 350.

log_filter — integer: Defines which event types to keep out of the scheduler's 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 11.17 "Use and Maintenance of Logfiles" on page 480.
Default: `1280` (DEBUG2 & DEBUG3)

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 query the MOMs to set the value of `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 vnode. On a multi-vnoded machine with a natural vnode, all vnodes will share anything set in `mom_resources`.

node_sort_key — string: Defines sorting on resource values on vnodes. Resource must be numerical, for example, `long` or `float`.
Syntax:

```
node_sort_key: "<resource>|job_priority \
    HIGH|LOW"
node_sort_key: "<resource> HIGH|LOW \
    total|assigned|unused"
```

"total": Use the resources_available value.
"assigned": Use the resources_assigned value.
"unused": Use the value given by resources_available - resources_assigned.
See section 9.6.8.1 "Sorting Vnodes with node_sort_key" on page 331.
Note that up to 20 `node_sort_key` entries can be used, in which case the first entry will be the primary sort key, the second will be used to sort equivalent items from the first sort, etc.

Default:
```
node_sort_key: "job_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 non-primetime are defined in the `holidays` file. See also "Defining Primetime and Holidays" on page 346.
Default: `np_`

peer_queue — string: Defines the mapping of a remote queue to a local queue for Peer Scheduling. Maximum number is 50 peer queues per scheduler. For details, see section 9.17 "Enabling Peer Scheduling" on page 362.
Default: unset

preemptive_sched — string: Enable job preemption. See section 9.14 "Enabling Preemptive Scheduling" on page 351 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.
fairshare	When the entity owning a job exceeds its fairshare limit.
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 whose entities are over their fairshare limit are third highest. The second example shows that starving jobs whose entities 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_sort — Whether jobs most eligible for preemption will be sorted according to their start times. Allowable values: "min_time_since_start", or no `preempt_sort` setting. If set to "min_time_since_start", first job preempted will be that with most recent start time. If not set, job will be that with longest running time. See "Preemption Ordering by Start Time" on page 353.

preempt_starving — Deprecated. Add "starving_jobs" `preempt_prio` parameter.

preempt_suspend — Deprecated. Add an "S" to `preempt_order` parameter.

primetime_prefix — string: Queue names starting with this prefix are 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 346.
Default: `p_`

prime_exempt_anytime_queues — Determines whether anytime queues are controlled by `backfill_prime`. If set to true, jobs in an anytime queue will not be prevented from running across a primetime/non-primetime or non-primetime/primetime boundary. If set to false, the jobs in an anytime queue may not cross this boundary, except for the amount specified by their `prime_spill` setting. See also `backfill_prime, prime_spill`.
Boolean.
Default: false.

prime_spill — Specifies the amount of time a job can spill over from non-primetime into primetime or from primetime into non-primetime. This option is only meaningful if `backfill_prime` is true. Also note that this option can be separately specified for prime- and non-primetime. See also `backfill_prime, prime_exempt_anytime_queues`.
Units: time.
Default: `00:00:00`

For example, the first setting below means that non-primetime jobs can spill into primetime 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 enforced when scheduling jobs. Vnode-level boolean resources are automatically enforced and do not need to be listed here. Limits are set by setting `resources_available.resourceName` on the Server objects (vnodes, 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`).

Default: "`ncpus, mem, arch, host, vnode`" (number CPUs, memory, architecture). If host is not added to the resources line, when the user submits a job requesting a specific vnode in the following syntax:
`qsub -l select=host=vnodeName`
the job will run on any host.

resource_unset_infinite

Comma-delimited list of resources. Resources in this list will be treated as infinite if they are unset. Cannot be set differently for primetime and non-primetime. Default: empty list.

Example: resource_unset_infinite: "vmem, foo_licenses"

round_robin — boolean: If set to true, the scheduler will consider one job from the first queue, then one job from the second queue, and so on in a circular fashion. If `sort_queues` is set to true, the queues are ordered with the highest priority queue first. Each scheduling cycle starts with the same highest-priority queue, which will therefore get preferential treatment If `round_robin` is set to false, the scheduler will consider jobs according to the setting of the `by_queue` attribute.

When true, overrides the `by_queue` attribute.

Default: `false all`

server_dyn_res — string: Directs the Scheduler to replace the Server's `resources_available` values with new values returned by a site-specific external program. See section 10.5.1 "Dynamic Server-level Resources" on page 385 for details of usage.

smp_cluster_dist — string: Specifies how single-host jobs should be distributed to all hosts of the complex. Options are: `pack`, `round_robin`, and `lowest_load`. `pack` means keep putting jobs onto one host until it is "full" and then move onto the next.

round_robin is to put one job on each vnode in turn before cycling back to the first one. lowest_load means to put the job on the lowest loaded host. See also section 9.11 “Configuring SMP Cluster Scheduling” on page 349, and section 9.12 “Enabling Load Balancing” on page 350.
Default: pack all

sort_by — **Deprecated**. Use job_sort_key.

sort_queues — Boolean. When set to true, queues are sorted so that the highest priority queues are considered first. Queues are sorted by each queue’s priority attribute. The queues are sorted in a descending fashion, that is, a queue with priority 6 comes before a queue with priority 3. See section 9.8 “How Queues are Ordered” on page 345.

This is a prime option, which means it can be selectively applied to primetime or non-primetime.

Default: true ALL

strict_fifo — **Deprecated**. Use strict_ordering.

strict_ordering — boolean: specifies that jobs must be run in the order determined by whatever sorting parameters are being used. This means that a job cannot be skipped due to resources required not being available. The jobs are sorted at the server level, not the queue level. If a job due to run next cannot run, no job will run, unless backfilling is used. Jobs can be backfilled around the job that’s due to run next, if it is blocked. See section 9.18.1 “Enabling FIFO Scheduling with strict_ordering” on page 366. Default: false.
Example line in PBS_HOME/sched_priv/sched_config:

```
strict_ordering: true ALL
```

sync_time — time: The amount of time between writing the fairshare usage data to disk. Requires fair_share to be enabled.
Default: 1:00:00

unknown_shares — integer: The number of shares for the “unknown” group. These shares determine the portion of a resource to be allotted to that group via fairshare. Requires fair_share to be enabled. See section 9.15 “Using Fairshare” on page 354 for information on how to use fairshare.

The "unknown" group gets 0 shares unless set.

9.4 Scheduler Attributes

Scheduler attributes can be read only by the PBS Manager or Operator. All scheduler attributes are read-only.

pbs_version — The version of PBS for this scheduler. Available only to Manager/Operator.

sched_host — The hostname of the machine on which the scheduler runs. Available only to Manager/Operator.

9.5 How Jobs are Placed on Vnodes

Placement sets allow the administrator to group vnodes into useful sets, and have multi-vnode jobs run in one set. For example, it makes the most sense to run a job on vnodes that are all connected to the same high-speed switch. PBS places each job on one or more vnodes according to the job's resource request, whether and how the vnodes have been grouped, and whether the vnodes can be shared. For more on sharing, see section "sharing" on page 212.

Using placement sets, vnodes are partitioned according to the value of one or more resources. These resources are listed in the node_group_key attribute. Grouping nodes is enabled by setting `node_group_enable` to True. If you use the server's node_group_key, the resulting groups apply to all of the jobs in the complex. If you use a queue's node_group_key, only jobs in that queue will have those groups applied to them. In order to have the same behavior as in the old node grouping, group on a single resource. If this resource is a string array, it should only have one value on each vnode. This way, each vnode will only be in one node group.

When the partitioning is done according to the values of more than one resource, that is, node_group_key lists more than one resource, the resulting groups are called placement sets. In placement sets, a vnode may belong to more than one set. For example, if a given vnode is on switch S1 but not switch S2 and router R1, it can belong to the set of vnodes that all share resources_available.switch=S1 and also to the set that all share resources_available.router=R1. It will not be in the set that all share resources_available.switch=S2. Each placement set is defined by the value of exactly one

resource, not a combination of resources. A series of placement sets is created according to the values of a resource across all the vnodes. For example, if there are three switches, S1, S2 and S3, and there are vnodes with resources_available.switch that take on these three values, then there will be three placement sets in the series. All of the placement sets defined by all of the resources in `node_group_key` are called a placement pool.

PBS will attempt to place each job in the smallest possible group or set that is appropriate for the job.

9.6 Placement Sets and Task Placement

Placement sets are the sets of vnodes within which pbs will try to place a job. PBS tries to determine which vnodes are connected (i.e. should be grouped together into one set), and the scheduler groups vnodes that share a placement value together in an effort to select which vnodes to assign to a job. The scheduler tries to put a job in the smallest appropriate placement set.

Placement sets are defined by string or multi-valued string resources chosen by the administrator. A placement set is the set of vnodes that share a value for a specific resource. A vnode can belong to more than one placement set defined by a multi-valued string resource. For example, if the resource is called “router”, and the vnode’s router resource is set to “router1, router2”, then the vnode will be in the placement set defined by router = router1 and the set defined by router = router2.

A placement pool is the collection of sets defined by one or more resources. So if we use only the resource called router, if the router resources on all the vnodes have some combination of router1 and router2, then there will be two placement sets in the router placement pool.

PBS may create default platform-dependent placement sets depending upon topology information. You can look for placement set names in the PBS-generated MOM configuration files or in the server’s pnames attribute.

9.6.1 Definitions

Task placement	The process of choosing a set of vnodes to allocate to a job that will both satisfy the job's resource request (select and place specifications) and satisfy the configured Scheduling policy.
Placement Set	A set of vnodes. Placement sets are used to improve task place-

ment (optimizing to provide a "good fit") by exposing information on system configuration and topology. Placement sets are defined using vnode-level resources of type multi-valued string. A single placement set is defined by one resource name and a single value; all vnodes in a placement set include an identical value for that specified resource. For example, assume vnodes have a resource named "switch", which can have values "A", "B", or "C": the set of vnodes which match "switch=B" is a placement set.

Placement Set Series — A set of sets of vnodes. A placement set series is defined by one resource name and all its values. A placement set series is the set of placement sets where each set is defined by one value of the resource. If the resource takes on N values at the vnodes, then there are N sets in the series. For example, assume vnodes have a resource named "switch", which can have values "A", "B", or "C": there are three sets in the series. The first is defined by the value "A", where all the vnodes in that set have the value "A" for the resource "switch". The second set is defined by "B", and the third by "C".

Placement Pool — A set of placement sets used for task placement. A placement pool is defined by one or more vnode-level resource names and the values of these resources on vnodes. In the example above, "switch" defines a placement pool of three placement sets. `node_group_key` defines a placement pool.

Static Fit — A job statically fits into a placement set if the job could fit into the placement set if the set were empty. It might not fit right now with the currently available resources.

Dynamic Fit — A job dynamically fits into a placement set if it will fit with the currently available resources (i.e. the job can fit right now).

9.6.2 Configuring Placement Sets

Placement is turned on by setting:

```
qmgr> set server node_group_enable = True
qmgr> set server node_group_key = <resource list>
```

For example, to create a placement pool for the resources vnodes, hosts, L2 and L3:

```
qmgr> set server node_group_key = "vnode,host,L2,L3"
```

If there is a vnode level resource called "cbrick" set on the vnodes on the Altix, then the node_group_key should include cbrick too, i.e.,

```
qmgr> set server \
   node_group_key="vnode,host,cbrick,L2,L3"
```

9.6.3 Multihost Placement Sets

Placement pools and sets can span hosts. This applies to multi-vnode machines that have been partitioned into more than one system. To set up a multihost placement set, set a given resource on the vnodes for more than one host, then put that resource in the `node_group_key`. For example, create a string_array resource called "`span`" in the PBS_HOME/server_priv/resourcedef file:

```
span type=string_array
```

Add the resource "span" to node_group_key on the server or queue. Use qmgr to give it the same value on all the vnodes. You must write a script that sets the same value on each vnode that you want in your placement set.

9.6.4 Machines with Multiple Vnodes

Machines with multiple vnodes such as the SGI Altix are represented as a generic set of vnodes. Placement sets are used to allocate resources on a single machine to improve performance and satisfy scheduling policy and other constraints. Jobs are placed on vnodes using placement set information.

For a cpusetted Altix running ProPack 4 or 5, the placement information for cpusets is generated by PBS. For a cpusetted Altix running ProPack 2 or 3, the placement information must be generated by another means. section 6.5.6.13 "Generate Vnode Definitions File for ProPack 2, 3" on page 134.

Node grouping allows vnodes to be in multiple placement sets. The string resource is a multi-valued string resource. Each value of the resource defines a different placement set. This creates a greater number of placement sets, and they may overlap (a vnode can be in more then one placement set). Not all placement sets have to contain the same number of vnodes.

Neither placement sets nor node grouping can be used with the IBM Blue Gene.

9.6.5 Order of Precedence for Job Placement

Different placement pools can be defined complex-wide (server-level), and per-queue. A server-level placement pool is defined by setting the server's `node_group_key`. A queue-level placement pool is defined by setting the queue's `node_group_key`. Jobs can only define placement sets. A per-job placement set is defined by the `-l place` statement in the job's resource request. Since the job can only request one value for the resource, it can only request one placement set. The scheduler uses the most specific placement pool for task placement for a job:

(a) If there is a per-job placement set defined, it is used, otherwise,
(b) If there is a per-queue placement pool defined for the queue the job is in, it is used, otherwise,
(c) If there is a complex-wide placement pool defined, it is used, otherwise,
(d) The placement pool consisting of one placement set of all vnodes is used.

This means that a job's place=group resource request overrides the sets defined by the queue's or server's node_group_key.

9.6.6 Defining Placement Sets

A placement pool is defined by one or more vnode-level resource names and the values of these resources on vnodes. This includes values that are unset or zero. For a single vnode-level resource RES which has `N` distinct values, `v1, ..., vN`, the placement set series defined by RES contains `N` sets of vnodes. Each set corresponds to one value of RES. For example, the placement set corresponding to RES and `v5` has the property that all vnodes in the set include `v5` in the value of RES. The placement pool defined by multiple resource names is simply the union of the placement pools defined by each individual resource name.

Server node_group_key attribute is an array of strings, e.g.,

```
Qmgr: set server node_group_key="res1,res2, …, resN"
```

Queue-level node_group_key attribute (also an array of strings):

```
Qmgr: set queue QNAME node_group_key="res1, …resN"
```

The complex-wide placement pool is defined by all resource names listed in the server-level `node_group_key`. Similarly, per-queue placement pools are defined by the queue-level `node_group_key`. Either of these pools can be defined using multiple resource names. Per-job placement pools are defined by the single resource name given in the place directive (group=RES).

On a multi-vnoded system which is set up to do so, MOM sends the Server a list of resource names to be used by the Scheduler for placement set information.

9.6.7 Placement Sets Defined by Unset Resources

If you have ten vnodes, on which there is a string resource COLOR, where two have COLOR set to “red”, two are set to “blue”, two are set to “green” and the rest are unset, there will be four placement sets defined by the resource COLOR. This is because the fourth placement set consists of the four vnodes where COLOR is unset. This placement set will also be the largest.

9.6.8 Ordering and Choosing Placement Sets

The selected `node_group_key` defines the placement pool. The scheduler will order the placement sets in the placement pool.

The sets are sorted in this order:

1. Static total ncpus of all vnodes in set
2. Static total mem of all vnodes in set
3. Dynamic free ncpus of all vnodes in set
4. Dynamic free mem of all vnodes in set

The vnodes are sorted within a set in this order:

5. Vnodes sorted by node_sort_key if using node_sort_key
   (see “Sorting Vnodes with node_sort_key”below)
6. Order the vnodes are returned by pbs_statnode() if no node_sort_key.
   This is the default order the vnodes appear in the output of the command: “pbsnodes -a”.

If a job can fit statically within any of the placement sets in the placement pool, then the scheduler places a job in the first placement set in which it dynamically fits. This ordering ensures the scheduler will use the smallest possible placement set in which the job will dynamically fit.

If a job cannot statically fit into any placement set in the placement pool, then the scheduler places the job in the placement set consisting of all vnodes. Note that if the user specifies **`-lplace=group=switch`**, but the job cannot statically fit into any switch placement set, then the job will still run, but not in a switch placement set.

9.6.8.1 Sorting Vnodes with `node_sort_key`

The vnodes within each placement set are sorted according to the `node_sort_key` option. The values sorted by `node_sort_key` must be numerical. The placement sets themselves are then ordered according to the criteria described in section 9.6.8 "Ordering and Choosing Placement Sets" on page 330. Up to 20 `node_sort_key` entries can be used, in which case the first entry will be the primary sort key, the second will be used to sort equivalent items from the first sort, etc.

Syntax:

```
node_sort_key: "<resource>|job_priority HIGH|LOW"
node_sort_key: "<resource> HIGH|LOW \
   total|assigned|unused"
```

Specifying a <resource> such as mem or ncpus sorts vnodes by the resource specified.

total	Use the resources_available value.
assigned	Use the resources_assigned value.
unused	Use the value given by resources_available - resources_assigned.

If the third argument (total|assigned|unused) is not specified with a resource, "total" will be used. This provides backwards compatibility with previous releases.

Specifying `job_priority` sorts vnodes by their `priority` attribute, and cannot be used with a third argument (assigned|unused|total).

Default:

```
node_sort_key: "job_priority HIGH"
```

Examples

If we use

```
node_sort_key: "ncpus HIGH unused"
```

this will sort vnodes by the highest number of unused cpus.

If we use

```
node_sort_key: "mem HIGH assigned"
```

this will sort vnodes by the highest amount of memory assigned to vnodes.

The old “nodepack” behavior can be achieved by

```
node_sort_key: “ncpus low unused”
```

In this example of the interactions between placement sets and `node_sort_key`, we have 8 vnodes numbered 1-8. The vnode priorities are the same as their numbers. We use:

```
node_sort_key: “job_priority LOW”
```

Using node_sort_key, the vnodes are sorted in order, 1 to 8. We have three placement sets:

A: 1, 2, 3, 4 when sorted by node_sort_key; 4, 1, 3, 2 when no node_sort_key is used
B: 5, 6, 7, 8 when sorted by node_sort_key; 8, 7, 5, 6 when no node_sort_key is used
C: 1-8 when sorted, 4, 1, 3, 2, 8, 7, 5, 6 when not sorted.

A 6-vnode job will not fit in either A or B, but will fit in C. Without the use of node_sort_key, it would get vnodes 4, 1, 3, 2, 8, 7. With node_sort_key, it would get vnodes 1 - 6, still in placement set C.

Caveats

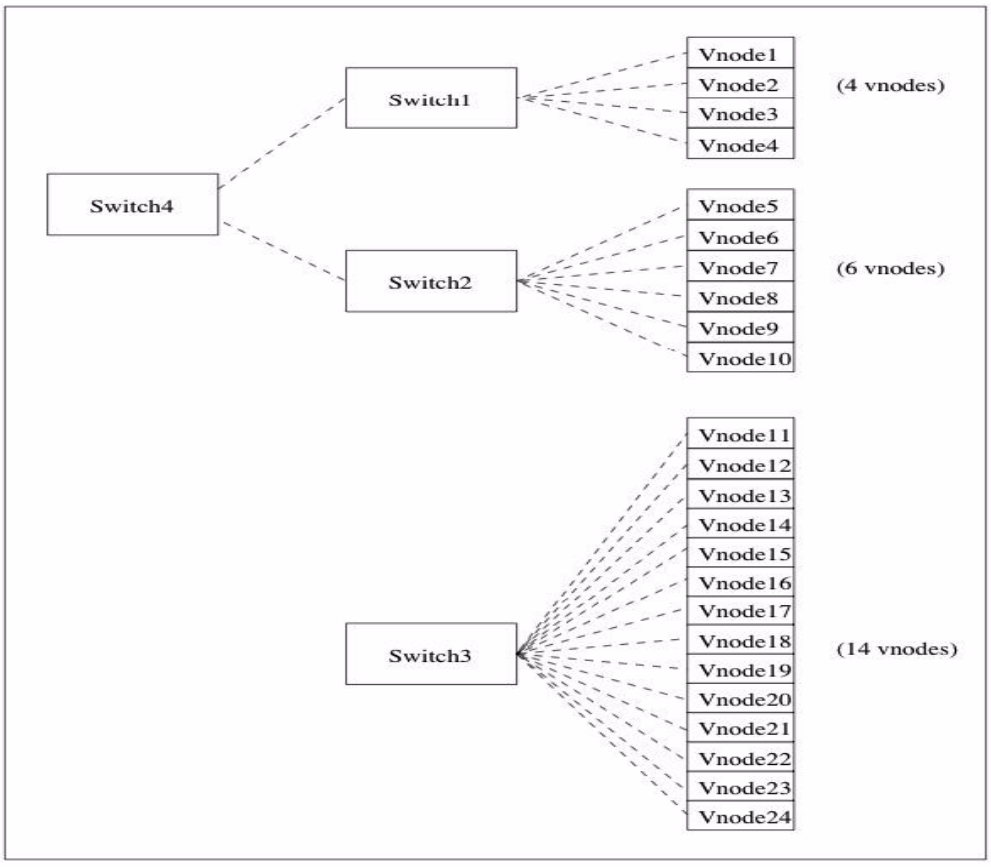
Sorting on a resource and using “unused” or “assigned” cannot be used with `load_balancing`. If both are used, load balancing will be disabled.

Sorting on a resource and using “unused” or “assigned” cannot be used with `smp_cluster_dist` when it is set to anything but “pack”. If both are used, `smp_cluster_dist` will be set to “pack”.

9.6.9 Placement Set Examples

9.6.9.1 Cluster with Four Switches

This cluster is arranged as shown with vnodes 1-4 on Switch1, vnodes 5-12 on Switch2, and vnodes 13-24 on Switch3. Switch1 and Switch2 are on Switch4.



To make the placement sets group the vnodes as they are grouped on the switches:

Create a custom resource called *switch*:

```
switch  type=string_array  flag=h
```

On vnodes[1-4] set:

```
resources_available.switch="switch1,switch4"
```

On vnodes[5-12] set:

```
resources_available.switch="switch2,switch4"
```

On vnodes[13-24] set:

```
resources_available.switch="switch3"
```

On the server set:
node_group_enable=true
node_group_key=switch

So you have 4 placement sets:

The placement set "switch1" has 4 vnodes
The placement set "switch2" has 6 vnodes
The placement set "switch3" has 14 vnodes
The placement set "switch4" has 10 vnodes

PBS will try to place a job in the smallest available placement set. Does the job fit into the smallest set (switch1)? If not, does it fit into the next smallest set (switch2)? This continues until it finds one where the job will fit.

PBS will choose the smallest currently available set in which the job fits dynamically. If no set in which the job fits dynamically is available, it will wait any set to become available. If the job will not statically fit in any placement set, it will run in the placement set made up of all vnodes.

9.6.9.2 Examples of Configuring Placement Sets on an Altix

To define new placement sets on an Altix, you can either use the qmgr command or you can create a site-defined MOM configuration file. See “Creation of Site-defined MOM Configuration Files” on page 259 and the `-s script_options` option to `pbs_mom` in “Options to pbs_mom” on page 409.

In this example, we define a new placement set using the new resource “NewRes”. We create a file called SetDefs that contains the changes we want.

Step 1 Add the new resource to the server’s resourcedef file:

```
NewRes type=string
```

Step 2 Add "NewRes" to the server's node_group_key

```
qmgr> set server \
node_group_key="vnode,host,L2,L3,NewRes"
```

Step 3 Restart the server

Step 4 Add "NewRes" to the value of the `pnames` attribute for the natural vnode. Add a line like this to SetDefs:

```
altix3:  resources_available.pnames = \
    L2,L3,NewRes
```

Step 5 For each vnode, V, that's a member of a new placement set you're defining, add a line of the form:

```
V:  resources_available.NewRes = \
    <new set name>
```

All the vnodes in <new set name> should have lines of that form, with the same <new set name> value, in the new config file. That is, if vnodes A, B, and C comprise a placement set, add lines that specify the value of <new set name>. Here the value of <new set name> is "P".

```
A:  resources_available.NewRes = P
B:  resources_available.NewRes = P
C:  resources_available.NewRes = P
```

For each new placement set you define, use a different value for <new set name>.

Step 6 Add SetDefs and tell MOM to read it, to make a site-defined MOM configuration file NewConfig.

```
pbs_mom -s insert NewConfig SetDefs
pkill -HUP pbs_mom
```

You can define more than one placement set at a time. Next we will use NewRes2 and give it two values, so that we have two placement sets.

Step 1 Add the new resource to the server's resourcedef file:

```
NewRes type=string_array
```

Step 2 Add "NewRes2" to the server's node_group_key

```
qmgr> set server \

node_group_key="vnode,host,L2,L3,NewRes2"
```

Step 3 Restart the server

Step 4 Add “NewRes2” to the value of the `pnames` attribute for the natural vnode. Add a line like this to SetDefs2:

```
altix3:  resources_available.pnames = \
    L2,L3,NewRes2
```

Step 5 For each vnode, V, that's a member of a new placement set you're defining, add a line of the form:

```
V:  resources_available.NewRes = \
    “<new set name1>,<new set name2>”
```

Here, we’ll put vnodes A, B and C into one placement set, and vnodes B, C and D into another.

```
A:  resources_available.NewRes2 = P
B:  resources_available.NewRes2 = “P,Q”
C:  resources_available.NewRes2 = “P,Q”
D:  resources_available.NewRes2 = Q
```

Step 6 Add SetDefs2 and tell MOM to read it, to make a site-defined MOM configuration file NewConfig.

```
pbs_mom -s insert NewConfig SetDefs2
pkill -HUP pbs_mom
```

You can also use the qmgr command to set the values of the new resource on the vnodes.

```
Qmgr: set node B resources_available.NewRes2=”P,Q”
```

9.6.9.3 Example of Placement Pool

In this example, we have vnodes connected to four cbricks and two L2 connectors. Since these come from the MOM, they are automatically added to the server’s resourcedef file.

Enable placement sets:

```
Qmgr: s s node_group_enable=True
```

Define the pool you want:

```
Qmgr: s s node_group_key="cbrick, L2"
```

If the vnodes look like this, from "pbsnodes -av ! egrep '(^[^]) | cbrick" or "pbsnodes -av ! egrep '(^[^]) | L2" :

```
vnode1
   resources_available.cbrick=cbrick1
   resources_available.L2=A
vnode2
   resources_available.cbrick=cbrick1
   resources_available.L2=B
vnode3
   resources_available.cbrick=cbrick2
   resources_available.L2=A
vnode4
   resources_available.cbrick=cbrick2
   resources_available.L2=B
vnode5
   resources_available.cbrick=cbrick3
   resources_available.L2=A
vnode6
   resources_available.cbrick=cbrick3
   resources_available.L2=B
vnode7
   resources_available.cbrick=cbrick4
   resources_available.L2=A
vnode8
   resources_available.cbrick=cbrick4
   resources_available.L2=B
```

There are six resulting placement sets.
cbrick=cbrick1: {vnode1, vnode2}
cbrick=cbrick2: {vnode3, vnode4}

cbrick=cbrick3: {vnode5, vnode6}
cbrick=cbrick4: {vnode7, vnode8}
L2=A: {vnode1, vnode3, vnode5, vnode7}
L2=B: {vnode2, vnode4, vnode6, vnode8}

9.6.9.4 Colors Example

A placement pool is defined by two resources: `colorset1` and `colorset2,` by using "`node_group_key=colorset1,colorset2"`. If a vnode has:

```
resources_available.colorset1=blue, red
resources_available.colorset2=green
```

The placement pool contains three placement sets. These are

```
{resources_available.colorset1=blue}
{resources_available.colorset1=red}
{resources_available.colorset2=green}
```

This means the vnode is in all three placement sets. The same result would be given by using one resource and setting it to all three values, e.g. `colorset=blue,red,green`.

Example: We have five vnodes `v1 - v5:`

```
v1 color=red host=mars
v2 color=red host=mars
v3 color=red host=venus
v4 color=blue host=mars
v5 color=blue host=mars
```

The placement pools are defined by

```
node_group_key=color
```

The resulting node groups would be: `{v1, v2, v3}, {v4, v5}`

9.6.9.5 Simple Node Grouping on Switch Example

Say you have a cluster with two high-performance switches each with half the vnodes 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". See "Defining New Custom Resources" on page 374.

Next, we need to enable node grouping and specify the resource to use:

```
Qmgr: set server node_group_enable=True
Qmgr: set server node_group_key=switch
```

Now, set the value for switch on each vnode:

```
Qmgr: active node vnode1,vnode2,vnode3
Qmgr: set node resources_available.switch=A
Qmgr: active node vnode4,vnode5,vnode6
Qmgr: set node resources_available.switch=B
```

Now there are two placement sets:
switch=A: {vnode1, vnode2, vnode3}
switch=B: {vnode4, vnode5, vnode6}

9.6.10 Breaking Chunks Across Vnodes

Chunks can be broken up across vnodes that are on the same host. This is generally used for jobs requesting a single chunk. On vnodes with sharing=defalt_excl, jobs are assigned entire vnodes exclusively. For vnodes with sharing=default_shared, this causes a different allocation: unused memory on otherwise-allocated vnodes is allocated to the job. The exec_vnode attribute will show this allocation. Chunks are only placed on vnodes whose state is "free".

On the Altix, the scheduler will share memory from a chunk even if all the cpus are used. It will first try to put a chunk entirely on one vnode. If it can, it'll run it there. If not, it'll break the chunk up across any vnode it can get resources from, even for small amounts of unused memory.

9.6.11 Reservations

The same rules about placement sets are used for reservation jobs as are used for regular jobs.

9.6.12 Node Grouping

Node grouping is the same as one placement set series, where the placement sets are defined by one resource. This is also called complex-wide node grouping.

9.6.13 Non-backward-compatible Change in Node Grouping

Given the following example configuration:
node1: switch=A
node2: switch=A
node3: switch=B
node4: switch=B
node5: switch unset
Qmgr: s s node_group_key=switch

There is no change in the behavior of jobs submitted with qsub -l ncpus=1
version 7.1: The job can run on any node: node1 .. node5
version 8.0: The job can run on any node: node1 .. node5

Example of 8.0 and later behavior: jobs submitted with qsub -l nodes=1
version 7.1: The job can only run on nodes: node1, node2, node3, node4
It will never use node5
version 8.0: The job can run on any node: node1 .. node5

Overall, the change for version 8.0 was to include every vnode in node grouping (when enabled). In particular, if a resource is used in node_group_key, PBS will treat every vnode as having a value for that resource, hence every vnode will appear in at least one placement set for every resource. For vnodes where a string resource is "unset", PBS will behave as if the value is ““.

9.7 Job Priorities in PBS Professional

There are various classes of default job priorities within PBS Professional, which can be enabled and combined based upon customer needs. The following table illustrates the inherent ranking of the defaults for these different classes of priorities. This is the ordering that the scheduler uses. A higher ranking class always takes precedence over lower ranking classes, but within a given class the jobs are ranked according to the attributes specific to that class. For example, since the Reservation class is the highest ranking class, jobs in that class will be run (if at all possible) before jobs from other classes. If a job qualifies for more than one category, it falls into the higher-ranked category. In the following table, higher-ranked classes are shown above lower-ranked.

Table 17: Classes of Job Priorities

Class	Description
Reservation	Jobs submitted to an Advance Reservation, thus resources are already reserved for the job.
Express	High-priority ("express" jobs). See discussion in section 9.14 "Enabling Preemptive Scheduling" on page 351.
Starving	Jobs that have waited longer than the starving job threshold. See also the Scheduler configuration parameters `help_starving_jobs`, `max_starve`, and `backfill`.
Suspended	Jobs that have been suspended by higher priority work.
round_robin or by_queue	Queue-based scheduling may affect order of jobs depending on whether these options are enabled.
job_sort_formula, fairshare, or job_sort_key	Jobs are sorted as specified by the formula in `job_sort_formula`, if it exists, or by fairshare, if it is enabled and there is no formula, or if neither of those is used, by `job_sort_key`.

You can specify a formula for sorting jobs. This formula determines how jobs are sorted in the lowest ranked category in the table above. See section 9.7.2 "Tunable Formula for Computing Job Priorities" on page 342.

While the lowest category does sort jobs at the finest granularity, most of the work of sorting jobs is done in this category. The precedence of the categories cannot be changed.

9.7.1 Running Jobs in Submission Order

To run jobs in the order in which they were submitted, comment out the default `job_sort_key` in sched_priv/sched_config, and do not provide a job sorting formula in job_sort_formula. For example, to run jobs by queue priority, and then by submission order, with strict ordering and backfill, set the following:

by_queue: true
strict_odering: true
backfill: true

Give each queue a priority value.

9.7.2 Tunable Formula for Computing Job Priorities

You can choose to use a formula by which to sort jobs at the finest-granularity level. These levels are shown in the table "Classes of Job Priorities" on page 341. This formula will override both `job_sort_key` and fairshare for sorting at that level. You specify the formula in the server's `job_sort_formula` attribute. If that attribute contains a formula, the scheduler will use it. If not, the scheduler computes job priorities according to fairshare, if fairshare is enabled. If neither is defined, the scheduler uses `job_sort_key`. When the scheduler sorts jobs according to the formula, it computes a priority for each job, where that priority is the value produced by the formula. Jobs with a higher value get higher priority.

The formula can only direct how jobs are sorted at the finest level of granularity. However, that is where most of the sorting work is done.

Once you set `job_sort_formula` via qmgr, it takes effect with the following scheduling cycle. The range for the formula is defined by the IEEE floating point standard for a double. If you use queue priority in the formula and the job is moved to another server through peer scheduling, the queue priority used in the formula will be that of the queue to which the job is moved. Variables are evaluated at the start of the scheduling cycle.

To set the `job_sort_formula` attribute, use the `qmgr` command.

```
Qmgr> s s job_sort_formula = "<formula>"
```

The formula can be made up of any number of *expressions*, where expressions contain *terms* which are added, subtracted or multiplied. You cannot use division. Multiplication takes precedence over addition or subtraction. You cannot use two operators in a row. For example, "A +-B" is disallowed.

Table 18: Terms in Tunable Formula

Terms	Allowable Value
Constants	NUM or NUM.NUM

Table 18: Terms in Tunable Formula

Terms		Allowable Value
Attribute values	queue_priority	Value of `priority` attribute for queue in which job resides
	job_priority	Value of the job's `priority` attribute
	fair_share_perc	Percentage of fairshare tree for this job's entity
Resources		ncpus
		mem
		walltime
		cput
Custom numeric job-wide resources		Uses the amount requested, not the amount used. Must be of type long, float, or size. See section 10.1.1 "Custom Resource Formats" on page 372.

Examples of formulas:
Example 1: 10 * ncpus + 0.01*walltime + A*mem
Where "A" is a custom resource

Example 2: ncpus + 0.0001*mem

Example 3 : To change the formula on a job-by-job basis, alter the value of a resource in the job's Resource_List.RES. So if the formula is A *queue_priority + B*job_priority + C*ncpus + D*walltime, where A-D are custom numeric resources. These resources can have a default value via resources_default.A .. resources_default.D. You can change the value of a job's resource through qalter.

Example 4: ncpus*mem

Example 5: Set via qmgr:

```
qmgr -c 'set server job_sort_formula=
   5*ncpus+0.05*walltime'
```

Following this, the output from **qmgr -c 'print server'** will look like

```
Set server job_sort_formula="5*ncpus+0.05*walltime"
```

Example 6:

```
Qmgr> s s job_sort_formula=ncpus
```

Example 7:

```
Qmgr> s s job_sort_formula='queue_priority + ncpus'
```

Example 8:

```
Qmgr> s s job_sort_formula=
        '5*job_priority + 10*queue_priority'
```

9.7.3 Units

The variables you can use in the formula have different units. Make sure that some terms do not overpower others by normalizing them where necessary. Resources like ncpus are from 1..N, size resources like mem are in kb, so 1gb is 1048576kb, and time resources are in seconds (e.g. walltime). Therefore, if you want a formula that combines memory and ncpus, you'll have to account for the factor of 1024 difference in the units.

The following are the units for the supported built-in resources:

Time resources:	seconds
Memory:	kb, so 1gb => 1048576kb
ncpus:	1..N

Example: if you use '1 * ncpus + 1 * mem', where mem=2mb, ncpus will have almost no effect on the formula result. However, if you use '1024 * ncpus + 1 * mem', the scaled mem won't overpower ncpus.

Example: you are using gb of mem:

```
qmgr> s s job_sort_formula='1048576 * ncpus + 2 * mem'
```

Example: if you want to add days of waiting to to queue priority, you might want to multiply the time by 0.0000115, equivalent to dividing by the number of seconds in a day:

```
qmgr> s s job_sort_formula =
   '.0000115*walltime + queue_priority'
```

9.7.4 Caveats and Error Messages

It is invalid to set both `job_sort_formula` and `job_sort_key` at the same time. If they are both set, `job_sort_key` is ignored and the following error message is logged:

```
"Job sorting formula and job_sort_key are incompatible.  The
job sorting formula will be used."
```

If the formula overflows or underflows the sorting behavior is undefined.

If you set the formula to an invalid formula, `qmgr` will reject it, with one of the following error messages:

```
"Invalid Formula Format"
"Formula contains invalid keyword"
"Formula contains a resource of an invalid type"
```

9.7.5 Logging

For each job, the evaluated formula answer is logged at the highest logging level (DEBUG3):

```
"Formula  Evaluation = <answer>"
```

9.8 How Queues are Ordered

The order in which jobs are considered by the scheduler depends upon which queues those jobs are in, and the ordering of those queues. A queue's priority determines where it is in the list of queues examined. If queues don't have priority assigned to them, then the order in which they are considered is essentially random. So if you wish to have queues considered in a particular order, give each queue a different priority.

9.9 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 9.3 "Scheduler Configuration Parameters" on page 315). The format of entries is:

```
# From Date-Time   To Date-Time
# MM/DD/YYYY HH:MM MM/DD/YYYY HH:MM
# For example
04/15/2007 12:00 04/15/2007 15:30
```

In order to use a dedicated time queue, jobs must have a walltime. Jobs that do not have a walltime will never run.

To force the Scheduler to re-read the dedicated time file (needed after modifying the file), restart or reinitialize (HUP) the Scheduler. (For details, see "Starting and Stopping PBS: UNIX and Linux" on page 405 and "Starting and Stopping PBS: Windows 2000 / XP" on page 421.)

9.10 Defining Primetime and Holidays

Often is it useful to change scheduler policy at predetermined intervals over the course of the work week or day. *Prime* and *nonprime* are times when prime or non-primetime start. To have the Scheduler enforce a distinction between primetime (usually, the normal work day) and non-primetime (usually nights and weekends), as well as enforcing non-primetime scheduling policy during holidays, edit the *PBS_HOME*/sched_priv/holidays file to specify the appropriate values for the begin and end of primetime, and any 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>
```

If there is no YEAR line in the holidays file, primetime will be in force at all times. *Day* can be *weekday, monday, tuesday, wednesday, thursday, friday, saturday, or sunday.* The ordering of <day> lines in the holidays file controls how primetime is determined. A later line takes precedence over an earlier line.

For example:

```
weekday 0630    1730
friday  0715    1600
```

means the same as

```
monday  0630    1730
tuesday 0630    1730
wednesday 0630  1730
thursday 0630   1730
friday  0715    1600
```

However, if a specific day is followed by "weekday",

```
friday  0700    1600
weekday 0630    1730
```

the "weekday" line takes precedence, so Friday will have the same primetime as the other weekdays. Each line must have all three fields. In order to have the equivalent of primetime overnight, swap the definitions of prime and non-prime in the scheduler's configuration file.

Times can either be HHMM with no colons(:) or the word "`all`" or "`none`" to specify that a day is all primetime or non-primetime.

```
<day of year> <date> <holiday>
```

PBS Professional uses the <day of year> field and ignores the <date> string. *Day of year* is the julian 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"). Day names must be lowercase.

```
YEAR    2007
*             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
   15        Jan 15       Dr. M.L. King Day
   50        Feb 19       President's Day
  148        May 28       Memorial Day
  185        Jul 4        Independence Day
  246        Sep 3        Labor Day
  281        Oct 8        Columbus Day
  316        Nov 12       Veteran's Day
  326        Nov 22       Thanksgiving
  359        Dec 25       Christmas Day
```

Reference copies of the holidays file for years 2007, 2008 and 2009 are provided in `PBS_HOME/sched_priv/holiday.2007`, `PBS_HOME/sched_priv/holiday.2008`, and `PBS_HOME/sched_priv/holiday.2009`. To use any of these as the holidays file, copy it to `PBS_HOME/sched_priv/holidays` -- note the “s” on the end of the filename.

If backfill_prime is set to True, the scheduler won’t run any jobs which would overlap the boundary between primetime and non-primetime. This assures that jobs restricted to running in either primetime or non-primetime can start as soon as the time boundary happens.

If prime_exempt_anytime_queues is set to True, anytime queues are not controlled by backfill_prime, which means that jobs in an anytime queue will not be prevented from running across a primetime/nonprimetime or non-primetime/primetime boundary. If set to False, the jobs in an anytime queue may not cross this boundary, except for the amount specified by their prime_spill setting.

The scheduler logs a message at the beginning of each scheduling cycle saying whether it is primetime or not, and when this period of primetime or non-primetime will end. The message is at debug level DEBUG2. The message is of this form:
“It is primetime and it will end in NN seconds at MM/DD/YYYY HH:MM:SS”

or
"It is non-primetime and it will end in NN seconds at MM/DD/YYYY HH:MM:SS"

9.11 Configuring SMP Cluster Scheduling

The scheduler schedules SMP clusters in an efficient manner. Instead of scheduling only via load average of hosts, it takes into consideration the resources specified at the server, queue, and vnode 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 hosts are selected.

The available choices are `pack` (pack one vnode until full), `round_robin` (put one job on each vnode in turn), or `lowest_load` (put one job on the lowest loaded host). The `smp_cluster_dist` parameter should be used in conjunction with `node_sort_key` to ensure efficient scheduling. (Optionally, you may wish to enable "load balancing" in conjunction with SMP cluster scheduling. For details, see section 9.12 "Enabling Load Balancing" on page 350.)

Important: This feature only applies to single-host jobs where the number of chunks is 1, and *place=pack* has been specified.

Note that on a multi-vnode machine, `smp_cluster_dist` will distribute jobs across vnodes but the jobs will end up clustered on a single host.

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 vnodes via `qmgr` just like on the server or queues. For example, to set maximum limits on a host called "host1" to 10 CPUs and 2 GB of memory:

```
Qmgr: set node host1 resources_available.ncpus=10
Qmgr: set node host1 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 vnode. 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, host”
```

9.12 Enabling Load Balancing

The load balancing scheduling algorithm will balance the computational load of single-vnode jobs (i.e. not multi-vnode jobs) across a complex. The load balancing takes into consideration the load on each host as well as all resources specified in the “resource” list. Load balancing uses the value for “loadave” returned by the operating system. For UNIX/Linux, this is the raw one minute averaged "loadave"; for Windows, there is one choice for “loadave”.

When the loadave is above max_load, that node is marked “busy”. The scheduler won’t place jobs on a node marked “busy”. When the loadave drops below ideal_load, the “busy” mark is removed. Consult your OS documentation to determine values that make sense.

The load average will slowly increase over time and more jobs than you want may be started at first. Over a period of time, the load average will move up to a point where no additional jobs will be started on that node. As jobs terminate the load average will slowly move lower and it will take time before the node is the best choice for new jobs.

To configure load balancing, first enable the option in the Scheduler’s configuration file:

```
load_balancing: True ALL
```

Next, configure SMP scheduling as discussed in the previous section, section 9.11 “Configuring SMP Cluster Scheduling” on page 349.

Next, configure the ideal and maximum desired load in each execution host’s MOM configuration file. (See also the discussion of these two MOM options in section 8.2.2 “Syntax and Contents of Default Configuration File” on page 260.)

```
$ideal_load 30
$max_load   32
```

Last, set each host's `resources_available.ncpus` to the maximum number of CPUs you wish to allocate on that host.

9.13 Managing Load Levels on Hosts

The "loadave" reported by MOM is the raw one minute averaged "loadave" returned by the operating system. When the loadave is above max_load, that node is marked "busy". The scheduler won't place jobs on a node marked "busy". When the loadave drops below ideal_load, the "busy" mark is removed. Consult your OS documentation to determine values that make sense.

If you wish to run non-PBS processes on a host, you can prevent PBS from using more than you want on that host. Set ideal_load and max_load in MOM's configuration file to values that are low enough to allow other processes to use some of the host.

If you want to prevent PBS from placing jobs on an already-overloaded machine, set max_load and ideal_load to the values you want for the host. When the load goes above max_load, no more jobs will be run on that host. This will prevent jobs from being started on a host where rogue processes are taking up all the CPU time.

9.14 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 315). 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 and Restart" on page 273), 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.

Important: If the Scheduler cannot find enough work to preempt in order to run a given job, it will not preempt any work.

When a job is suspended, its FLEX licenses are returned to the license pool, subject to the constraints of the server's pbs_license_min and pbs_license_linger_time attributes. The scheduler checks to make sure that FLEX licenses are available before resuming any job. If the required licenses are not available, the scheduler will log a message and add a comment to the job. See section 5.9.1.1 "Licensing and Job States" on page 102.

There are several Scheduler parameters to control preemption. The `preemptive_sched` parameter turns preemptive scheduling on and off. You can set the minimum queue priority needed to identify a queue as an express queue via the `preempt_queue_prio` parameter. 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 (If a job does not fall into any of the specified levels, it will be placed into `normal_jobs`.). If you want normal jobs to preempt other lower priority jobs, put `normal_jobs` 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. The `preempt_order` parameter can be used to specify the preemption method(s) to be used. If one listed method fails, the next one will be attempted.

Soft run limits can be set or unset via `qmgr.` If unset, the limit will not be applied to the job. However if soft run limits are specified on the Server, either of `queue_softlimits` or `server_softlimits` need to be added to the `preempt_prio` line of the Scheduler's configuration file in order to have soft limits enforced by the Scheduler.

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. When you are using soft limits, you want to have jobs that are not over their soft limits have higher priority. This is so that a job over its soft limit will not be run, just to be preempted later in the cycle by a job that is not over its soft limits. To do this, use

```
job_sort_key:"preempt_priority HIGH"
```

Note that any queue with a priority 150 (default value) or higher is treated as an express (i.e. high priority) queue.

For example: One group of users, group A, has submitted enough jobs that the group is over their soft limit. A second group, group B, submits a job and are under their soft limit. If preemption is enabled, jobs from group A will be preempted until the job from group B can run.

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

# specify 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,
starving_jobs+server_softlimits, server_softlimits"

# specify when to use each preemption method. If the first
# method fails, try the next method. If a job 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"
```

9.14.1 Preemption Ordering by Start Time

PBS has a feature that allows a different ordering of preemption of jobs. The default behavior will order preemption of jobs by most recent start time. If "preempt_sort" is disabled, then the first submitted job will be preempted.

For example, if we have two jobs, job A submitted at 10:00 a.m. and job B submitted at 10:30 a.m., the default behavior will preempt job A, and the alternate behavior will preempt job B.

In PBS_HOME/sched_priv/sched_config, the keyword preempt_sort can be set to “min_time_since_start” to enable this alternate behavior.

9.15 Using Fairshare

Fairshare provides a way to enforce a site's resource usage policy. It is a method for ordering the start times of jobs based on two things: how a site's resources are apportioned, and the resource usage history of site members. Fairshare ensures that jobs are run in the order of how deserving they are. The scheduler performs the fairshare calculations each scheduling cycle. If fairshare is enabled, all jobs have fairshare applied to them and there is no exemption from fairshare.

The administrator can employ basic fairshare behavior, or can apply a policy of the desired complexity.

9.15.1 Outline of How Fairshare Works

The owner of a PBS job can be defined for fairshare purposes to be a user, a group, an accounting string, etc. For example, you can define owners to be groups, and can explicitly set each group’s relationship to all the other groups by using the tree structure. You can define one group to be part of a larger department.

The usage of exactly one resource is tracked for all job owners. So if you defined job owners to be groups, and you defined cput to be the resource that is tracked, then only the cput usage of groups is considered. PBS tries to ensure that each owner gets the amount of resources that you have set for it.

If you don’t explicitly list an owner, it will fall into the “unknown” catchall. All owners in “unknown” get the same resource allotment.

9.15.2 The Fairshare Tree

Fairshare uses a tree structure, where each vertex in the tree represents some set of job owners and is assigned usage *shares*. Shares are used to apportion the site’s resources. The default tree always has a root vertex and an *unknown* vertex. The default behavior of fairshare is to give all users the same amount of the resource being tracked. In order to apportion a site's resources according to a policy other than equal shares for each user, the administrator creates a fairshare tree to reflect that policy. To do this, the administrator edits the file `PBS_HOME/sched_priv/resource_group`, which describes the fairshare tree.

9.15.3 Enabling Basic Fairshare

If the default fairshare behavior is enabled, all users with queued jobs will get an equal share of CPU time. The root vertex of the tree will have one child, the `unknown` vertex. All users will be put under the unknown vertex, and appear as children of the `unknown` vertex.

Basic fairshare is enabled by doing two things: in `PBS_HOME/sched_priv/sched_config`, set the scheduler configuration parameter `fair_share` to true, and uncomment the `unknown_shares` setting so that it is set to `unknown_shares: 10`.

Note that a variant of basic fairshare has all users listed in the tree as children of root. Each user can be assigned a different number of shares. This must be explicitly created by the administrator.

9.15.4 Using Fairshare to Enforce Policy

The administrator sets up a hierarchical tree structure made up of interior vertices and leaves. Interior vertices are *departments*, which can contain both departments and leaves. Leaves are for *fairshare entities*, defined by setting `fairshare_entity` to one of the following: `euser`, `egroup`, `egroup:euser`, `account_string`, or queues. Apportioning of resources for the site is among these entities. These entities' usage of the designated resource is used in determining the start times of the jobs associated with them. All fairshare entities must be the same type. If you wish to have a user appear in more than one department, you can use `egroup:euser` to distinguish between that user's different resource allotments.

Table 19: Using Fairshare Entities

Keyword	Fairshare Entities	Purpose
euser	Username	Individual users are allotted shares of the resource being tracked. Each username may only appear once, regardless of group.
egroup	Group name	Groups as a whole are allotted shares of the resource being tracked.

Table 19: Using Fairshare Entities

Keyword	Fairshare Entities	Purpose
egroup:euser	Combinations of username and group name	Useful when a user is a member of more than one group, and needs to use a different allotment in each group.
account_string	Account IDs	Shares are allotted by account.
queues	Queues	Shares are allotted between queues.

9.15.4.1 Shares in the Tree

The administrator assigns shares to each vertex in the tree. The actual number of shares given to a vertex or assigned in the tree is not important. What is important is the ratio of shares among each set of sibling vertices. Competition for resources is between siblings only. The sibling with the most shares gets the most resources.

9.15.4.2 Shares Among Unknown Entities

The root vertex always has a child called `unknown`. Any entity not listed in `PBS_HOME/sched_priv/resource_group` will be made a child of `unknown`, designating the entity as unknown. The shares used by unknown entities are controlled by two parameters in `PBS_HOME/sched_priv/sched_config: unknown_shares` and `fairshare_enforce_no_shares`.

The parameter `unknown_shares` controls how many shares are assigned to the `unknown` vertex. The `unknown` vertex will have 0 shares if `unknown_shares` is commented out. If `unknown_shares` is not commented out, the `unknown` vertex's shares default to 10. The children of the `unknown` vertex have equal amounts of the shares assigned to the `unknown` vertex.

The parameter `fairshare_enforce_no_shares` controls whether an entity without any shares can run jobs. If `fairshare_enforce_no_shares` is true, then entities without shares cannot run jobs. If it is set to false, entities without any shares can run jobs, but only when no other entities' jobs are available to run.

9.15.4.3 Format for Describing the Tree

The file describing the fairshare tree contains four columns to describe the vertices in the tree. The columns are for a vertex's name, its fairshare ID, the name of its parent vertex, and the number of shares assigned to that vertex. Vertex names and IDs must be unique. Vertex IDs are integers.

Neither the root vertex nor the `unknown` vertex is described in PBS_HOME/sched_priv/resource_group. They are always added automatically. Parent vertices must be listed before their children.

For example, we have a tree with two top-level departments, Math and Phys. Under math are the users Bob and Tom as well as the department Applied. Under Applied are the users Mary and Sally. Under Phys are the users John and Joe. Our PBS_HOME/sched_priv/resource_group looks like this:

```
Math          100   root      30
Phys          200   root      20
Applied       110   Math      20
Bob           101   Math      20
Tom           102   Math      10
Mary          111   Applied   1
Sally         112   Applied   2
John          201   Phys      2
Joe           202   Phys      2
```

If you wish to use egroup:euser as your entity, and Bob to be in two UNIX/Windows groups pbsgroup1 and pbsgroup2, and Tom to be in two groups pbsgroup2 and pbsgroup3:

```
Math          100   root      30
Phys          200   root      20
Applied       110   Math      20
pbsgroup1:Bob 101   Phys      20
pbsgroup2:Bob 102   Math      20
pbsgroup2:Tom 103   Math      10
pbsgroup3:Tom 104   Applied   10
```

A user's egroup, unless otherwise specified, will default to their primary UNIX/Windows group. When a user submits a job using the -Wgroup_list=<group>, the job's egroup will be <group>. For example, user Bob is in pbsgroup1 and pbsgroup2. Bob uses "qsub

-Wgroup_list= pbsgroup1 to submit a job that will be charged to pbsgroup1, and qsub -Wgroup_list=pbsgroup2 to submit a job that will be charged to pbsgroup2.

9.15.4.4 Computing How Much Each Vertex Deserves

How much resource usage each entity deserves is its portion of all the shares in the tree, divided by its past and current resource usage.

A vertex's portion of all the shares in the tree is called *tree percentage*. It is computed for all of the vertices in the tree. Since the leaves of the tree represent the entities among which resources are to be shared, their tree percentage sums to 100 percent.

The scheduler computes the tree percentage for the vertices this way:

First, it gives the root of the tree a tree percentage of 100 percent. It proceeds down the tree, finding the tree percentage first for immediate children of root, then their children, ending with leaves.

For each internal vertex A:
sum the shares of its children;
For each child J of vertex A:
divide J's shares by the sum to normalize the shares;
multiply J's normalized shares by vertex A's tree percentage to find J's tree percentage.

9.15.5 Tracking Resource Usage

The administrator selects exactly one resource to be tracked for fairshare purposes by setting the scheduler configuration parameter `fairshare_usage_res` in `PBS_HOME/sched_priv/sched_config`. The default for this resource is `cput`, CPU time. Another resource is the exact string "`ncpus*walltime`" which multiplies the number of cpus used by the walltime in seconds. An entity's usage always starts at 1. Resource usage tracking begins when the scheduler is started.

Each entity's current usage of the designated resource is combined with its previous usage. Each scheduler cycle, the scheduler adds the usage increment between this cycle and the previous cycle to its sum for the entity. Each entity's usage is *decayed*, or cut in half periodically, at the interval set in the `half_life` parameter in `PBS_HOME/sched_priv/sched_config`. This interval defaults to 24 hours.

This means that an entity with a lot of current or recent usage will have low priority for starting jobs, but if the entity cuts resource usage, its priority will go back up after a few decay cycles.

Note that if a job ends between two scheduling cycles, its resource usage between the end of the job and the following scheduling cycle will not be recorded. The scheduler's default cycle interval is 10 minutes. The scheduling cycle can be adjusted via the `qmgr` command. Use `qmgr: set server scheduler_iteration=<new value>`

9.15.6 Finding the Most Deserving Entity

The most deserving entity is found by starting at the root of the tree, comparing its immediate children, finding the most deserving, then looking among that vertex's children for the most deserving child. This continues until a leaf is found. In a set of siblings, the most deserving vertex will be the vertex with the lowest ratio of resource usage divided by tree percentage.

9.15.7 Choosing Which Job to Run

The job to be run next will be selected from the set of jobs belonging to the most deserving entity. The jobs belonging to the most deserving entity are sorted according to the methods the scheduler normally uses. This means that fairshare effectively becomes the primary sort key. If the most deserving job cannot run, then the next most is selected to run, and so forth. All of the most deserving entity's jobs would be examined first, then those of the next most deserving entity, et cetera.

At each scheduling cycle, the scheduler attempts to run as many jobs as possible. It selects the most deserving job, runs it if it can, then recalculates to find the next most deserving job, runs it if it can, and so on.

When the scheduler starts a job, all of the job's requested usage is added to the sum for the owner of the job for one scheduling cycle. The following cycle, the job's usage is set to the actual usage used between the first and second cycles. This prevents one entity from having all its jobs started and using up all of the resource in one scheduling cycle.

9.15.8 Files and Parameters Used in Fairshare

`PBS_HOME/sched_priv/sched_config`

fair_share	[true/false] Enable or disable fairshare
fairshare_usage_res	Resource whose usage is to be tracked; default is cput

half_life	Decay time period; default is 24 hours
sync_time	Time between writing all data to disk; default 1 hour
unknown_shares	Number of shares for unknown vertex; default 10, 0 if commented out
fairshare_entity	The kind of entity which is having fairshare applied to it. Leaves in the tree are this kind of entity. Default: euser.
fairshare_enforce_no_shares	If an entity has no shares, this controls whether it can run jobs. T: an entity with no shares cannot run jobs. F: an entity with no shares can only run jobs when no other jobs are available to run.
by_queue	If on, queues cannot be designated as fairshare entities, and fairshare will work queue by queue instead of on all jobs at once.

PBS_HOME/sched_priv/resource_group
Contains the description of the fairshare tree.

PBS_HOME/sched_priv/usage
Contains the usage database.

qmgr
Used to set scheduler cycle frequency; default is 10 minutes.
Qmgr: set server scheduler_iteration=<new value>

job attributes
Used to track resource usage:
resources_used.<resource>
Default is cput.

9.15.9 Fairshare and Queues

The scheduler configuration parameter by_queue in the file PBS_HOME/sched_priv/sched_config is set to on by default. When by_queue is true, fairshare cycles through queues, not overall jobs. So first fairshare is applied to Queue1, then Queue2, etc. If by_queue is true, queues cannot be designated as fairshare entities.

9.15.10 Fairshare and Strict Ordering

Fairshare dynamically reorders the jobs with every scheduling cycle. Strict ordering is a rule that says we always run the next-most-deserving job. If there were no new jobs submitted, strict ordering could give you a snapshot of how the jobs would run for the next n days. Hence fairshare appears to break that. However, looked at from a dynamic standpoint, fairshare is another element in the strict order.

9.15.11 Viewing and Managing Fairshare Data

The `pbsfs` command provides a command-line tool for viewing and managing some fairshare data. You can display the tree in tree form or in list form. You can print all information about an entity, or set an entity's usage to a new value. You can force an immediate decay of all the usage values in the tree. You can compare two fairshare entities. You can also remove all entities from the `unknown` department. This makes the tree easier to read. The tree can become unwieldy because entities not listed in the file `PBS_HOME/sched_priv/resource_group` all land in the `unknown` group.

The fairshare usage data is written to the file `PBS_HOME/sched_priv/usage` at an interval set in the scheduler configuration parameter `sync_time`. The default interval is one hour. To have the scheduler write out usage date prior to being killed, issue a **`kill -HUP`**. Otherwise, any usage data acquired since the last write will be lost.

See the `pbsfs(8B)` manual page for more information on using the `pbsfs` command.

9.15.12 Caveats

Do not use fairshare with the combination of strict_ordering and backfilling.

9.16 Enabling Strict Priority

Not to be confused with fairshare (which considers past usage of each entity in the selection of jobs), the scheduler offers a sorting key called "`fair_share_perc`" (see also section 9.3 "Scheduler Configuration Parameters" on page 315). Selecting this option enables the sorting of jobs based on the priorities specified in the fairshare 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 fairshare 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
```

9.17 Enabling Peer Scheduling

PBS Professional includes a feature to have different PBS complexes automatically run jobs from each other's queues. This provides the facility to dynamically load-balance across multiple, separate PBS complexes. These cooperating PBS complexes are referred to as "Peers". In peer scheduling, PBS server A pulls jobs from one or more Peer Servers and runs them locally. When Complex A pulls a job from Complex B, Complex A is the "pulling" complex and Complex B is the "furnishing" complex. When the pulling Scheduler determines that another complex's job can immediately run locally, it will move the job to the specified queue on the pulling Server and immediately run the job. A job is pulled only when it can run immediately.

You can set up peer scheduling so that A pulls from B and C, and so that B also pulls from A and C.

9.17.1 Prerequisites for Peer Scheduling

The pulling and furnishing queues must be created before peer scheduling can be configured. See section 7.6.2 "Creating Queues" on page 197 on how to create queues.

When configuring Peer Scheduling, it is *strongly* recommended to use the same version of PBS Professional at all Peer locations.

Under Windows, if `single_signon_password_enable` is set to "true" among all peer Servers, then users must have their password cached on each Server. For details see section 7.15.3 "Single Signon and Peer Scheduling" on page 241.

9.17.2 Configuring for Peer Scheduling

To configure your complex for peer scheduling, you must:

Define a flat user namespace on all complexes
Map pulling queues to furnishing queues
Grant manager access to each pulling server
If possible, make user-to-group mappings be consistent across complexes

These steps are described next.

9.17.2.1 Defining a Flat User Namespace

Peer Scheduling requires a flat user namespace in all complexes involved. This means that user "joe" on the remote Peer system(s) must be the same as user "joe" on the local system. Your site must have the same mapping of user to UID across all hosts, and a one-to-one mapping of UIDs to user names. It means that PBS does not need to check whether X@hostA is the same as X@hostB; it can just assume that this is true. Set `flatuid` to true:

Qmgr: **set server flatuid = true**

9.17.2.2 Mapping Pulling Queues to Furnishing Queues

You configure for peer scheduling by mapping a furnishing Peer's queue to a pulling Peer's queue. You can map a pulling queue to more than one furnishing queue, or more than one pulling queue to a furnishing queue.

The pulling and furnishing queues must be *execution* queues, not *route* queues. However, the queues can be either ordinary queues that the complex uses for normal work, or special queues set up just for peer scheduling.

You map pulling queues to furnishing queues by setting the `peer_queue` scheduler configuration option in `PBS_HOME/sched_priv/sched_config`. The format is:

```
peer_queue: "<pulling queue>
     <furnishing queue>@<furnishing server>.domain"
```

For example, Complex A's queue "workq" is to pull from Complex B's queue "workq", as well as Complex C's queue "slowq". Complex B's server is ServerB and Complex C's server is ServerC. You would add this to Complex A's `PBS_HOME/sched_priv/sched_config`:

```
peer_queue: "workq workq@ServerB.domain.com"
```

```
peer_queue: "workq slowq@ServerC.domain.com"
```

Or if you wish to direct Complex B's jobs to queue Q1 on Complex A, and Complex C's jobs to Q2 on Complex A:

```
peer_queue: "Q1 workq@ServerB.domain.com"
peer_queue: "Q2 fastq@ServerC.domain.com"
```

In one complex, you can create up to 50 mappings between queues. This means that you can have up to 50 lines in `PBS_HOME/sched_priv/sched_config` beginning with "peer_queue".

9.17.2.3 Granting Manager Access to Pulling Servers

Each furnishing Peer Server must grant manager access to each pulling Server. If you wish jobs to move in both directions, where Complex A will both pull from and furnish jobs to Complex B, ServerA and ServerB must grant manager access to each other.

On the furnishing complex:
For UNIX:

Qmgr: **set server managers += root@*pullingServer.domain.com***

For Windows:

Qmgr: **set server managers += pbsadmin@***

9.17.2.4 Making User-to-group Mappings Consistent Across Complexes

If possible, ensure that for each user in a peer complex, that user is in the same group in all participating complexes. So if user "joe" is in groupX on Complex A, user "joe" should be in groupX on Complex B. This means that a job's `egroup` attribute will be the same on both complexes, and any group limit enforcement can be properly applied.

There is a condition when using Peer Scheduling in which group hard limits may not be applied correctly. This can occur when a job's effective group, which is its `egroup` attribute, i.e. the job's owner's group, is different on the furnishing and pulling systems. When the job is moved over to the pulling complex, it can evade group limit enforcement if the group under which it will run on the pulling system has not reached its hard limit. The reverse is also true; if the group under which it will run on the pulling system has already reached its hard limit, the job won't be pulled to run, although it should.

This situation can also occur if the user explicitly specifies a group via `qsub -W group_list`.

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.

9.17.3 Peer Scheduling and Failover Configuration

If you are configuring peer scheduling so that Complex A will pull from Complex B where Complex B is configured for failover, you must configure Complex A to pull from both of Complex B's servers.

For example, the furnishing servers are ServerB1 and ServerB2, the furnishing queues are both called workq, and the pulling server's queue is pull_queue. Configure complex A's peer_queue setting in `PBS_HOME/sched_priv/sched_config` this way:

```
peer_queue: "pull_queue workq@ServerB1.example.com"
peer_queue: "pull_queue workq@ServerB2.example.com"
```

9.17.4 Jobs That Have Been Moved to Another Server

Since the Scheduler maps the remote jobs to its own 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 from local jobs, this can be done on the queue level. A queue can be created exclusively for remote jobs to allow queue level policy to be set for remote jobs. For example, you can set a priority value for each queue, and enable sorting by priority to ensure that pulled jobs are always lower (or higher!) priority than locally submitted jobs. For example, this means that if the local queue for pulled jobs has lower priority, the pulling complex will only pull a job when there are no higher-priority jobs that can run.

If you are connected to ServerA and a job submitted to ServerA has been moved from ServerA to ServerB through peer scheduling, in order to display it via `qstat`, give the job ID as an argument to `qstat`. If you only give the `qstat` command, the job will not appear to exist. For example, the job `123.ServerA` is moved to ServerB. In this case, use

```
qstat 123
```

or

```
qstat 123.ServerA
```

To list all jobs at ServerB, you can use:

```
qstat @ServerB
```

9.18 Using strict_ordering

With strict_ordering, all jobs on the server are considered as a group. This is different from considering first the jobs in one queue, then the jobs in another queue.

With strict_ordering, (sorting at the server level) if there are two queues, and each queue has one starving job and one lower-priority job, those two starving jobs as a group will go ahead of the two lower-priority jobs.

If the jobs were sorted at the queue level, then the starving job in one queue would go, followed by the lower-priority job in that queue, _then_ the starving job in the other queue would go, followed by the other lower-priority job.

Queue A jobs: StarveA, LowPriA
Queue B jobs: StarveB, LowPriB

Order of jobs when sorting at server level:
StarveA & StarveB (in some order like job submission)
LowPriA & LowPriB (ditto)

Order of jobs when sorting at the queue level:
StarveA
LowPriA
StarveB
LowPriB

9.18.1 Enabling FIFO Scheduling with strict_ordering

True first-in, first-out (FIFO) scheduling means sorting jobs into the order submitted, and then running jobs in that order. Furthermore, it means that when the Scheduler reaches a job in the sorted list that cannot run, then no other jobs will be considered until that job can run. In many situations, this results in an undesirably low level of system utilization. However, some customers have a job-mix or a usage policy for which FIFO scheduling is appropriate. When strict_ordering is used, it orders jobs according to the table in section 9.7 “Job Priorities in PBS Professional” on page 340.

Because true FIFO runs counter to many of the efficiency algorithms in PBS Professional, several options must be set in order to achieve true FIFO scheduling within a given queue. In order to have jobs within individual queues be run in true FIFO order, set the following parameters to the indicated values in the Scheduler’s configuration file:

```
strict_ordering:     True   ALL
round_robin:         False  ALL
job_sort_key:        False  ALL
fairshare            False  ALL
help_starving_jobs   False  ALL
backfill:            False  ALL
```

If you are using a single execution queue, you can have true FIFO scheduling for your jobs. You can give priority to queues and have FIFO on all the jobs in the complex in the order in which the queues are sorted.

9.18.2 Combining `strict_ordering` and Backfilling

Strict ordering can be combined with backfilling. If the next job in the ordering cannot run, jobs can be backfilled around the job that cannot run. Note that this is not precisely FIFO anymore.

9.18.3 Caveats

It is inadvisable to use `strict_ordering` and `backfill` with fairshare. The results may be non-intuitive. Fairshare will cause relative job priorities to change with each scheduling cycle. It is possible that a job from the same entity or group will be chosen as the small job. The usage from these small jobs will lower the priority of the most deserving job.

Using dynamic resources with strict_ordering and backfilling may result in unpredictable scheduling. See "Backfilling Caveats" on page 369.

Using preemption with strict_ordering and backfilling may change which job is being backfilled around.

9.19 Starving Jobs

If the `help_starving_jobs` parameter is set to True, jobs become *starving* when they have remained queued beyond a certain amount of time. These jobs are assigned the priority level of starving. Therefore these jobs will have higher priority according to the scheduler's standard sorting order. See section 9.7 "Job Priorities in PBS Professional" on

page 340. In addition, the order in which starving jobs can preempt other jobs or be preempted is set via the `preempt_prio` configuration option. See “preempt_prio” on page 320.

When a job is running, it keeps the starving status it had when it was started. While a job is running, if it wasn’t starving before, it can’t become starving. However, it keeps its starving status if it became starving while queued.

Subjobs that are queued can become starving. Starving status is applied to individual subjobs in the same way it is applied to jobs. The queued subjobs of a job array can become starving while others are running. If a job array has starving subjobs, then the job array is starving.

The `max_starve` parameter sets the amount of time a job must be queued before it can become starving. The default time period to become starving is 24 hours.

Jobs lose their starvingness whenever they are requeued, as with the `qrerun` command. This includes when they are checkpointed or requeued (but not suspended) during preemption. Suspended jobs do not lose their starving status. However, when they become suspended, the amount of time since they were submitted is counted towards being starving. For example, if a job was submitted, then remained queued for 1 hour, then ran for 26 hours, then was suspended, if max_starve is 24 hours, then the job will become starving.

9.20 Using Backfilling

Backfilling means fitting smaller jobs around the jobs that the scheduler was going to run anyway. Backfilling is only used around starving jobs and with `strict_ordering`. The scheduler keeps track of which job is due to run next (the “most deserving job”) according to the policy that has been set, but in addition, it looks for the next job according to policy where that job is also small enough to fit in the available slot (the “small job”). It runs the small job as long as that won’t change the start time of the most deserving job due to run next.

The scheduler recalculates everything at each scheduling cycle, so the most deserving job and the small job may change from one cycle to the next.

When `strict_ordering` is on, the scheduler chooses the next job in the standard order. The scheduler also chooses its small job in the standard order. See section 9.7 “Job Priorities in PBS Professional” on page 340

The configuration parameters `backfill_prime` and `prime_exempt_anytime_queues` do not relate to backfilling. They control the time boundaries of regular jobs with respect to primetime and non-primetime.

9.20.0.1 Backfilling Caveats

Using dynamic resources and backfilling may result in some jobs not being run even though resources are available. This may happen when a job requesting a dynamic resource is selected as the most deserving job. The scheduler must estimate when resources will become available, but it can only query for available resources, not resources already in use, so it will not be able to predict when resources in use become available. Therefore the scheduler won't be able to schedule the job. In addition, since dynamic resources are outside of the control of PBS, they may be consumed between the time the scheduler queries for the resource and the time it starts a job.

Chapter 10

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.

10.1 Overview of Custom Resource Types

Custom resources can be static or dynamic. Dynamic custom resources can be defined at the server or host. Static custom resources are defined ahead of time, at the server, queue or vnode. Custom resources are defined to the server, then set on one or more vnodes.

For static custom resources the Server maintains the status of the custom resource, and the Scheduler queries the Server for the resource. Static custom resource values at vnode, queue and server can be established via qmgr, setting resources_available.<custom resource name> = <some value>.

For dynamic server-level custom resources the scheduler uses a script to get resource availability. The `script` needs to report the amount of the resource to the Scheduler via `stdout`, in a single line ending with a newline.

For dynamic host-level custom resources, the Scheduler will send 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.

For a dynamic host-level resource, values are established by a MOM directive which defines a script which returns a dynamic value via stdout when executed. For a dynamic server-level custom resource, the value is established by the script defined in the `server_dyn_res` line in `PBS_HOME/sched_priv/sched_config`.

For information on resources shared across vnodes, see "Vnodes and Shared Resources" on page 220.

10.1.1 Custom Resource Formats

The names of custom numeric resources must be alphanumeric with a leading alphabetic: [a-zA-Z][a-zA-Z0-9_]*. Allowable values for float and long resources are the same as for built-in resources. Custom boolean, time, size, string or string array resources must have the same format as built-in resources. See section 7.9.7 "Resource Types" on page 223.

10.2 How to Use Custom Resources

10.2.1 Choosing Dynamic or Static, Server or Host

Use dynamic resources for quantities that PBS does not control, such as externally-managed licenses or scratch space. PBS runs a script or program that queries an external source for the amount of the resource available and returns the value via stdout. Use static resources for things PBS does control, such as licenses managed by PBS. PBS tracks these resources internally.

Use server-level resources for things that are not tied to specific hosts, that is, they can be available to any of a set of hosts. An example of this is a floating license. Use host-level resources for things that are tied to specific hosts, like the scratch space on a machine or node-locked licenses.

10.2.2 Using Custom Resources for Application Licenses

The following table lists application licenses and what kind of custom resource to define for them. For specific instructions on configuring each type of license, see examples of configuring custom resources for application licenses in section 10.7 "Application Licenses" on page 388.

Table 20: Custom Resources for Application Licenses

Floating or Node-locked	Unit Being Licensed	How License is Managed	Level	Resource Type
Floating (site-wide)	Token	External license manager	Server	Dynamic
Floating (site-wide)	Token	PBS	Server	Static
Node-locked	Host	PBS	Host	Static
Node-locked	CPU	PBS	Host	Static
Node-locked	Instance of Application	PBS	Host	Static

10.2.3 Using Custom Resources for Scratch Space

You can configure a custom resource to report how much scratch space is available on machines. Jobs requiring scratch space can then be scheduled onto machines which have enough. This requires dynamic host-level resources. See section 10.6 "Scratch Space" on page 387 and section 10.4.1 "Dynamic Host-level Resources" on page 380.

10.2.3.1 Dynamic Resource Scripts/Programs

You create the script or program that PBS uses to query the external source. The external source can be a license manager or a command, as when you use the `df` command to find the amount of available disk space. If the script is for a server-level dynamic resource, it is placed on the server. The script must be available to the scheduler, which runs the script. If you have set up peer scheduling, make sure that the script is available to any scheduler that must run it. If it is for a host-level resource, it is placed on the host(s) where it will be used. The script must return its output via stdout, and the output must be in a single line ending with a newline.

In Windows, if you use Notepad to create the script, be sure to explicitly put a newline at the end of the last line, otherwise none will appear, causing PBS to be unable to properly parse the file.

10.2.4 Relationship Between Hosts, Nodes, and Vnodes

A host is any computer. Execution hosts used to be called nodes. However, some machines such as the Altix can be treated as if they are made up of separate pieces containing CPUs, memory, or both. Each piece is called a vnode. See "Vnodes: Virtual Nodes" on page 205. Some hosts have a single vnode and some have multiple vnodes. PBS treats all vnodes alike in most respects. Chunks cannot be split across hosts, but they can be split across vnodes on the same host.

Resources that are defined at the host level are applied to vnodes. If you define a dynamic host-level resource, it will be shared among the vnodes on that host. This sharing is managed by the MOM. If you define a static host-level resource, you can set its value at each vnode, or you can set it on one vnode and make it indirect at other vnodes. See "Vnodes and Shared Resources" on page 220.

10.3 Defining New Custom Resources

To define one or more new resources, the Administrator creates or updates the Server resource definition file, *PBS_HOME*/server_priv/resourcedef. Each line in the file defines a new resource.

Once you have defined the new resource(s), you must restart the Server in order for these changes to take effect (see section 10.3.4 on page 379). When the Server restarts, users will be able to submit jobs requesting the new resource, using the normal syntax to which they are accustomed. See also section 10.6 "Scratch Space" on page 387 and section 10.7 "Application Licenses" on page 388.

10.3.1 The `resourcedef` File

The format of each line in *PBS_HOME*/server_priv/resourcedef is:

```
RESOURCE_NAME [type=RTYPE] [flag=FLAGS]
```

RESOURCE_NAME is any string made up of alphanumeric characters, where the first character is alphabetic. Resource names must start with an alphabetic character and can contain alphanumeric, underscore ("_"), and dash ("-") characters.

If a string resource value contains spaces or shell metacharacters, enclose the string in quotes, or otherwise escape the space and metacharacters. Be sure to use the correct quotes for your shell and the behavior you want. If the string resource value contains commas, the string must be enclosed in an additional set of quotes so that the command (e.g. qsub, qalter) will parse it correctly. If the string resource value contains quotes, plus signs, equal signs, colons or parentheses, the string resource value must be enclosed in yet another set of additional quotes.

The length of each line in `PBS_HOME/server_priv/resourcedef` file should not be more than 254 characters. There is no limit to the number of custom resources that can be defined.

RTYPE is the type of the resource value, which can be one of the following keywords, or will default to `long`.

See "Resource Types" on page 223 for a description of each resource type. See "Resource Flags" on page 225 for a description of how resource flags are used.

10.3.2 Defining and Using a Custom Resource

In order for jobs to use a new custom resource, the resource must be:

Step 1 Defined to the server in the server's `resourcedef` file

Step 2 Put in the "resources" line in .`PBS_HOME/sched_priv/sched_config`

Step 3 Set either via `qmgr` or by adding it to the correct configuration line

Step 4 If the resource is dynamic, it must be added to the correct line in the scheduler's configuration file: if it's a host -level dynamic resource, it must be added to the `mom_resources` line, and if it's a server-level resource, it must be added to the `server_dyn_res` line

If the resource is not put in the scheduler's "resources" line, when jobs request the resource, that request will be ignored. If the resource is ignored, it cannot be used to accept or reject jobs at submission time. For example, if you create a string String1 on the server, and set it to "foo", a job requesting "-l String1=bar" will be accepted.

Depending on the type of resource, the server, scheduler and MOMs must be restarted. For detailed steps, see "Configuring Host-level Custom Resources" on page 380 and "Configuring Server-level Resources" on page 385.

10.3.2.1 Example of Defining Each Type of Custom Resource

In this example, we add five custom resources: a static and a dynamic host-level resource, a static and a dynamic server-level resource, and a static queue-level resource.

1. The resource must be defined to the server, with appropriate flags set:
   Add resource to `PBS_HOME/server_priv/resourcedef`

```
staticserverresource     type=long flag=q
statichostresource     type=long flag=nh
dynamicserverresource     type=long
dynamichostresource     type=long flag=h
staticqueueresource     type=long flag=q
```

2. The resource must be added to the scheduler's list of resources:
   Add resource to "resources" line in
   `PBS_HOME/sched_priv/sched_config`

```
resources:  "staticserverresource,statichostresource,\
   dynamicserverresource, dynamichostresource, \
   staticqueueresource"
```

3. If the resource is static, use `qmgr` to set it at the host, queue or server level.

```
Qmgr: set node Host1 \
   resources_available.statichostresource=1
Qmgr: set queue Queue1 \
   resources_available.staticqueueresource=1
Qmgr: set server \
   resources_available.staticserverresource=1
```

   See "The qmgr Command" on page 173.

4. If the resource is dynamic:
   a. If it's a host-level resource, add it to the "mom_resources" line in

PBS_HOME/sched_priv/sched_config:
mom_resources: dynamichostresource
Also add it to the MOM config file PBS_HOME/mom_priv/config:
dynamichostresource !path-to-command

b. If it's a server-level resource, add it to the "server_dyn_res" line in PBS_HOME/sched_priv/sched_config:
server_dyn_res: "dynamicserverresource !path-to-command"

.

Table 21: Adding Custom Resources

Resource Type	Server-level	Queue-level	Host-level
static	Set via qmgr	Set via qmgr	Set via qmgr
dynamic	Add to server_dyn_res line in PBS_HOME/sched_priv/sched_config	Cannot be used.	Add to MOM config file PBS_HOME/mom_priv/config and mom_resources line in PBS_HOME/sched_priv/sched_config

10.3.2.2 Discussion of Scheduling Custom Resources

The last step in creating a new custom resource is configuring the Scheduler to (a) query your new resource, and (b) include the new resource in each scheduling cycle. Whether you set up server-level or host-level 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 server-level custom resource, add the server_dyn_res configuration parameter to the Scheduler *PBS_HOME/*sched_priv/sched_config file:

```
server_dyn_res: "RESOURCE_NAME !path-to-command"
```

where RESOURCE_NAME should be the same as used in the Server's *PBS_HOME/*server_priv/resourcedef file. (See also section 9.3 "Scheduler Configuration Parameters" on page 315).

To direct the Scheduler to use a new dynamic host-level 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 that in the Server's resourcedef file and the MOM's config file. (see also section 8.2.2 "Syntax and Contents of Default Configuration File" on page 260).

Next, tell the Scheduler to include the custom resource as a constraint in each scheduling cycle by appending the new resource to the resources configuration parameter in the Scheduler sched_config file:

```
resources: "ncpus, mem, arch, RESOURCE_NAME"
```

Examples are provided in section 10.6 "Scratch Space" on page 387 and section 10.7 "Application Licenses" on page 388.

Once you have defined the new resource(s), you must restart/reinitialize the Scheduler in order for these changes to take effect (see section 10.3.4 on page 379).

10.3.3 Getting an Accurate Picture of Available Resources

Because some 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 and use that resource, only to have an outside entity take that resource before the job is able to use it.

For example, say you had an external resource of "scratch space" and your local query script simply checked to see how much disk space was free. It would be possible for a job to be started on a host with the requested space, but for another application to use the free space before the job did.

10.3.4 PBS Restart Steps for Custom Resources

In order to have new custom resources recognized by PBS, the individual PBS components must either be restarted or reinitialized for the changes to take effect. The subsequent sections of this chapter will indicate when this is necessary, and refer to the details of this section for the actual commands to type.

The procedures below apply to the specific circumstances of defining custom resources. For general restart procedures, see section 11.4 "Starting and Stopping PBS: UNIX and Linux" on page 405 and section 11.5 "Starting and Stopping PBS: Windows 2000 / XP" on page 421.

Server restart procedures are:

On UNIX:
```
qterm -t quick
PBS_EXEC/sbin/pbs_server
```

On Windows:
Admin> **qterm -t quick**
Admin> **net start pbs_server**

MOM restart / reinitialization procedures are:

On UNIX: Use the "`ps`" command to determine the process ID of current instance of PBS MOM, and then terminate MOM via `kill` using the PID returned by `ps`. Note that `ps` arguments vary among UNIX systems, thus "`-ef`" may need to be replaced by "`-aux`". Note that if your custom resource gathering script/program takes longer than the default ten seconds, you can change the alarm timeout via the `-a alarm` command line start option as discussed in section 11.4.4 "Manually Starting MOM" on page 407. You will typically want to use the -p option when starting MOM:

```
ps –ef | grep pbs_mom
kill -HUP <MOM PID>
PBS_EXEC/sbin/pbs_mom -p
```

On Windows:
```
Admin> net stop pbs_mom
Admin> net start pbs_mom
```

If your custom resource gathering script/program takes longer than the default ten seconds, you can change the alarm timeout via the `-a alarm` command line start option as discussed in section 11.5.1 “Startup Options to PBS Windows Services” on page 422.)

Scheduler restart / reinitialization procedures are:

On UNIX:
```
ps –ef | grep pbs_sched
kill -HUP <Scheduler PID>
PBS_EXEC/sbin/pbs_sched
```

On Windows:
```
Admin> net stop pbs_sched
Admin> net start pbs_sched
```

10.4 Configuring Host-level Custom Resources

Host-level custom resources can be static and consumable, static and not consumable, or dynamic. Dynamic host-level resources are used for things like scratch space.

10.4.1 Dynamic Host-level Resources

A dynamic resource could be scratch space on the host. The amount of scratch space is determined by running a script or program which returns the amount via stdout. This script or program is specified in the `mom_resources` line in `PBS_HOME/sched_priv/sched_config.`

These are the steps for configuring a dynamic host-level resource:

Step 1 Write a script, for example `hostdyn.pl`, that returns the available amount of the resource via stdout, and place it on each host where it will be used. For example, it could be placed in `/usr/local/bin/hostdyn.pl`

Step 2 Configure each MOM to use the script by adding the resource

and the path to the script in `PBS_HOME/mom_priv/config`.

```
dynscratch !/usr/local/bin/hostdyn.pl
```

Step 3 Restart the MOMs. See section 10.3.4 "PBS Restart Steps for Custom Resources" on page 379.

Step 4 Define the resource, for example dynscratch, in the server resource definition file *`PBS_HOME`*`/server_priv/resourcedef`.

```
dynscratch type=size flag=h
```

Step 5 Restart the server. See section 10.3.4 "PBS Restart Steps for Custom Resources" on page 379.

Step 6 Add the new resource to the "resources" line in `PBS_HOME/sched_priv/sched_config`.

```
resources: "ncpus, mem , arch, dynscratch"
```

Step 7 Restart the scheduler. See section 10.3.4 "PBS Restart Steps for Custom Resources" on page 379.

Step 8 Add the new resource to the "mom_resources" line in PBS_HOME/sched_priv/sched_config. Create the line if necessary.

```
mom_resources: "dynscratch"
```

To request this resource, the resource request would include

```
-l select=1:ncpus=N:dynscratch=10MB
```

See section 10.6.1 "Host-level "scratchspace" Example" on page 387 for a more complete discussion of dynamic host-level resources.

The script must return, via stdout, the amount available in a single line ending with a newline.

10.4.1.1 Discussion of Dynamic Host-level Resources

If the new resource you are adding is a dynamic host-level resource, configure each MOM to answer the resource query requests from the Scheduler.

Each MOM can be instructed in how to respond to a Scheduler resource query by adding a shell escape to the MOM configuration file *PBS_HOME*/mom_priv/config. The shell escape provides a means for MOM to send information to the Scheduler. The format of a shell escape line is:

```
RESOURCE_NAME !path-to-command
```

The RESOURCE_NAME specified should be the same as the corresponding entry in the Server's *PBS_HOME*/server_priv/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.

On Windows, be sure to place double-quote (" ") marks around the *path-to-command* if it contains any whitespace characters.

Typically, what follows the shell escape (i.e. "!") 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. Once the shell escape script/program is started, MOM waits for output. The wait is by default ten seconds, but can be changed via the -a alarm command line start option. (For details of use, see section 11.4.4 "Manually Starting MOM" on page 407 and section 11.5.1 "Startup Options to PBS Windows Services" on page 422.) If the alarm time passes and the shell escape process has not finished, a log message, "resource read alarm" is written to the MOM's log file. The process is given another alarm period to finish and if it does not, an error is returned, usually to the scheduler, in the form of "? 15205". Another log message is written. The ? indicates an error condition and the value 15205 is PBSE_RMSYSTEM. The user's job may not run.

In order for the changes to the MOM config file to take effect, the pbs_mom process must be either restarted or reinitialized (see section 10.3.4 on page 379). For an example of configuring scratch space, see section 10.6.1 "Host-level "scratchspace" Example" on page 387.

10.4.2 Static Host-level Resources

Use static host-level resources for node-locked application licenses managed by PBS, where PBS is in full control of the licenses. These resources are "static" because PBS tracks them internally, and "host-level" because they are tracked at the host.

Node-locked application licenses can be per-host, where any number of instances can be running on that host, per-CPU, or per-run, where one license allows one instance of the application to be running. Each kind of license needs a different form of custom resource.

If you are configuring a custom resource for a per-host node-locked license, where the number of jobs using the license does not matter, use a host-level boolean resource on the appropriate host. This resource is set to True. When users request the license, they can use:

For a two-CPU job on a single vnode:

```
-l select=1:ncpus=2:license=1
```

For a multi-vnode job:

```
-l select=2:ncpus=2:license=1 -l place=scatter
```

Users can also use "license=True", but this way they do not have to change their scripts.

If you are configuring a custom resource for a per-CPU node-locked license, use a host-level consumable resource on the appropriate vnode. This resource is set to the maximum number of CPUs you want used on that vnode. Then when users request the license, they will use:

For a two-CPU, two-license job:

-l select=1:ncpus=2:license=2

If you are configuring a custom resource for a per-use node-locked license, use a host-level consumable resource on the appropriate host. This resource is set to the maximum number of of instances of the application allowed on that host. Then when users request the license, they will use:

For a two-CPU job on a single host:

-l select=1:ncpus=2:license=1

For a multi-vnode job where vnodes need two CPUs each:

-l select=2:ncpus=2:license=1 -l place=scatter

The rule of thumb is that the chunks have to be the size of a single host so that one license in the chunk corresponds to one license being taken from the host.

These are the steps for configuring a static host-level resource:

Step 1 Define the resource, for example hostlicense, in the server resource definition file *PBS_HOME*/server_priv/resourcedef.

For per-CPU or per-use:
```
hostlicense type=long flag=nh
```

For per-host:
```
hostlicense type=boolean flag=h
```

Step 2 Restart the server. See section 10.3.4 “PBS Restart Steps for Custom Resources” on page 379.

Step 3 Use the qmgr command to set the value of the resource on the host.

Qmgr: set node Host1 hostlicense=(number of uses, number of CPUs, or True if boolean)

Step 4 Add the new resource to the “resources” line in PBS_HOME/sched_priv/sched_config.
resources: “ncpus, mem , arch, hostlicense”

Step 5 Restart the scheduler. See section 10.3.4 “PBS Restart Steps for Custom Resources” on page 379.

For examples of configuring each kind of node-locked license, see section 10.7.6 “Per-host Node-locked Licensing Example” on page 395, section 10.7.7 “Per-use Node-locked Licensing Example” on page 397, and section 10.7.8 “Per-CPU Node-locked Licensing Example” on page 400.

10.5 Configuring Server-level Resources

10.5.1 Dynamic Server-level Resources

Dynamic server-level resources are usually used for site-wide externally-managed floating licenses. The availability of licenses is determined by running a script or program specified in the `server_dyn_res` line of `PBS_HOME/sched_priv/sched_config`. The script must return the value via stdout in a single line ending with a newline. For a site-wide externally-managed floating license you will need two resources: one to represent the licenses themselves, and one to mark the vnodes on which the application can be run. The first is a server-level dynamic resource and the second is a host-level boolean, set on the vnodes to send jobs requiring that license to those vnodes.

These are the steps for configuring a dynamic server-level resource for a site-wide externally-managed floating license. If this license could be used on all vnodes, the boolean resource would not be necessary.

Step 1 Define the resources, for example floatlicense and CanRun, in the server resource definition file *PBS_HOME/*`server_priv/resourcedef`.

```
floatlicense type=long
CanRun type=boolean flag=h
```

Step 2 Write a script, for example serverdyn.pl, that returns the available amount of the resource via stdout, and place it on the server's host. For example, it could be placed in `/usr/local/bin/serverdyn.pl`

Step 3 Restart the server. See section 10.3.4 "PBS Restart Steps for Custom Resources" on page 379.

Step 4 Configure the scheduler to use the script by adding the resource and the path to the script in the `server_dyn_res` line of `PBS_HOME/sched_priv/sched_config`.

```
server_dyn_res: "floatlicense \
    !/usr/local/bin/serverdyn.pl"
```

Step 5 Add the new dynamic resource to the "resources" line in `PBS_HOME/sched_priv/sched_config`:

```
resources: "ncpus, mem , arch, \
    floatlicense"
```

Step 6 Restart the scheduler. See section 10.3.4 "PBS Restart Steps for Custom Resources" on page 379.

Step 7 Set the boolean resource on the vnodes where the floating licenses can be run. Here we designate vnode1 and vnode2 as the vnodes that can run the application:

```
Qmgr: active node vnode1,node2
Qmgr: set node resources_available.CanRun=True
```

To request this resource, the job's resource request would include

```
-l floatlicense=<number of licenses or tokens required>
-l select=1:ncpus=N:CanRun=1
```

See section 10.6.1 "Host-level "scratchspace" Example" on page 387 for more discussion of dynamic host-level resources.

10.5.2 Static Server-level Resources

Static server-level resources are used for floating licenses that PBS will manage. PBS keeps track of the number of available licenses instead of querying an external license manager.

These are the steps for configuring a static server-level resource:

Step 1 Define the resource, for example sitelicense, in the server resource definition file *PBS_HOME*`/server_priv/resourcedef.`

```
sitelicense type=long flag=q
```

Step 2 Restart the server. See section 10.3.4 "PBS Restart Steps for Custom Resources" on page 379.

Step 3 Use the qmgr command to set the value of the resource on the server.

Qmgr: set server sitelicense=(number of licenses)

Step 4 Add the new resource to the "resources" line in PBS_HOME/sched_priv/sched_config.

resources: "ncpus, mem , arch, sitelicense"

Step 5 Restart the scheduler. See section 10.3.4 "PBS Restart Steps for Custom Resources" on page 379.

10.6 Scratch Space

10.6.1 Host-level "scratchspace" Example

Say you have jobs that require a large amount of scratch disk space during their execution. To ensure that sufficient space is available when starting the job, you first write a script that returns via stdout a single line (with new-line) the amount of space available. This script is placed in `/usr/local/bin/scratchspace` on each host. 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 231.) For this example, let's call our new resource "scratchspace". We'll set `flag=h` so that users can specify a minimum amount in their select statements.

```
scratchspace    type=size flag=h
```

Now restart the Server (see section 10.3.4 on page 379).

Once the Server recognizes the new resources, you may optionally specify any limits on that resource via `qmgr`, such as the maximum amount available of the new resources, or the maximum that a single user can request. For example, at the `qmgr` prompt you could type:

```
set server resources_max.scratchspace=1gb
```

Next, configure MOM to use the `scratchspace` script by entering one line into the `PBS_HOME/mom_priv/config` file:

On UNIX:

```
scratchspace !/usr/local/bin/scratchspace
```

On Windows:

```
scratchspace !"c:\Program Files\PBS Pro\scratchspace"
```

Then, restart / reinitialize the MOM (see section 10.3.4 on page 379).

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: "ncpus, mem, arch, scratchspace"
```

Then, restart / reinitialize the Scheduler (see section 10.3.4 on page 379).

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 vnode on which to run this job.

10.7 Application Licenses

10.7.1 Types of Licenses

Application licenses may be managed by PBS or by an external license manager. Application licenses may be floating or node-locked, and they may be per-cpu, per-use or per-host.

Whenever an application license is managed by an external license manager, you must create a custom dynamic resource for it. This is because PBS has no control over whether these licenses are checked out, and must query the external license manager for the availability of those licenses. PBS does this by executing the script or program that you specify in the dynamic resource. This script returns the amount via stdout, in a single line ending with a newline.

When an application license is managed by PBS, you can create a custom static resource for it. You set the total number of licenses using qmgr, and PBS will internally keep track of the number of licenses available.

10.7.2 License Units and Features

Different licenses use different license units to track whether an application is allowed to run. Some licenses track the number of CPUs an application is allowed to run on. Some licenses use tokens, requiring that a certain number of tokens be available in order to run. Some licenses require a certain number of features to run the application.

When using units, after you have defined *license_name* to the server, be sure to set `resources_available.`*`license_name`* to the correct number of units.

Before starting you should 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 license manager to obtain the available licenses of particular features?

With these questions answered you can begin configuring PBS Professional to query the license manager servers for the availability of application licenses. Think of a license manager feature as a resource. Therefore, you should associate a resource with each feature.

10.7.3 Simple Floating License Example

Here is an example of setting up floating licenses that are managed by an external license server.

For this example, we have a 6-host complex, with one CPU per host. The hosts are numbered 1 through 6. On this complex we have one licensed application which uses floating licenses from an external license manager. Furthermore we want to limit use of the application only to specific hosts. The table below shows the application, the number of licenses, the hosts on which the licenses should be used, and a description of the type of license used by the application.

Application	Licenses	Hosts	DESCRIPTION
AppF	4	3-6	uses licenses from an externally managed pool

For the floating licenses, we will use two resources. One is a dynamic server resource for the licenses themselves. The other is a boolean resource used to indicate that the floating license can be used on a given host.

Server Configuration

1. Define the new resource in the Server's `resourcedef` file. Create a new file if one does not already exist by adding the resource names, type, and flag(s).

   ```
   cd $PBS_HOME/server_priv/
   [edit] resourcedef
   ```

   Example `resourcedef` file with new resources added:

   ```
   AppF type=long
   runsAppF type=boolean flag=h
   ```

2. Restart the Server (see section 10.3.4 on page 379).

Host Configuration

3. Set the boolean resource on the hosts where the floating licenses can be used.

   qmgr: **active node host3,host4,host5,host6**
   qmgr: **set node resources_available.runsAppF = True**

Scheduler Configuration

Edit the Scheduler configuration file.

```
cd $PBS_HOME/sched_priv/
[edit] sched_config
```

4. Append the new resource names to the "resources:" line:

   ```
   resources: "ncpus, mem, arch, host, AppF,
   runsAppF"
   ```

5. Edit the "server_dyn_res" line:

   UNIX:

   ```
   server_dyn_res: "AppF !/local/flex_AppF"
   ```

   Windows:

   ```
   server_dyn_res: "AppF !C:\Program Files\
                    PBS Pro\flex_AppF"
   ```

6. Restart or reinitialize the Scheduler (see section 10.3.4 on page 379).

To request a floating license for AppF and a host on which AppF can run:

```
qsub -l AppF=1
     -l select=runsAppF=True
```

The example below shows what the host configuration would look like. What is shown is actually truncated output from the pbsnodes -a command. Similar information could be printed via the qmgr -c "print node @default" command as well.

```
host1
host2
host3
     resources_available.runsAppF = 1
host4
     resources_available.runsAppF = 1
host5
```

```
     resources_available.runsAppF = 1
host6
     resources_available.runsAppF = 1
```

10.7.4 Example of Floating, Externally-managed License with Features

This is an example of a floating license, managed by an external license manager, where the application requires a certain number of features to run. Floating licenses are treated as server-level dynamic resources. The license server is queried by an administrator-created script. This script returns the value via stdout in a single line ending with a newline.

The license script runs on the server's host once per scheduling cycle and queries the number of available licenses/tokens for each configured application. When submitting a job, the user's script, in addition to requesting CPUs, memory, etc., also requests licenses. When the scheduler looks at all the enqueued jobs, it evaluates the license request alongside the request for physical resources, and if all the resource requirements can be met the job is run. If the job's token requirements cannot be met, then it remains queued.

PBS doesn't actually check out the licenses; the application being run inside the job's session does that. Note that a small number of applications request varying amounts of tokens during a job run.

A common question that arises among PBS Professional customers is regarding how to use the dynamic resources to coordinate external floating license checking for applications. The following example illustrates how to implement such a custom resource. Our example needs four features to run an application, so we 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
```

Important: Note that in the above example the optional FLAG (third column of the resourcedef file) is not shown because it is a server-level resource which is not consumable.

Once these resources have been defined, you will need to restart the PBS Server (see section 10.3.4 on page 379).

Now that PBS is aware of the new custom resources we can begin configuring the Scheduler to query the license manager server, and schedule based on the availability of the licenses.

Within *PBS_HOME*/sched_priv/sched_config the following parameters will need to be updated, or introduced depending on your site configuration. The 'resources:' parameter should already exist with some default PBS 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 parameter 'server_dyn_res which allows the Scheduler to execute a program or script, that will need to be created, to query your license manager server for available licenses. For example.

UNIX:

```
server_dyn_res: "feature1 !/path/to/script [args]"
server_dyn_res: "feature3 !/path/to/script [args]"
server_dyn_res: "feature6 !/path/to/script [args]"
server_dyn_res: "feature8 !/path/to/script [args]"
```

Windows:

```
server_dyn_res: "feature1 !C:\Program Files\PBS Pro\script [args]"
server_dyn_res: "feature3 !C:\Program Files\PBS Pro\script [args]"
server_dyn_res: "feature6 !C:\Program Files\PBS Pro\script [args]"
server_dyn_res: "feature8 !C:\Program Files\PBS Pro\script [args]"
```

Once the PBS_HOME/sched_priv/sched_config has been updated, you will need to restart/reinitialize the pbs_sched process.

Essentially, the provided 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 an N CPU application.

10.7.5 Example of Floating License Managed by PBS

Here is an example of configuring custom resources for a floating license that PBS manages. For this you need a server-level static resource to keep track of the number of available licenses. If the application can only run on certain hosts, then you will need a host-level boolean resource to direct jobs running the application to the correct hosts.

In this example, we have six hosts numbered 1-6, and the application can run on hosts 3, 4, 5 and 6. The resource that will track the licenses is called AppM. The boolean resource is called RunsAppM.

Server Configuration

1. Define the new resource in the Server's `resourcedef` file. Create a new file if one does not already exist by adding the resource names, type, and flag(s).

   ```
   cd $PBS_HOME/server_priv/
   [edit] resourcedef
   ```

   Example `resourcedef` file with new resources added:

   ```
   AppM type=long flag=q
   runsAppM type=boolean flag=h
   ```

2. Restart the Server (see section 10.3.4 on page 379).

Host Configuration

3. Set the value of runsAppM on the hosts. (Ensure that each `qmgr` directive is typed on a single line.)

   ```
   qmgr: active node host3,host4,host5,host6
   qmgr: set node \
       resources_available.runsAppM = True
   ```

Scheduler Configuration

Edit the Scheduler configuration file.

```
cd $PBS_HOME/sched_priv/
[edit] sched_config
```

4. Append the new resource name to the "`resources:`" line.

   `resources: "ncpus, mem, arch, host, `**`AppM, runsAppM`**`"`

5. Restart or reinitialize the Scheduler (see section 10.3.4 on page 379).

To request both the application and a host that can run AppM:

```
qsub -l AppM=1
     -l select=1:runsAppM=1 <jobscript>
```

The example below shows what the host configuration would look like. What is shown is actually truncated output from the `pbsnodes -a` command. Similar information could be printed via the `qmgr -c "print node @default"` command as well. Since unset boolean resources are the equivalent of False, you do not need to explicitly set them to False on the other hosts. Unset Boolean resources will not be printed.

```
host1

host2

host3
     resources_available.runsAppM = True
host4
     resources_available.runsAppM = True
host5
     resources_available.runsAppM = True
host5
     resources_available.runsAppM = True
```

10.7.6 Per-host Node-locked Licensing Example

Here is an example of setting up node-locked licenses where one license is required per host, regardless of the number of jobs on that host.

For this example, we have a 6-host complex, with one CPU per host. The hosts are numbered 1 through 6. On this complex we have a licensed application that uses per-host node-locked licenses. We want to limit use of the application only to specific hosts. The table below shows the application, the number of licenses for it, the hosts on which the licenses should be used, and a description of the type of license used by the application.

Application	Licenses	Hosts	DESCRIPTION
AppA	1	1-4	uses a local node-locked application license

For the per-host node-locked license, we will use a boolean host-level resource called resources_available.runsAppA. This will be set to True on any hosts that should have the license, and will default to False on all others. The resource is not consumable so that more than one job can request the license at a time.

Server Configuration

1. Define the new resource in the Server's `resourcedef` file. Create a new file if one does not already exist by adding the resource names, type, and flag(s).

   ```
   cd $PBS_HOME/server_priv/
   [edit] resourcedef
   ```

   Example `resourcedef` file with new resources added:

   ```
   runsAppA type=boolean flag=h
   ```

2. Restart the Server (see section 10.3.4 on page 379).

Host Configuration

3. Set the value of runsAppA on the hosts. (Ensure that each `qmgr` directive is typed on a single line.)

   qmgr: **active node host1,host2,host3,host4**
   qmgr: **set node resources_available.runsAppA = True**

Scheduler Configuration

Edit the Scheduler configuration file.

```
cd $PBS_HOME/sched_priv/
[edit] sched_config
```

4. Append the new resource name to the "`resources:`" line.

   `resources: "ncpus, mem, arch, host, AppA"`

5. Restart or reinitialize the Scheduler (see section 10.3.4 on page 379).

To request a host with a per-host node-locked license for AppA:

```
qsub -l select=1:runsAppA=1 <jobscript>
```

The example below shows what the host configuration would look like. What is shown is actually truncated output from the `pbsnodes -a` command. Similar information could be printed via the `qmgr -c "print node @default"` command as well. Since unset boolean resources are the equivalent of False, you do not need to explicitly set them to False on the other hosts. Unset Boolean resources will not be printed.

```
host1
     resources_available.runsAppA = True
host2
     resources_available.runsAppA = True
host3
     resources_available.runsAppA = True
host4
     resources_available.runsAppA = True
host5

host6
```

10.7.7 Per-use Node-locked Licensing Example

Here is an example of setting up per-use node-locked licenses. Here, while a job is using one of the licenses, it is not available to any other job.

For this example, we have a 6-host complex, with 4 CPUs per host. The hosts are numbered 1 through 6. On this complex we have a licensed application that uses per-use node-locked licenses. We want to limit use of the application only to specific hosts. The licensed

hosts can run two instances each of the application. The table below shows the application, the number of licenses for it, the hosts on which the licenses should be used, and a description of the type of license used by the application.

Application	Licenses	Hosts	DESCRIPTION
AppB	2	1-2	uses a local node-locked application license

For the node-locked license, we will use one static host-level resource called resources_available.AppB. This will be set to 2 on any hosts that should have the license, and to 0 on all others. The “nh” flag combination means that it is host-level and it is consumable, so that if a host has 2 licenses, only two jobs can use those licenses on that host at a time.

Server Configuration

1. Define the new resource in the Server’s `resourcedef` file. Create a new file if one does not already exist by adding the resource names, type, and flag(s).

   ```
   cd $PBS_HOME/server_priv/
   [edit] resourcedef
   ```

   Example `resourcedef` file with new resources added:

   ```
   AppB type=long flag=nh
   ```

2. Restart the Server (see section 10.3.4 on page 379).

Host Configuration

3. Set the value of AppB on the hosts to the maximum number of instances allowed. (Ensure that each `qmgr` directive is typed on a single line.)

   qmgr: **active node host1,host2**
   qmgr: **set node resources_available.AppB = 2**

   qmgr: **active node host3,host4,host5,host6**
   qmgr: **set node resources_available.AppB = 0**

Edit the Scheduler configuration file.

```
cd $PBS_HOME/sched_priv/
[edit] sched_config
```

4. Append the new resource name to the "`resources:`" line. Host-level boolean resources do not need to be added to the "resources" line.

   ```
   resources: "ncpus, mem, arch, host, AppB"
   ```

5. Restart or reinitialize the Scheduler (see section 10.3.4 on page 379).

To request a host with a node-locked license for AppB, where you'll run one instance of AppB on two CPUs:

```
qsub -l select=1:ncpus=2:AppB=1
```

The example below shows what the host configuration would look like. What is shown is actually truncated output from the `pbsnodes -a` command. Similar information could be printed via the `qmgr -c "print node @default"` command as well.

```
host1
     resources_available.AppB = 2
host2
     resources_available.AppB = 2
host3
     resources_available.AppB = 0
host4
     resources_available.AppB = 0
host5
     resources_available.AppB = 0
host6
     resources_available.AppB = 0
```

10.7.8 Per-CPU Node-locked Licensing Example

Here is an example of setting up per-CPU node-locked licenses. Each license is for one CPU, so a job that runs this application and needs two CPUs must request two licenses. While that job is using those two licenses, they are unavailable to other jobs.

For this example, we have a 6-host complex, with 4 CPUs per host. The hosts are numbered 1 through 6. On this complex we have a licensed application that uses per-CPU node-locked licenses. We want to limit use of the application only to specific hosts. The table below shows the application, the number of licenses for it, the hosts on which the licenses should be used, and a description of the type of license used by the application.

Application	Licenses	Hosts	DESCRIPTION
AppC	4	3-4	uses a local node-locked application license

For the node-locked license, we will use one static host-level resource called resources_available.AppC. We will provide a license for each CPU on hosts 3 and 4, so this will be set to 4 on any hosts that should have the license, and to 0 on all others. The "nh" flag combination means that it is host-level and it is consumable, so that if a host has 4 licenses, only four CPUs can be used for that application at a time.

Server Configuration

1. Define the new resource in the Server's `resourcedef` file. Create a new file if one does not already exist by adding the resource names, type, and flag(s).

   ```
   cd $PBS_HOME/server_priv/
   [edit] resourcedef
   ```

   Example `resourcedef` file with new resources added:

   ```
   AppC type=long flag=nh
   ```

2. Restart the Server (see section 10.3.4 on page 379).

Host Configuration

3. Set the value of AppC on the hosts. (Ensure that each `qmgr`

directive is typed on a single line.)

qmgr: **active node host3,host4**
qmgr: **set node resources_available.AppC = 4**

qmgr: **active node host1,host2,host5,host6**
qmgr: **set node resources_available.AppC = 0**

Scheduler Configuration

Edit the Scheduler configuration file.

```
cd $PBS_HOME/sched_priv/
[edit] sched_config
```

4. Append the new resource name to the "`resources:`" line. Host-level boolean resources do not need to be added to the "resources" line.

   UNIX:

   `resources: "ncpus, mem, arch, host, AppC"`

   Windows:

   `resources: "ncpus, mem, arch, host, AppC"`

5. Restart or reinitialize the Scheduler (see section 10.3.4 on page 379).

To request a host with a node-locked license for AppC, where you'll run a job using two CPUs:

```
qsub -l select=1:ncpus=2:AppC=2
```

The example below shows what the host configuration would look like. What is shown is actually truncated output from the `pbsnodes -a` command. Similar information could be printed via the `qmgr -c "print node @default"` command as well.

```
host1
```

```
     resources_available.AppC = 0
host2
     resources_available.AppC = 0
host3
     resources_available.AppC = 4
host4
     resources_available.AppC = 4
host5
     resources_available.AppC = 0
host6
     resources_available.AppC = 0
```

10.8 Deleting Custom Resources

If the administrator deletes a resource definition from $PBS_HOME/server_priv/resourcedef and restarts the server, any and all jobs which requested that resource will be purged from the server when it is restarted. Therefore removing any custom resource definition should be done with extreme care.

Chapter 11
Integration & Administration

This chapter covers information on integrations and the maintenance and administration of PBS, and 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.

11.1 pbs.conf

During the installation of PBS Professional, 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_EXEC=/usr/pbs
PBS_HOME=/var/spool/PBS
PBS_START_SERVER=1
PBS_START_MOM=1
PBS_START_SCHED=1
PBS_SERVER=hostname.domain
```

This configuration file controls which components are to be running on the local system, directory tree location, and various runtime configuration options. Each vnode in a complex should have its own `pbs.conf` file. The following table describes the available parameters :

Parameters	Meaning
PBS_BATCH_SERVICE_PORT	Port Server listens on
PBS_BATCH_SERVICE_PORT_DIS	DIS Port Server listens on
PBS_SYSLOG	Controls use of syslog facility
PBS_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_LOCALLOG	Enables logging to local PBS log files
PBS_MANAGER_GLOBUS_SERVICE_PORT	Port Globus MOM listens on
PBS_MANAGER_SERVICE_PORT	Port MOM 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 causes PBS to first try `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

Parameters	Meaning
PBS_START_SERVER	Set to 1 if Server is to run on this vnode
PBS_START_MOM	Set to 1 if a MOM is to run on this vnode
PBS_START_SCHED	Set to 1 if Scheduler is to run on this vnode

11.2 Environment Variables

The settings in $PBS_HOME/pbs_environment are available to user job scripts. You have to HUP the MOM if you change the file. This file is useful for setting environment variables for mpirun etc.

11.3 Ports

PBS daemons listen for inbound connections at specific network ports. These ports have defaults, but can be configured if necessary. For the list of default ports and information on configuring ports, see section 4.9 "Network Addresses and Ports" on page 66. PBS daemons use ports numbered less than 1024 for outbound communication. For PBS daemon-to-daemon communication over TCP, the originating daemon will request a privileged port for its end of the communication.

11.4 Starting and Stopping PBS: UNIX and Linux

The daemons of PBS can be started by two different methods. These methods are not equivalent. The first method is to use the PBS start/stop script, and the second is to run the command that starts the daemon. When you run the PBS start/stop script, PBS will create any vnode definition files. These are not created through the method of running the command that starts a daemon.

The Server, Scheduler, MOM and the optional MOM Globus processes must run with the real and effective uid of root. Typically the components are started automatically by the system upon reboot. The location of the boot-time start/stop script for PBS varies by OS, shown in the following table.

OS	Location of PBS Startup Script
AIX	`/etc/rc.d/rc2.d/S90pbs`
bluegene	`/etc/init.d/pbs`
HP-UX	`/sbin/init.d/pbs`
IRIX	`/etc/init.d/pbs`
Linux	`/etc/init.d/pbs` `/etc/rc.d/init.d/pbs` (on some older linux versions)
NEC	`/etc/init.d/pbs`
OSF1	`/sbin/init.d/pbs`
Solaris	`/etc/init.d/pbs`
Tru64	`/sbin/init.d/pbs`

The PBS startup script reads the `pbs.conf` file to determine which components should be started.

11.4.1 Creation of Configuration Files

When the MOM on a vnode is started via the PBS start/stop script, PBS creates any PBS reserved MOM configuration files. These are not created by the MOM itself, and will not be created when MOM alone is started. Therefore, if you make changes to the number of CPUs or amount of memory that is available to PBS, or if a non-PBS process releases a cpuset, you should restart PBS in order to re-create the PBS reserved MOM configuration files. See section 8.2 “MOM Configuration Files” on page 258.

The startup script can also be run by hand to get status of the PBS components, and to start/stop PBS on a given host. The command-line syntax for the startup script is:

STARTUP_SCRIPT [status | stop | start | restart]

Alternatively, you can start the individual PBS components manually, as discussed in the following sections. Furthermore, you may wish to change the start-up options, as discussed below.

Important: The method by which the Server and MOMs are shut down and restarted has different effects on running jobs; review section 11.4.9 "Impact of Shutdown / Restart on Running Jobs" on page 419.

11.4.2 Starting MOM on Blue Gene

To start or restart the Blue Gene MOM on the service node, run the startup script:

```
/etc/init.d/pbs [start, restart]
```

11.4.3 Starting MOM on the Altix

The cpusetted MOM can be directed to use existing CPU and memory allocations for cpusets. See the option "-p" on page 410.

11.4.4 Manually Starting MOM

If you start MOM before the Server, she will be ready to respond to the Server's "are you there?" ping. However, for a cpusetted Altix cpuset and for Blue Gene, MOM must be started using the PBS startup script.

11.4.4.1 Using qmgr to Set Vnode Resources and Attributes

One of the PBS reserved configuration files is PBSvnodedefs, which is created by a placement set generation script. You can use the output of the placement set generation script to produce input to qmgr. The placement set generation script normally emits data for the PBSvnodedefs file. If the script is given an additional "-v type=q" argument it emits data in a form suitable for input to `qmgr`:

set node <ID> resources_available.<ATTRNAME> = <ATTRVALUE>

where <ID> is a vnode identifier unique within the set of hosts served by a pbs_server. Conventionally, although by no means required, the <ID> above will look like HOST[<localID>] where HOST is the host's FQDN stripped of domain suffixes and <localID> is a identifier whose meaning is unique to the execution host on which the referred to vnode resides. For invariant information, it will look like this:

set node <ID> pnames = RESOURCE[,RESOURCE ...]

11.4.4.2 Manual Creation of cpusets Not Managed by PBS

You may wish to create cpusets not managed by PBS on an Altix running ProPack 4 or greater. If you have not started PBS, create these cpusets before starting PBS. If you have started PBS, requeue any jobs, stop PBS, create your cpuset(s), then restart PBS.

11.4.4.3 Preserving Existing Jobs When Re-starting MOM

If you are starting MOM by hand, you may wish to keep long-running jobs in the running state, and tell MOM to track them. If you use the pbs_mom command with no options, MOM will allow existing jobs to continue to run. Use the -p option to the pbs_mom command to tell MOM to track the jobs.

If you are running PBS on an Altix running ProPack 4 or 5, note that the -p option will tell MOM to use existing cpusets.

Start MOM with the command line:

```
PBS_EXEC/sbin/pbs_mom -p
```

11.4.4.4 Restarting MOM After a Reboot

When a UNIX/Linux operating system is first booted, it begins to assign process IDs (PIDs) to processes as they are created. PID 1 is always assigned to the system "init" process. As new ones are created, they are either assigned the next PID in sequence or the first empty PID found, which depends on the operating system implementation. Generally, the session ID of a session is the PID of the top process in the session.

The PBS MOM keeps track of the session IDs of the jobs. If only MOM is restarted on a system, those session IDs/PIDs have not changed and apply to the correct processes.

If the entire system is rebooted, the assignment of PIDs by the system will start over. Therefore the PID which MOM thinks belongs to an earlier job will now belong to a different later process. If you restart MOM with -p, she will believe the jobs are still valid jobs and the PIDs belong to those jobs. When she kills the processes she believes to belong to one of her earlier jobs, she will now be killing the wrong processes, those created much later but with the same PID as she recorded for that earlier job.

Never restart pbs_mom with the -p or the -r option following a reboot of the host system.

11.4.4.5 Killing Existing Jobs When Re-starting MOM

If you wish to kill any existing processes, use the -r option to pbs_mom.

Start MOM with the command line:

```
PBS_EXEC/sbin/pbs_mom -r
```

11.4.4.6 Options to pbs_mom

These are the options to the pbs_mom command:

-a alarm_timeout
: Number of seconds before alarm timeout. Whenever a resource request is processed, an alarm is set for the given amount of time. If the request has not completed before alarm_timeout, the OS generates an alarm signal and sends it to MOM. Default: 10 seconds. Format: integer.

-C checkpoint_directory
: Specifies the path of the directory used to hold checkpoint files. Only valid on systems supporting checkpoint/restart. The default directory is PBS_HOME/spool/checkpoint. Any 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 the -d option. Format: string.

-c config_file
: MOM will read this alternate default configuration file instead of the normal default configuration file upon starting. If this is a relative file name it will be relative to PBS_HOME/mom_priv. If the specified file cannot be opened, pbs_mom will abort. See the -d option.

  MOM's normal operation, when the -c option is not given, is to 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 home_directory
: Specifies the path of the directory to be used in place of PBS_HOME by pbs_mom. The default directory is $PBS_HOME. Format: string.

Note that pbs_mom uses the default directory to find PBS reserved and site-defined configuration files. Use of the -d option is incompatible with these configuration files, since MOM will not be able to find them if the -d option is given.

-L logfile
: Specifies an absolute path and filename for the log file. The default is a file named for the current date in `PBS_HOME/mom_logs`. See the `-d` option. Format: string.

-M TCP_port
: Specifies the number of the TCP port on which MOM will listen for server requests and instructions. Default: 15002. Format: integer port number

-n nice_val
: Specifies the priority for the pbs_mom daemon. Format: integer

  Note that any spawned processes will have a nice value of zero. If you want all MOM's spawned processes to have the specified nice value, use the UNIX nice command instead: "`nice -19 pbs_mom`".

-p
: Specifies that when starting, MOM should track any running jobs, and allow them to continue running. Cannot be used with the -r option. MOM's default behavior is to allow these jobs to continue to run, but not to track them. MOM is not the parent of these jobs.

  Altix running ProPack 4 or greater
  : The Altix ProPack 4 cpuset pbs_mom will, if given the -p flag, use the existing CPU and memory allocations for cpusets. The default behavior is to remove these cpusets. Should this fail, MOM will exit, asking to be restarted with the -p flag.

-r
: Specifies that when starting, MOM should kill any job processes, mark the jobs as terminated, and notify the server. Cannot be used with the -p option. MOM's default behavior is to allow these jobs to continue to run. MOM is not the parent of these jobs.

Do not use the -r option after a reboot, because process IDs of new, legitimate tasks may match those MOM was previously tracking. If they match and MOM is started with the -r option, MOM will kill the new tasks.

-R UDP_port — Specifies the number of the UDP port on which MOM will listen for pings, resource information requests, communication from other MOMs, etc. Default: 15003. Format: integer port number.

-S server_port — Specifies the number of the TCP port on which pbs_mom initially contact the server. Default: 15001. Format: integer port number.

-s script_options — This option provides an interface that allows the administrator to add, delete, and display MOM's configuration files. See section 8.2 "MOM Configuration Files" on page 258. See the following table for a description of using `script_options`:

Table 22: How -s option i s Used

-s insert <scriptname> <inputfile>	Reads inputfile and inserts its contents in a new site-defined pbs_mom configuration file with the filename scriptname. If a site-defined configuration file with the name scriptname already exists, the operation fails, a diagnostic is presented, and pbs_mom exits with a nonzero status. Scripts whose names begin with the prefix "PBS" are reserved. An attempt to add a script whose name begins with "PBS" will fail. pbs_mom will print a diagnostic message and exit with a nonzero status.

Table 22: How -s option i s Used

-s remove <scriptname>	The configuration file named scriptname is removed if it exists. If the given name does not exist or if an attempt is made to remove a script with the reserved "PBS" prefix, the operation fails, a diagnostic is presented, and pbs_mom exits with a nonzero status.
-s show <script-name>	Causes the contents of the named script to be printed to standard output. If script-name does not exist, the operation fails, a diagnostic is presented, and pbs_mom exits with a nonzero status
-s list	Causes pbs_mom to list the set of PBS reserved and site-defined configuration files in the order in which they will be executed.

-x Disables the check for privileged-port connections.

11.4.5 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 pathname 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 attribute `scheduling`. If the option argument is “true” (“True”, “t”, “T”, or “1”), the server is *active* and the PBS 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` attribute.

-C The server starts up, creates the database, and exits. Windows only.

-d serverhome Specifies the path of the directory which is home to the Server's configuration files, `PBS_HOME`. The default configuration directory is `PBS_HOME` which is defined in `/etc/pbs.conf`.

-e mask Specifies a log event mask to be used when logging. See "log_events" on page 187.

-F seconds Specifies 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 process. Default is 15006.

-g globus_port Specifies the host name and/or port number on which the Server should connect the PBS MOM Globus process. The option argument, *globus_port*, has 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. Default is 15005.

-L logfile Specifies an absolute pathname 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*, has 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`. Default is 15002.

-N The server runs in standalone mode, not as a Windows service. Windows only.

-p port — Specifies the port number on which the Server will listen for batch requests. Default is 15001.

-R RPPport — Specifies the port number on which the Server should query the status of MOM. See the `-R` option for `pbs_mom`. Default is 15003.

-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. Default is 15004.

-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 queues (including jobs in those queues) and re-initialize the Server configuration to the default values. In addition, the Server is idled (scheduling set false). Positive confirmation is required before this direction is accepted.
`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. If a job cannot be restarted immediately because of a missing resource, such as a vnode 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.

Option	Effect Upon Job Running Prior to Server Shutdown
warm	All jobs in the Running state are retained in that state. All other jobs are maintained in their current state. The Scheduler will typically make new selections for which jobs are placed into execution. Warm is the default if -t is not specified.

11.4.6 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 scheduler. Available options are listed below.

-a alarm — Time in seconds to wait for a scheduling cycle to finish. If this 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 1000 seconds.

assign_ssinodes — Deprecated. Do not use.

-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 — The absolute path and filename of the log file. If this option is not given, the scheduler will open a file named for the current date in the PBS_HOME/sched_logs directory. See the -d option.

-n — This will tell the scheduler to not restart itself if it receives a sigsegv or a sigbus. The scheduler will by default restart itself if it receives either of these two signals. The scheduler will not restart itself if it receives either one within five minutes of its start.

-p file — Any output which is written to standard out or standard error will be written to this file. The pathname can be absolute or relative, in which case it will be relative to PBS_HOME/sched_priv. If this option is not given, the file used will be PBS_HOME/sched_priv/sched_out. See the -d option.

-R port — The port for MOM to use. If this option is not given, the port number is taken from PBS_MANAGER_SERVICE_PORT, in pbs.conf. Default: 15003.

-S port — The port for the scheduler to use. If this option is not given, the default port number for the PBS scheduler is taken from PBS_SCHEDULER_SERVICE_PORT, in pbs.conf. Default: 15004.

-N — Instructs the scheduler not to detach itself from the current session.

--version — The pbs_sched command returns its PBS version information and exits. this option can only be used alone.

The options that specify file names may be absolute or relative. If they are relative, their root directory will be `PBS_HOME/sched_priv`.

11.4.7 Manually Starting Globus MOM

The optional Globus MOM should be started at boot time if Globus support is desired. Note that the provided PBS 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 Professional External Reference Specification** (or the `pbs_mom_globus`(1B) manual page) for a more complete explanation.

If the `pbs_mom_globus` process 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). Consequently, `pbs_mom_globus` will request that any Globus jobs that were being tracked and which where running be canceled and requeued.

11.4.8 Stopping PBS

There are two ways to stop PBS. The first is to use the PBS start/stop script, and the second is to use the qterm command.

When you use the pbs start/stop script, by typing "pbs stop",
the server gets a a qterm -t quick (preserving jobs)
MOM gets a SIGTERM - MOM terminates all running children and exits.

The `qterm` command is used to shut down, selectively or inclusively, the various PBS components. It does not perform any of the other cleanup operations that are performed by the PBS shutdown script. The command usage is:

```
qterm [-f | -i | -F] [-m] [-s] [-t type] [server...]
```

The available options, and description of each, follows.

Table 23: qterm Options

(no option)	The `qterm` command defaults to `-t quick` if no options are given.
`-f`	Specifies that the Secondary Server, in a Server failover configuration, should be shut down as well as the Primary Server. If this option is not used in a failover 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 failover 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` components should also be told to shut down. This request is relayed by the Server to each MOM. Jobs are left running subject to other options to `qterm`.
`-s`	Specifies that the Scheduler, `pbs_sched`, should also be terminated.

Table 23: qterm Options

-t <type>	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 rerunnable attribute is true. Otherwise, jobs are killed. Normally the Server will not shut down until there are no jobs in the running state. If the Server is unable to contact the MOM of a 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 shut down 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 single MOM but leave jobs managed by her running, you have two options. The first is to send MOM a SIGINT. This will cause her to shut down in an orderly fashion. The second is to kill MOM with a `SIGKILL` (`-9`). Note that MOM will need to be restarted with the `-p` option in order reattach to the jobs.

11.4.9 Impact of Shutdown / Restart on Running Jobs

The method of how PBS is shut down (and which components are stopped) will affect 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 stop 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 checkpoint and requeue checkpointable jobs, you requeue rerunnable jobs, kill any non-rerunnable jobs, then restart and run jobs that were previously running:

Shutdown: qterm `-t immediate -m -s`

Restart: pbs_mom
pbs_server -t hot
pbs_sched

3. To checkpoint and requeue checkpointable jobs, you requeue rerunnable jobs, kill any non-rerunnable jobs, then restart and run jobs without taking prior state into account:

Shutdown: `qterm -t immediate -m -s`

Restart:
```
pbs_mom
pbs_server -t warm
pbs_sched
```

11.4.10 Stopping / Restarting a Single MOM

If you wish to shut down and restart a single MOM, be aware of the following effects on jobs.

Methods of manual shutdown of a single MOM:

Table 24: Methods for Shutting Down 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` `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:

Table 25: MOM Restart Options

`pbs_mom`	Job processes will continue to run, but the jobs themselves are requeued.

Table 25: MOM Restart Options

`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.

11.5 Starting and Stopping PBS: Windows 2000 / XP

When PBS Professional 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 system boots and shuts down. However, there may come a time when 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 privilege.

```
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 and not as a Windows service, as follows:

```
Admin> pbs_server -N <options>
Admin> pbs_mom -N <options>
Admin> pbs_sched -N <options>
Admin> pbs_rshd -N <options>
```

11.5.1 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->Settings->Control Panel->Administrative Tools->Services` (in Win2000) or `Start Menu->Control Panel->Performance and Maintenance->AdministrativeTools->Services` (in Windows XP).

2. Select the PBS Service you wish 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 alternate 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.

The pbs_server service has two Windows-specific options. These are:

-C The Server starts up, creates the database, and exits.

-N The Server runs in standalone mode, not as a Windows service.

11.6 Checkpoint / Restart Under PBS

PBS Professional 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 Professional to use any method of checkpoint and restart. For details see

section 8.5.2 "Site-Specific Job Checkpoint and Restart" on page 273. (In addition, users may manage their own checkpointing from within their application. This is discussed further in the **PBS Professional User's 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

Note: checkpointing is not supported for job arrays. On systems that support checkpointing, subjobs are not checkpointed; instead they run to completion.

11.6.1 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 Professional Users Guide**).

11.6.2 Checkpointing Jobs During PBS Shutdown

The PBS start/stop script will not result in PBS checkpointing jobs (on systems which provide OS-level checkpointing). This behavior allows for a faster shutdown 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.

11.6.3 Suspending/Checkpointing Multi-vnode Jobs

The PBS suspend/resume and checkpoint/restart capabilities are supported for multi-vnode 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 checkpoint 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(7B)` 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.

11.6.4 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 an 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 reinitialized via `SIGHUP`) only when shutting down PBS just prior to upgrading your system.

11.7 Security

There are three parts to security in the PBS system:

Internal security	Can the component itself be trusted?
Authentication	How do we believe a client about who it is?
Authorization	Is the client entitled to have the requested action performed?

11.7.1 Internal Security

A significant effort has been made to ensure the various PBS components themselves cannot be a target of opportunity in an attack on the system. The two major parts of this effort are the security of files used by PBS and the security of the 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 component resets its environment when it starts. If it does not already exist, the `environment` 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: 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 latter case, the value for the variable is obtained before the environment is reset.

11.7.2 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.

11.7.3 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", the name returned by `gethostname`(2), and the host named by `PBS_SERVER` from `/etc/pbs.conf` are allowed. See the man page for `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.

11.7.4 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`.

11.7.5 User Authorization

PBS as shipped does not assume a consistent user name space within the set of systems which make up a PBS complex. However, the Administrator can enable this assumption, if desired, by setting the server's flatuid attribute to true. This works when running PBS in an environment that *does* have a flat user namespace. To set the `flatuid` Server attribute 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 (UNIX) or the User Database (Windows), nor a `.rhosts` entry on HostY for HostX, nor must HostX appear in HostY's `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*. Note that flatuid may open a security hole in the case where a host has been logged into by someone impersonating a genuine user.

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 (via the `-u user_list` option of the `qsub`(1B) command). If one is not supplied, the name of the job owner is chosen to be the execution name. 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).

Additional information on user authorization is given in section 3.6 "UNIX User Authorization" on page 19 andsection 3.8 "Windows User Authorization" on page 30, as well as in the **PBS Professional User's Guide**.

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 section 7.5 "Server Configuration Attributes" on page 182 and section 7.6.3 "Queue Configuration Attributes" on page 198.)

11.7.6 Group Authorization

PBS allows a user to submit jobs and specify under which group the job should be run at the execution host(s). 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 (and the user's default/primary group will be used).

Under UNIX, 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` is 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".

11.7.7 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, vnodes, and to act on other people's jobs. For specific limitations on "user" privilege, and additional attributes available to Managers and Operators, review the

following: “section 7.1 “The qmgr Command” on page 173; the introduction to “Administrator Commands” on page 491; and the discussion of user commands in the **PBS Professional User’s Guide**.

11.7.8 Enabling Hostbased Authentication on Linux

Hostbased authentication will allow users within your complex to execute commands on or transfer files to remote machines. This can be accomplished for both the r-commands (e.g., rsh, rcp), and secure-commands (e.g., ssh, scp). The following procedure does not enable root to execute any r-commands or secure-commands without a password. Further configuration of the root account would be required.

Correct name resolution is important. Using fully qualified domain names on one machine and short names on another will not work. Name resolution must be consistent across all machines.

11.7.8.1 RSH/RCP

1 Verify that the rsh-server and rsh-client packages are installed on each host within the complex.

2 Verify that the rsh and rlogin services are on on each host within the complex.
Example:

```
chkconfig --list | grep -e rsh -e rlogin
rlogin: on
rsh:    on
```

3 On the headnode (for simplicity) add the hostname of each host within the complex to `/etc/hosts.equiv`, and distribute it to each host within the complex.
Example file (filename: `/etc/hosts.equiv`):

```
headnode
node01
node02
node03
node04
node05
```

11.7.8.2 SSH/SCP

1 Verify that the openSSH package is installed on each host within the complex.

2 Verify that the openSSH service is on on each host within the complex.
Example:

```
chkconfig --list | grep ssh
sshd        0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

3 Modify the following `ssh config` files on each host within the complex to enable the hostbased authentication. These options may be commented out, and so must be uncommented and set.

a. `/etc/ssh/sshd_config`

```
HostbasedAuthentication yes
```

b. `/etc/ssh/ssh_config`

```
HostbasedAuthentication yes
```

4 Stop and start the openSSH service on each host within the complex.

```
/etc/init.d/sshd stop
/etc/init.d/sshd start
```

5 On the headnode (for simplicity) create a file which contains the hostname and IP address of each host within the complex, where the hostname and IP address are comma delimited. Each entry should have all of the information from the line in `/etc/hosts`.
Example file (filename: `ssh_hosts`):

```
headnode,headnode.company.com,192.168.1.100
node01,node01.company.com,192.168.1.1
node02,node02.company.com,192.168.1.2
node03,node03.company.com,192.168.1.3
node04,node04.company.com,192.168.1.4
node05,node05.company.com,192.168.1.5
```

So that if your `/etc/hosts` file has:

```
192.168.1.7 host05.company.com host05
```

the line in `ssh_hosts` would be:

```
node05,node05.company.com,192.168.1.7
```

6 Gather each host's public ssh host key within the complex by executing `ssh-keyscan` against the `ssh_hosts` file created in Step 5, and distribute the output to each host within the complex.

```
ssh-keyscan -t rsa -f ssh_hosts > \
       /etc/ssh/ssh_known_hosts2
```

7 Create the `/etc/ssh/shosts.equiv` file for all of the machines in the complex. This must list the first name given in each line in the `/etc/hosts` file. Using the example from step 5:
Your `/etc/hosts` file has:

```
192.168.1.7 host05.company.com host05
```

The shosts.equiv file should have:

```
node05.company.com
```

8 Every machine in the complex will need to have `ssh_config` and `sshd_config` updated. These files can be copied out to each machine.

SPECIAL NOTES:
The configurations of OpenSSH change (frequently). Therefore, it is important to understand what you need to set up. Here are some tips on some versions.

OpenSSH_3.5p1:
Procedure above should work.

OpenSSH_3.6.1p2:
Procedure above should work with the following additional step:
1. Define “EnableSSHKeysign yes” in the `/etc/ssh/ssh_config` file

OpenSSH_3.9p1:
Procedure above should work with the following two additional steps:
1. Define “EnableSSHKeysign yes” in the `/etc/ssh/ssh_config` file
2. **`chmod 4755 /usr/lib/ssh/ssh-keysign`**
Was 0755 before chmod.
This file is required to be setuid to work.

NOTE for LAM:

Use “`ssh -x`” instead of “`ssh`”.

If you want to use SSH you should enable ‘PermitUserEnvironment yes' so that the user's environment will be passed to the other hosts within the complex. Otherwise, you will see an issue with `tkill` not being in the user's PATH when executing across the hosts.

11.7.9 Security Considerations for Copying Files

If using Secure Copy (scp), then PBS will first try to deliver output or stagein/out files using scp. If scp fails, PBS will try again using rcp (assuming that scp might not exist on the remote host). If rcp also fails, the above cycle will be repeated after a delay, in case the problem is caused by a temporary network problem. All failures are logged in MOM's log, and an email containing the errors is sent to the job owner.

Attempts:

1a scp
1b rcp
2a scp
2b rcp
3a scp
3b rcp
4a scp
4b rcp

11.8 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/Administrator, is listed in the Server attribute `acl_roots`.

The Windows version of PBS considers as a "root" account the following:

Local SYSTEM account
Account that is a member of the local Administrators group on the local host
Account that is a member of the Domain Admins group on the domain
Account that is a member of the Administrators group on the domain controller
Account that is a member of the Enterprise Admins group on the domain
Account that is a member of the Schema Admins group on the domain

In order to submit a job from this "root" account on the local host, be sure to set acl_roots. For instance, if user foo is a member of the Administrators group, then you need to set:

```
qmgr:  set server acl_roots += foo
```

in order to submit jobs and not get a “bad uid for job execution” message.

Important: Allowing “root” jobs means that they can run on a configured host under the same account which could also be a privileged account on that host.

11.9 Managing PBS and Multi-vnode Parallel Jobs

Many customers use PBS Professional in cluster configurations for the purpose of managing multi-vnode parallel applications. This section provides the PBS Administrator with information specific to this situation.

11.9.1 The PBS_NODEFILE

For each job, PBS will create a job-specific “host file” or “node file”—a text file containing the name of the vnode(s) allocated to that job, listed one per line. The file will be created by the MOM on the first vnode in `PBS_HOME/aux/JOB_ID`, where *`JOB_ID`* is the actual job identifier for that job. The full path and name for this file is written to the job’s environment via the variable `PBS_NODEFILE`. (See also details on using this environment variable in Chapter 10 of the **PBS Professional User’s Guide**.)

The order in which hosts appear in the PBS_NODEFILE is the order in which chunks are specified in the selection directive. The order in which hostnames appear in the file is hostA X times, hostB Y times, where X is the number of MPI processes on hostA, Y is the number of MPI processes on hostB, etc. See the definition of the resources “mpiprocs” and “ompthreads” in “Resource Types” on page 223.

The number of MPI processes for a job is controlled by the value of the resource mpiprocs. The mpiprocs resource controls the contents of the PBS_NODEFILE on the host which executes the top PBS task for the PBS job (the one executing the PBS job script.) See “Built-in Resources” on page 29. The PBS_NODEFILE contains one line per MPI process with the name of the host on which that process should execute. The number of lines in PBS_NODEFILE is equal to the sum of the values of mpiprocs over all chunks requested by the job. For each chunk with mpiprocs=P, (where P > 0), the host name (the value of the allocated vnode's resources_available.host) is written to the PBS_NODEFILE exactly P times.

The number of OpenMP threads for a job is controlled by the value of the resource ompthreads. The ompthreads resource controls the values of the NCPUS and OMP_NUM_THREADS environment variables for every PBS task (including the top

PBS task).

If a chunk requests ncpus=N, with N > 1, PBS will only create one MPI process for that chunk, but set the number of OpenMP threads to N.

11.10 Support for MPI

PBS Professional is tightly integrated with several implementations of MPI. PBS can track resource usage for all of the tasks run under these MPIs. Some of the MPI integrations use pbs_attach, which means MOM polls for usage information like CPU time. The amount of usage data lost between polling cycles will depend on the length of the polling cycle. See "Configuring MOM's Polling Cycle" on page 270.

11.10.1 Interfacing MPICH with PBS Professional on UNIX

The existing mpirun command can be modified to check for the PBS environment and use the PBS-supplied host file. Do this 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-vnode Jobs" on page 217.

11.10.1.1 MPICH on Linux

On Linux systems running MPICH with P4, the existing `mpirun` command is replaced with `pbs_mpirun` The `pbs_mpirun` command is a shell script which attaches a user's MPI tasks to the PBS job.

11.10.1.2 The `pbs_mpirun` Command

The PBS command `pbs_mpirun` replaces the standard `mpirun` command in a PBS MPICH job using P4. The usage is the same as `mpirun` except for the `-machinefile` option. The value for this option is generated by `pbs_mpirun`. All other options are

passed directly to mpirun. The value used for the -machinefile option is a temporary file created from the PBS_NODEFILE in the format expected by mpirun. If the -machinefile option is specified on the command line, a warning will be output saying "Warning, -machinefile value replaced by PBS". The default value for the -np option is the number of entries in PBS_NODEFILE.

11.10.1.3 Transparency to the User

Users should be able to continue to run existing scripts. To be transparent to the user, pbs_mpirun should replace standard mpirun. To do this, the link for mpirun should be changed to point to pbs_mpirun:

* install MPICH into /usr/local/mpich (or note path for mpirun)
* mv /usr/local/mpich/bin/mpirun /usr/local/mpich/bin/mpirun.std
* create link called “mpirun” pointing to pbs_mpirun in /usr/local/mpich/bin/
* edit pbs_mpirun to change "mpirun" call to "mpirun.std"

At this point, using "mpirun" will actually invoke pbs_mpirun.

When pbs_mpirun is run, it runs pbs_attach, which attaches the user’s MPI process to the job.

11.10.1.4 Environment Variables and PATHs

The PBS_RSHCOMMAND environment variable should not be set by the user. For pbs_mpirun to function correctly for users who require the use of ssh instead of rsh, several approaches are possible:

1Set P4_RSHCOMMAND in the login environment.

2Set P4_RSHCOMMAND externally to the login environment, then pass the value to PBS via qsub(1)'s -v or -V arguments:

```
qsub -vP4_RSHCOMMAND=ssh ...
```

or

```
qsub -V ...
```

3A PBS administrator may set P4_RSHCOMMAND in the pbs_environment file in PBS_HOME and advise users to not set P4_RSHCOMMAND in the login environment

PATH on remote machines must contain PBS_EXEC/bin. Remote machines must all have pbs_attach in the PATH.

11.10.1.5 Notes

When using SuSE Linux, use "`ssh -n`" in place of "`ssh`".

Usernames must be identical across vnodes.

11.10.2 Integration with LAM MPI

11.10.2.1 The pbs_lamboot Command

The PBS command `pbs_lamboot` replaces the standard `lamboot` command in a PBS LAM MPI job, for starting LAM software on each of the PBS execution hosts.

Usage is the same as for LAM's lamboot. All arguments except for `bhost` are passed directly to `lamboot`. PBS will issue a warning saying that the `bhost` argument is ignored by PBS since input is taken automatically from $PBS_NODEFILE. The `pbs_lamboot` program will not redundantly consult the $PBS_NODEFILE if it has been instructed to boot the hosts using the `tm` module. This instruction happens when an argument is passed to `pbs_lamboot` containing "-ssi boot tm" or when the `LAM_MPI_SSI_boot` environment variable exists with the value `tm`.

11.10.2.2 The pbs_mpilam Command

The PBS command pbs_mpilam replaces the standard mpirun command in a PBS LAM MPI job, for executing programs. It attaches the user's processes to the PBS job. This allows PBS to collect accounting information, and to manage the processes.

Usage is the same as for LAM mpirun. All options are passed directly to mpirun. If the where argument is not specified, pbs_mpilam will try to run the user's program on all available CPUs using the C keyword.

11.10.2.3 PATH

The PATH for `pbs_lamboot` and `pbs_mpilam` on all remote machines must contain PBS_EXEC/bin.

11.10.2.4 Transparency to the User

Both pbs_lamboot and pbs_mpilam should be transparent to the user. Users should be able to run existing scripts.

To be transparent to the user, pbs_lamboot should replace LAM lamboot. The link for lamboot should be changed to point to pbs_lamboot.

•Install LAM MPI into `/usr/local/lam-<version>`
·mv /usr/local/lam-<version>/bin/lamboot
/user/local/lam-<version>/bin/lamboot.lam
•Edit `pbs_lamboot` to change “lamboot” call to “lamboot.lam”
•Rename `pbs_lamboot` to `lamboot`:

```
cd /usr/local/lam-<version>/bin
ln  -s PBS_EXEC/bin/pbs_lamboot lamboot
```

At this point, using “lamboot” will actually invoke `pbs_lamboot`.

To be transparent to the user, `pbs_mpilam` should replace `LAM mpirun`. The link for `mpirun` should be changed to point to `pbs_mpilam`.

•Install LAM MPI into `/usr/local/lam-<version>`
·mv /usr/local/lam-<version>/bin/mpirun
/user/local/lam-<version>/bin/mpirun.lam
•Edit `pbs_mpirun` to change “mpirun” call to “mpirun.lam”
•Rename `pbs_mpilam` to `mpirun`:

```
cd /usr/local/lam-<version>/bin
ln -s PBS_EXEC/bin/pbs_mpilam mpirun
```

Either LAMRSH or LAM_SSI_rsh_agent will need to have the value "ssh -x", depending on whether you are using rsh or ssh.

11.10.3 Integration with HP MPI on HP-UX and Linux

11.10.3.1 The pbs_mpihp Command

The PBS command pbs_mpihp replaces the standard mpirun and mpiexec commands in a PBS HP MPI job on HP-UX and Linux, for executing programs. It attaches the user's processes to the PBS job. This allows PBS to collect accounting information, and to manage the processes.

11.10.3.2 Transparency to the User

To be transparent to the user, `pbs_mpihp` should replace HP `mpirun`. The recommended steps for making `pbs_mpihp` transparent to the user are:

Rename HP's `mpirun`:

```
cd <MPI installation location>/bin
mv mpirun mpirun.hp
```

Link the user-callable "mpirun" to `pbs_mpihp`:

```
cd <MPI installation location>/bin
ln -s $PBS_EXEC/bin/pbs_mpihp mpirun
```

Create a link to `mpirun.hp` from `PBS_EXEC/etc/pbs_mpihp`. `pbs_mpihp` will call the real HP `mpirun`:

```
cd $PBS_EXEC/etc
ln -s <MPI installation location>/bin/mpirun.hp
 pbs_mpihp
```

When wrapping HP MPI with `pbs_mpihp`, note that rsh is the default used to start the mpids. If you wish to use ssh or something else, be sure to set the following or its equivalent in $PBS_HOME/pbs_environment:

```
PBS_RSHCOMMAND=ssh
```

11.10.4 SGI MPI on the Altix Running ProPack 4 or 5

PBS supplies its own `mpiexec` on the Altix. This `mpiexec` uses the standard SGI `mpirun`. No unusual setup is required for either `mpiexec` or `mpirun`, however, there are prerequisites. See the following section. If executed on a non-Altix system, PBS's `mpiexec` will assume it was invoked by mistake. In this case it will use the value of

PATH (outside of PBS) or PBS_O_PATH (inside PBS) to search for the correct mpiexec and if one is found, exec it. The name of the array to use when invoking mpirun is user-specifiable via the PBS_MPI_SGIARRAY environment variable.

The PBS mpiexec is transparent to the user; MPI jobs submitted outside of PBS will run as they would normally. MPI jobs can be launched across multiple Altixes. PBS will manage, track, and cleanly terminate multi-host MPI jobs. PBS users can run MPI jobs within specific partitions.

If CSA has been configured and enabled, PBS will collect accounting information on all tasks launched by an MPI job. CSA information will be associated with the PBS job ID that invoked it, on each execution host. While each host involved in an MPI job will record CSA accounting information for the job if able to do so on the execution hosts, there is no tool to consolidate the accounting information from multiple hosts.

If the PBS_MPI_DEBUG environment variable's value has a nonzero length, PBS will write debugging information to standard output.

PBS uses the MPI-2 industry standard mpiexec interface to launch MPI jobs within PBS.

11.10.4.1 Prerequisites

In order to run single-host or multi-host jobs, the SGI Array Services must be correctly configured. An Array Services daemon (arrayd) must run on each host that will run MPI processes. For a single-host environment, arrayd only needs to be installed and activated. However, for a multi-host environment where applications will run across hosts, the hosts must be properly configured to be an array.

Altix systems communicating via SGI's Array Services must all use the same version of the sgi-mpt and sgi-arraysvcs packages. Altix systems communicating via SGI's Array Services must have been configured to interoperate with each other using the default array. See SGI's array_services(5) man page.

"**rpm -qi sgi-arraysvcs**" should report the same value for Version on all systems.

"**rpm -qi sgi-mpt**" should report the same value for Version on all systems.

"**chkconfig array**" must return "on" for all systems

/usr/lib/array/arrayd.conf must contain an array definition

that includes all systems.

`/usr/lib/array/arrayd.auth` must be configured to allow remote access:

The "`AUTHENTICATION NOREMOTE`" directive must be commented out or removed

Either "`AUTHENTICATION NONE`" should be enabled or keys should be added to enable the `SIMPLE` authentication method.

If any changes have been made to the arrayd configuration files (`arrayd.auth` or `arrayd.conf`), the array service must be restarted.

`rsh`(1) must work between the systems.

PBS uses SGI's `mpirun`(1) command to launch MPI jobs. SGI's `mpirun` must be in the standard location.

The location of `pbs_attach`(8B) on each vnode of a multi-vnode MPI job must be the same as it is on the mother superior vnode.

11.10.4.2 Environment Variables

The PBS `mpiexec` script sets the `PBS_CPUSET_DEDICATED` environment variable to assert exclusive use of the resources in the assigned cpuset.

The PBS `mpiexec` checks the `PBS_MPI_DEBUG` environment variable. If this variable has a nonzero length, debugging information is written.

If the `PBS_MPI_SGIARRAY` environment variable is present, the PBS `mpiexec` will use its value as the name of the array to use when invoking mpirun.

The `PBS_ENVIRONMENT` environment variable is used to determine whether `mpiexec` is being called from within a PBS job.

The PBS `mpiexec` uses the value of `PBS_O_PATH` to search for the correct `mpiexec` if it was invoked by mistake.

11.10.5 SGI's MPI (MPT) Over InfiniBand

PBS jobs can run using SGI's MPI, called MPT, over InfiniBand. To use InfiniBand, set the MPI_USE_IB environment variable to 1.

11.10.6 The pbsrun_wrap Mechanism

PBS provides a mechanism for wrapping several versions/flavors of mpirun so that PBS can control jobs and perform accounting. PBS also provides a mechanism for unwrapping these versions of mpirun. The administrator wraps a version of mpirun using the `pbsrun_wrap` script, and unwraps it using the `pbsrun_unwrap` script. The `pbsrun_wrap` script is the installer script that wraps mpirun in a script called "pbsrun". The `pbsrun_wrap` script instantiates the pbsrun script for each version of mpirun, renaming it to reflect the version/flavor of mpirun being wrapped. When executed inside a PBS job, the pbsrun script calls a version-specific initialization script which sets variables to control how the pbsrun script uses options passed to it. The pbsrun script uses `pbs_attach` to give MOM control of jobs.

The `pbsrun_wrap` command has a "-s" option. If `-s` is specified, then the "strict_pbs" options set in the various initialization scripts (e.g. pbsrun.bgl.init, pbsrun.ch_gm.init, etc...) will be set to 1 from the default 0. This means that the mpirun being wrapped by pbsrun will only get executed if inside a PBS environment. Otherwise, the user will get the error:

```
Not running under PBS
exiting since strict_pbs is enabled; execute only in PBS
```

The `pbsrun_wrap` command has this format:

```
pbsrun_wrap [-s] <path_to_actual_mpirun> pbsrun.<keyword>
```

If the mpirun wrapper script is run inside a PBS job, then it will translate any mpirun call of the form:

mpirun [options] <executable> [args]

into

mpirun [options] pbs_attach [special_option_to_pbs_attach] <executable> [args]

where [special options] refers to any option needed by pbs_attach to do its job (e.g. -j $PBS_JOBID).

If the wrapper script is executed outside of PBS, a warning is issued about "not running under PBS", but it proceeds as if the actual program had been called in standalone fashion.

Any mpirun version/flavor that can be wrapped has an initialization script ending in ".init", found in $PBS_EXEC/lib/MPI:

$PBS_EXEC/lib/MPI/pbsrun.<mpirun version/flavor>.init.

The pbsrun_wrap script instantiates the pbsrun wrapper script as pbsrun.<mpirun version/flavor> in the same directory where pbsrun is located, and sets up the link to the actual mpirun call via the symbolic link

$PBS_EXEC/lib/MPI/pbsrun.<mpirun version/flavor>.link

For example, running:

pbsrun_wrap /opt/mpich-gm/bin/mpirun.ch_gm pbsrun.ch_gm

causes the following actions:

Save original mpirun.ch_gm script:

```
mv /opt/mpich-gm/bin/mpirun.ch_gm \
/opt/mpich/gm/bin/mpirun.ch_gm.actual
```

Instantiate pbsrun wrapper script as pbsrun.ch_gm:

```
cp $PBS_EXEC/bin/pbsrun $PBS_EXEC/bin/pbsrun.ch_gm
```

Link "mpirun.ch_gm" to actually call "pbsrun.ch_gm":

```
ln -s $PBS_EXEC/bin/pbsrun.ch_gm \
/opt/mpich-gm/bin/mpirun.ch_gm
```

Create a link so that "pbsrun.ch_gm" calls "mpirun.ch_gm.actual":

```
ln -s /opt/mpich-gm/bin/mpirun.ch_gm.actual \
$PBS_EXEC/lib/MPI/pbsrun.ch_gm.link
```

The mpirun being wrapped must be installed and working on all the vnodes in the PBS cluster.

For all wrapped MPIs, the maximum number of ranks that can be launched in a job is the number of entries in the $PBS_NODEFILE.

11.10.6.1 The pbsrun Script

The pbsrun wrapper script is not meant to be executed directly but instead it is instantiated by `pbsrun_wrap`. It is copied to the target directory and renamed "pbsrun.<mpirun version/flavor>" where <mpirun version/flavor> is a string that identifies the mpirun version being wrapped (e.g. ch_gm).

The pbsrun script, if executed inside a PBS job, runs an initialization script, named $PBS_EXEC/lib/MPI/pbsrun.<mpirun version/flavor>.init, then parses mpirun-like arguments from the command line, sorting which options and option values to retain, to ignore, or to transform, before calling the actual mpirun script with a "pbs_attach" prefixed to the executable. The actual mpirun to call is found by tracing the link pointed to by $PBS_EXEC/lib/MPI/pbsrun.<mpirun version/flavor>.link.

11.10.6.2 The pbsrun Initialization Script

The initialization script, called $PBS_EXEC/lib/MPI/pbsrun.<mpirun version/flavor>.init, where <mpirun version/flavor> reflects the mpirun flavor/version being wrapped, can be modified by an administrator to customize against the local flavor/version of mpirun being wrapped.
Inside this sourced init script, 8 variables are set:

```
options_to_retain="-optA -optB <val> -optC <val1> val2> ..."
options_to_ignore="-optD -optE <n> -optF <val1> val2> ..."
options_to_transform="-optG -optH <val> -optI <val1> val2> ..."
options_to_fail="-optY -optZ ..."
options_to_configfile="-optX <val> ..."
options_with_another_form="-optW <val> ..."
pbs_attach=pbs_attach
options_to_pbs_attach="-J $PBS_JOBID"
```

options_to_retain	Space-separated list of options and values that pbsrun.<mpirun version/flavor> passes on to the actual mpirun call. options must begin with "-" or "--", and option arguments must be specified by some arbitrary name with left and right arrows, as in "<val1>".
options_to_ignore	Space-separated list of options and values that pbsrun.<mpirun version/flavor> does not pass on to the actual mpirun call. Options must begin with "-" or "--", and option arguments must be specified by arbitrary names with left and right arrows, as in "<n>".

options_to_transform	Space-separated list of options and values that pbsrun modifies before passing on to the actual mpirun call.
options_to_fail	Space-separated list of options that will cause pbsrun to exit upon encountering a match.
options_to_configfile	Single option and value that refers to the name of the "config-file" containing command line segments found in certain versions of mpirun.
options_with_another_form	Space-separated list of options and values that can be found in options_to_retain, options_to_ignore, or options_to_transform, whose syntax has an alternate, unsupported form.
pbs_attach	Path to pbs_attach, which is called before the <executable> argument of mpirun.
options_to_pbs_attach	Special options to pass to the pbs_attach call. You may pass variable references (e.g. $PBS_JOBID) and they are substituted by pbsrun to actual values.

If pbsrun encounters any option not found in options_to_retain, options_to_ignore, and options_to_transform, then it is flagged as an error.

These functions are created inside the init script. These can be modified by the PBS administrator.

```
transform_action () {
        # passed actual values of $options_to_transform
        args=$*
}

boot_action () {
        mpirun_location=$1
}

evaluate_options_action () {
        # passed actual values of transformed options
        args=$*
}
```

```
configfile_cmdline_action () {
        args=$*
}

end_action () {
        mpirun_location=$1
}
```

transform_action() The pbsrun.<mpirun version/flavor> wrapper script invokes the function transform_action() (called once on each matched item and value) with actual options and values received matching one of the "options_to_transform". The function returns a string to pass on to the actual mpirun call.

boot_action() Performs any initialization tasks needed before running the actual mpirun call. For instance, GM's MPD requires the MPD daemons to be user-started first. This function is called by the pbsrun.<mpirun version/flavor> script with the location of actual mpirun passed as the first argument. Also, the pbsrun.<mpirun version/flavor> checks for the exit value of this function to determine whether or not to progress to the next step.

evaluate_options_action()
Called with the actual options and values that resulted after consulting options_to_retain, options_to_ignore, options_to_transform, and executing transform_action(). This provides one more chance for the script writer to evaluate all the options and values in general, and make any necessary adjustments, before passing them on to the actual mpirun call. For instance, this function can specify what the default value is for a missing -np option.

configfile_cmdline_action()
Returns the actual options and values to be put in before the options_to_configfile parameter.

configfile_firstline_action()
Returns the item that is put in the first line of the configuration file specified in the options_to_configfile parameter.

end_action()	Called by pbsrun.<mpirun version/flavor> at the end of execution. It undoes any action done by transform_action(), like cleanup of temporary files. It is also called when pbsrun.<mpirun version/flavor> is prematurely killed. This function is called with the location of actual mpirun passed as first argument.

The actual mpirun program to call is the path pointed to by $PBS_EXEC/lib/MPI/pbsrun.<mpirun version/flavor>.link.

11.10.6.3 Modifying *.init Scripts

In order for administrators to modify *.init scripts without breaking package verification in RPM, master copies of the initialization scripts are named *.init.in. `pbsrun_wrap` instantiates the *.init.in files as *.init. For instance, `$PBS_EXEC/lib/MPI/pbsrun.mpich2.init.in` is the master copy, and `pbsrun_wrap` instantiates it as `$PBS_EXEC/lib/MPI/pbsrun.mpich2.init`. `pbsrun_unwrap` takes care of removing the *.init files.

11.10.6.4 Wrapping Multiple MPI's with the Same Name

You may want more than one MPI environment with the same name, for example a 32-bit and a 64-bit version of MPICH2.

Create two new MPICH2 initialization scripts by copying that for MPICH2:

```
# cd $PBS_EXEC/lib/MPI
# cp pbsrun.mpich2.init.in pbsrun.mpich2_32.init.in
# cp pbsrun.mpich2.init.in pbsrun.mpich2_64.init.in
```

Then wrap them:

```
# pbsrun_wrap <path to 32-bit MPICH2>/bin/mpirun \
    pbsrun.mpich2_32
# pbsrun_wrap <path to 64-bit MPICH2>/bin/mpirun \
    pbsrun.mpich2_64
```

Calls to "<path to 32-bit MPICH2>/bin/mpirun" will invoke /usr/pbs/bin/pbsrun.mpich2_32. The 64-bit version is invoked with calls to

"<path to 64-bit MPICH2>/bin/mpirun".

When you are done using them, unwrap them:

```
# pbsrun_unwrap pbsrun.mpich2_32
# pbsrun_unwrap pbsrun.mpich2_64
```

11.10.7 Wrapping MPICH-GM's mpirun.ch_gm with rsh/ssh

The PBS wrapper script to MPICH-GM's mpirun (mpirun.ch_gm) with rsh/ssh process startup method is named pbsrun.ch_gm. If executed inside a PBS job, this allows for PBS to track all MPICH-GM processes started by rsh/ssh so that PBS can perform accounting and have complete job control. If executed outside of a PBS job, it behaves exactly as if standard mpirun.ch_gm was used.

To wrap MPICH-GM's mpirun script:

```
pbsrun_wrap [MPICH-GM_BIN_PATH]/mpirun.ch_gm \
      pbsrun.ch_gm
```

To unwrap MPICH-GM's mpirun script:

```
pbsrun_unwrap pbsrun.ch_gm
```

11.10.8 Wrapping MPICH-MX's mpirun.ch_gm with rsh/ssh

The PBS wrapper script to MPICH-MX's mpirun (mpirun.ch_gm) with rsh/ssh process startup method is named pbsrun.ch_mx. If executed inside a PBS job, this allows for PBS to track all MPICH-MX processes started by rsh/ssh so that PBS can perform accounting and has complete job control. If executed outside of a PBS job, it behaves exactly as if standard mpirun.ch_mx was used.

To wrap MPICH-MX's mpirun script:

```
pbsrun_wrap [MPICH-MX_BIN_PATH]/mpirun.ch_mx
          pbsrun.ch_mx
```

To unwrap MPICH-MX's mpirun script:

```
pbsrun_unwrap pbsrun.ch_mx
```

11.10.9 Wrapping MPICH-GM's mpirun.ch_gm with MPD

The PBS wrapper script to MPICH-GM's mpirun (mpirun.ch_gm) with MPD process startup method is called pbsrun.gm_mpd. If executed inside a PBS job, this allows for PBS to track all MPICH-GM processes started by the MPD daemons so that PBS can perform accounting have and complete job control. If executed outside of a PBS job, it behaves exactly as if standard mpirun.ch_gm with MPD was used.

To wrap MPICH-GM's mpirun script with MPD:

```
pbsrun_wrap [MPICH-GM_BIN_PATH]/mpirun.mpd
          pbsrun.gm_mpd
```

To unwrap MPICH-GM's mpirun script with MPD:

```
pbsrun_unwrap pbsrun.gm_mpd
```

11.10.10 MPICH-MX's mpirun.ch_mx with MPD

The PBS wrapper script to MPICH-MX's mpirun (mpirun.ch_mx) with MPD process startup method is called pbsrun.mx_mpd. If executed inside a PBS job, this allows for PBS to track all MPICH-MX processes started by the MPD daemons so that PBS can perform accounting and have complete job control. If executed outside of a PBS job, it behaves exactly as if standard mpirun.ch_mx with MPD was used.

The script starts MPD daemons on each of the unique hosts listed in $PBS_NODEFILE, using either rsh or ssh method, based on value of environment variable RSHCOMMAND -- rsh is the default. The script also takes care of shutting down the MPD daemons at the end of a run.

To wrap MPICH-MX's mpirun script with MPD:

```
pbsrun_wrap [MPICH-MX_BIN_PATH]/mpirun.mpd
          pbsrun.mx_mpd
```

To unwrap MPICH-MX's mpirun script with MPD:

```
pbsrun_unwrap pbsrun.mx_mpd
```

11.10.11 Wrapping MPICH2's mpirun

The PBS wrapper script to MPICH2's mpirun is called pbsrun.mpich2. If executed inside a PBS job, this allows for PBS to track all MPICH2 processes so that PBS can perform accounting and have complete job control. If executed outside of a PBS job, it behaves exactly as if standard MPICH2's mpirun was used.

The script takes care of ensuring that the MPD daemons on each of the host listed in the $PBS_NODEFILE are started. It also takes care of ensuring that the MPD daemons have been shut down at the end of MPI job execution.

To wrap MPICH2's mpirun script:

```
pbsrun_wrap [MPICH2_BIN_PATH]/mpirun pbsrun.mpich2
```

To unwrap MPICH2's mpirun script:

```
pbsrun_unwrap pbsrun.mpich2
```

11.10.12 Wrapping Intel MPI's mpirun

The PBS wrapper script to Intel MPI's mpirun is called pbsrun.intelmpi. If executed inside a PBS job, this allows for PBS to track all Intel MPI processes so that PBS can perform accounting and have complete job control. If executed outside of a PBS job, it behaves exactly as if standard Intel MPI's mpirun was used.

Intel MPI's mpirun itself takes care of starting/stopping the MPD daemons. pbsrun.intelmpi always passes the arguments -totalnum=<number of mpds to start> and -file=<mpd_hosts_file> to the actual mpirun, taking its input from unique entries in $PBS_NODEFILE.

To wrap Intel MPI's mpirun script:

```
pbsrun_wrap [INTEL_MPI_BIN_PATH]/mpirun pbsrun.intelmpi
```

To unwrap Intel MPI's mpirun script:

```
pbsrun_unwrap pbsrun.intelmpi
```

11.10.13 Wrapping MVAPICH1's mpirun

MVAPICH1 allows the use of InfiniBand. The PBS wrapper script to MVAPICH1's mpirun is called pbsrun.mvapich1. If executed inside a PBS job, this allows for PBS to track all MPI processes so that PBS can perform accounting and have complete job control. If executed outside of a PBS job, it behaves exactly as if standard MVAPICH1's mpirun was used.

If executed inside a PBS job script, all mpirun options given are passed on to the actual mpirun call with these exceptions:

-map <list>	The map option is ignored.
-exclude <list>	The exclude option is ignored.
-machinefile <file>	The machinefile option is ignored.
-np	If not specified, the number of entries found in the $PBS_NODEFILE is used.

To wrap the MVAPICH1 mpirun:

```
pbsrun_wrap [MVAPICH1_BIN_PATH]/mpirun pbsrun.mvapich1
```

To MVAPICH1 mpirun:

```
pbsrun_unwrap pbsrun.mvapich1
```

11.10.14 Wrapping MVAPICH2's mpiexec

MVAPICH2 allows the use of InfiniBand. The PBS wrapper script to MVAPICH2's mpiexec is called pbsrun.mvapich2. If executed inside a PBS job, this allows for PBS to track all MPI processes so that PBS can perform accounting and have complete job control. If executed outside of a PBS job, it behaves exactly as if standard MVAPICH2's mpiexec had been used.

pbsrun.mvapich2 takes care of starting and stopping the MPD daemons if the user doesn't explicitly start and stop them.

If executed inside a PBS job script, all mpiexec options given are passed on to the actual mpiexec call with these exceptions:

-host <host>	The host argument contents are ignored.
-machinefile <file>	The file argument contents are ignored and replaced by the contents of the $PBS_NODEFILE.

To wrap the MVAPICH2 mpiexec:

```
pbsrun_wrap [MVAPICH2_BIN_PATH]/mpiexec pbsrun.mvapich2
```

To unwrap MVAPIC21 mpiexec:

```
pbsrun_unwrap pbsrun.mvapich2
```

11.10.15 Wrapping IBM's poe

MPI is supported under IBM's Parallel Operating Environment (POE) on AIX. Under AIX, the program poe is used to start user processes on remote machines. PBS will manage the IBM HPS in US (User Space) mode.

The PBS wrapper script to IBM's poe is called pbsrun.poe. If executed inside a PBS job, this allows for PBS to track all poe processes so that PBS can perform accounting and have complete job control. If executed outside of a PBS job, it behaves exactly as if standard IBM poe had been used.

If executed inside a PBS job script, all pbsrun.poe options given are passed on to standard poe with these exceptions:

-hostfile <file>	The file argument contents are ignored.
-procs <numranks>	If the -procs option or the MP_PROCS environment variable is not set by the user, a default of the number of entries in the file $PBS_NODEFILE is used.
-euilib {ip | us}	If the command line option -euilib is set, it will take precedence over the MP_EUILIB environment variable. If the -euilib option is set to us , user mode is set for the job. If the option is set to any other value, that value is passed to standard poe.

-msg_api This option can only take the values "MPI" or "LAPI".

Environment Variables

MP_EUILIB If the MP_EUILIB environment variable is set to us , user mode is set for the job. If the variable is set to any other value, that value is passed to standard poe.

MP_HOSTFILE The MP_HOSTFILE environment variable is excised.

MP_PROCS If the -procs option or the MP_PROCS environment variable is not set by the user, a default of the number of entries in the file $PBS_NODEFILE is used.

MP_MSG_API This variable can only take the values "MPI" or "LAPI".

To wrap IBM poe:

```
pbsrun_wrap [POE_BIN_PATH]/poe pbsrun.poe
```

To unwrap IBM poe:

```
pbsrun_unwrap pbsrun.poe
```

You can use set the number of HPS US mode jobs MOM will accept:

Example: set node aix_15 to only accept one HPS US mode job at any one time:

```
# qmgr -c 'set node aix_15 resources_available.hps = 1'
```

Example: set node aix_75 to accept multiple HPS US mode jobs at any one time:

```
# qmgr -c 'set node aix_75 resources_available.hps =
99999'
```

You will need to set up a custom resource for the HPS so that hps is a static consumable host-level resource. See section 10.3.2 "Defining and Using a Custom Resource" on page 375. Users will need to request the "hps" resource in their select statements.

If you have some machines in the complex that are not on the HPS, be sure that those machines have their hps resource set to zero.

```
# qmgr -c 'set node not_ibm resources_available.hps \
   = 0'
```

As an alternative, you can use "sharing=force_excl" to limit the number of HPS US mode jobs to 1, but it would be more restrictive. In this case, one and only one job could run on the HPS.

An example of the way to do this (in this case, changing the "sharing" attribute for a vnode named aix_15) uses the script “change_sharing”. See section 8.2.1 “Creation of Site-defined MOM Configuration Files” on page 259.

```
# cat change_sharing
$configversion 2
aix_15:  sharing = force_excl
# . /etc/pbs.conf
# $PBS_EXEC/sbin/pbs_mom -s insert force_excl
      change_sharing
# pkill -HUP pbs_mom
```

11.11 Support for IBM Blue Gene

11.11.1 PBS on Blue Gene

A Blue Gene job contains an executable, its arguments, and owner (one who submitted the job). It runs exclusively on a 3d, rectangular, contiguous, isolated set of compute nodes called a partition or bglblock. Valid partition sizes are as follows:

64 CPUs (1/16 base partition, or BP)
256 CPUs (1/4 BP)
1024 CPUs (1 BP)
one or more BPs

See the PBS Professional User’s Guide for more information about partitions and how jobs run.

Partitions can initially be defined and overlapping. When the time comes for a job to use a partition, it must be initialized/booted. This will only succeed if any sub-partitions that are overlapping with the given partition are free and usable. Booting a partition takes about 20 seconds for a small partition, or 10 minutes for a large one of 64 base partitions. A partition can be reused by another job having the same requirement to avoid the overhead of rebooting.

There are two ways of partitioning a system. One is called static partitioning where a system administrator pre-defines a set of partitions in advance to satisfy users' requirements. Then users simply specify the partition name to run under in their mpirun request. Another way is called dynamic partitioning where some entity like a scheduler creates partitions on the fly according to users' workload. PBS supports static partitions.

Partitions go through various states. When a partition is pre-created, it will have a state of FREE. If it has been initialized/allocated/booted, then it goes into a state of READY. If a job is running on the partition, an internal partition attribute will have this information.

Users invoke `mpirun` in their job scripts to run their executables. Users can specify the compute node execution mode and the number of tasks. Compute nodes can be under-allocated, but not over-allocated. See the PBS Professional User's Guide.

The PBS server/scheduler/clients run on one of the Blue Gene front-end nodes, and MOM runs on the service node. The front-end node and service node are running Linux SuSE 9 on an IBM power processor server. There's no need to allow submission of jobs from a non-front end, non-IBM machine (e.g. desktop.) During installation of PBS, the administrator "wraps" the Blue Gene `mpirun` so that users can continue to use "mpirun" in their scripts. If you wish to limit mpirun so that it will only execute inside the PBS environment, wrap the mpiruns on the front-end node and the service node by specifying pbsrun_wrap -s, to ensure no Blue Gene partitions are spawned outside of PBS. See section 4.8.8 "Installing on IBM Blue Gene" on page 60 and section 11.10.6 "The pbsrun_wrap Mechanism" on page 440 for more information about "wrapping" `mpirun`.

IBM's `mpirun` takes care of instantiating a user's executable on the assigned partition.

All previously-defined partitions (containing midplanes) will uniformly have either "torus" or "mesh" as connection type. Therefore, users don't need to specify the connection type when submitting jobs.

On a machine with partitions P1,P2, ..., PN, partitions are reported as

```
resources_available.partition=<mom_short_name>-P1,
```

```
<mom_short_name>-P2, ..., <mom_short_name>-PN
```

and the scheduler setting of a job's pset=partition=P1 is "pset=partition=<mom_short_name>-P1". For instance:

```
pset = partition=bgsn-R011
```

11.11.2 Requirements

The Blue Gene machine must have already been fully partitioned (this is static partitioning) by the system administrator before PBS is run. PBS finds these previously-defined partitions, and schedules jobs on them. PBS will not create any new partitions (PBS does not do dynamic partitioning).

The Blue Gene administrator must have configured each partition to mount the shared file system, otherwise, mpirun calls would fail with a “login failed:” message.

There must be at least one partition defined on the system.

11.11.3 Configuration on Blue Gene

The PBS MOM calls the Blue Gene `mpirun` on the service node, which results in the Blue Gene mpirun front-end program being called which performs an “rsh” or “ssh” to the same local host in order to start up the mpirun back-end program (i.e. mpirun_be). Thus, a PBS user account on the service node must be allowed to rsh or ssh to itself, which can be done via a `$HOME/.rhosts` entry, `$HOME/.ssh/authorized_keys` entry, or `/etc/hosts.equiv` entry allowing accounts locally to rsh/ssh to themselves:
Example:

```
userA@service_node> cat $HOME/.rhosts
service_node userA
```

Or:

```
root@service_node> cat /etc/hosts.equiv
service_node
```

In order to prevent any MPI jobs from being spawned outside of PBS, it is recommended that the Blue Gene `mpirun` that is normally installed on the front-end node (not the service node) be made off-limits to users. This is to prevent any user on the front-end node from executing that `mpirun` and getting assigned partitions that are managed by PBS.

Running MPI jobs on a Blue Gene depends on the shared location in the cluster wide filesystem (CWFS) that has been set up for a site. This shared location is what is mounted on the partition as it boots up, and is accessible by the Blue Gene I/O nodes for creation,

duplication of input/output/error files. It is recommended that users create their MPI programs in such a way that input is read, and output/error files are created under this shared location.

The administrator must define a server-level resources_max.ncpus to the maximum number of ncpus available in the Blue Gene system. That way, any user who submits a job with more than this number will automatically be rejected instead of sitting around and never running.

11.11.3.1 Configuring the Blue Gene MOM

In order to prevent PBS from scheduling jobs on one or more vnodes, designate those vnodes as offline. For example,

```
# pbsnodes –o bgl_svc[R000] bgl_svc[R010]
```

The above ensures that any partition involving midplane R000 and R010 will not be assigned to a PBS job.

MOM checks the configuration file option called "$restrict_user" to determine if it needs to completely take over the bluegene partitions.

When `$restrict_user` is set to {1, on, true, yes}, any processes on the service node belonging to non-privileged, non-PBS users are killed.

In addition, if `$restrict_user` is enabled, MOM takes control of all the unreserved partitions found in the system. That is, `pbs_mom` periodically monitors each partition, and if bluegene jobs that don't belong to PBS jobs are found, then they are automatically canceled.

The "$restrict_user_exceptions" option lists up to 10 usernames whose IDs are not the system IDs (<= 999) but may still run DB2 processes. Processes belonging to these users are exempt from being killed. The special DB2 accounts, "bglsysdb" and "bgdb2cli" are automatically added to the `$restrict_user_exceptions` list. The administrator can add up to 8 names to the list.

Format:

```
$restrict_user {1, on, true, yes, 0, off, false, no}
$restrict_user_exceptions <comma-separated list of up
```

```
to 10 user names>
```

Example:

```
$restrict_user 1
```

This is FALSE by default.

```
$restrict_user_exceptions bglbar, bglfoo
```

After any change to the Blue Gene MOM's configuration files or to the Blue Gene hardware, MOM must be restarted. To start or restart the Blue Gene MOM on the service node, run the startup script:

`/etc/init.d/pbs [start, restart]`

11.11.3.2 Blue Gene Environment Variables

Before PBS is started, the administrator needs to set the following environment variables in the general .profile or .cshrc. The default values are shown.

BRIDGE_CONFIG_FILE:

# Points to the configuration file containing the machine's serial
# number and the images to load on the I/O Nodes and Compute nodes.
BRIDGE_CONFIG_FILE=/bgl/BlueLight/ppcfloor/bglsys/bin/bridge.config

DB_PROPERTY:

# Points to a configuration file that defines the control system
# database schema to be accessed by the back end mpirun.
DB_PROPERTY=/bgl/BlueLight/ppcfloor/bglsys/bin/db.properties

MMCS_SERVER_IP:

# IP address of the service node
MMCS_SERVER_IP=<Mom's full hostname>

DB2DIR:

# DB2 installation path
DB2DIR=<result of executing
"source /bgl/BlueLight/ppcfloor/bglsys/bin/db2profile;echo $DB2DIR">

DB2INSTANCE:

# The name of the DB2 database instance to connect to
DB2INSTANCE=<result of executing

"source /bgl/BlueLight/ppcfloor/bglsys/bin/db2profile;echo $DB2INSTANCE">

The PBS MOM will try to detect these environment variables. They are required for `pbs_mom` to come up as a Blue Gene MOM. The MOM will figure out a value for each variable at runtime if it has not been set in the `pbs_environment` file.

If a value for at least one of the variables cannot be determined, then MOM will exit with an appropriate message in the logs:

> 0060325:03/25/2006 12:26:52;0002;pbs_mom;n/a;dep_initialize;Could not start as a Blue Gene Mom, please provide values for the env variables BRIDGE_CONFIG_FILE, DB_PROPERTY, MMCS_SERVER_IP, DB2DIR, DB2INSTANCE in file: /var/spool/PBS/pbs_environment

On Blue Gene, in the job's executing environment, the following environment variables are always set by PBS:

```
MPIRUN_PARTITION=<partition_name>
MPIRUN_PARTITION_SIZE=<# of ncpus>
```

where MPIRUN_PARTITION is the partition assigned to the PBS job, and MPIRUN_PARTITION_SIZE is the number of CPUs making up the assigned partition.

11.11.3.3 Blue Gene Configuration Examples

Our example system's hierarchy looks like:

R_32 = 8192 CPUs (4 racks, full system bglblock)
- R0 = 4096 CPUs (2 racks)
  - R00 = 2048 CPUs (1 rack)
    - R000 = 1024 CPUs
    - R001 = 1024 CPUs
  - R01 = 2048 CPUs (1 rack)
    - R010 = 1024 CPUs
    - R011 = 1024 CPUs
- R1 = 4096 CPUs (2 racks)
  - R10 = 2048 CPUs (1 rack)
    - R100 = 1024 CPUs
      - R1000 = 256 CPUs

R1001 = 256 CPUs
R1002 = 256 CPUs
R1003 = 256 CPUs
R101 = 1024 CPUs
R11 = 2048 CPUs (1 rack)
R110 = 1024 CPUs
R111 = 1024 CPUs

11.11.3.4 Creating Blue Gene Queues by Size of Job

The system administrator creates PBS queues, and each queue is assigned some default partition size, and users are allowed to submit jobs directly to a particular queue:

```
create queue smalljobs
set queue smalljobs queue_type = Execution
set queue smalljobs
resources_default.select=128:ncpus=2
set queue smalljobs resources_max.ncpus=256
set queue smalljobs resources_min.ncpus=2

create queue midplane
set queue midplane queue_type = Execution
set queue midplane
resources_default.select=512:ncpus=2
set queue midplane resources_max.ncpus=1024
set queue midplane resources_min.ncpus=257

create queue rack
set queue rack queue_type = Execution
set queue rack resources_default.select=1024:ncpus=2
set queue rack resources_max.ncpus=2048
set queue rack resources_min.ncpus=1025

create queue half_machine
set queue half_machine queue_type = Execution
set queue half_machine
resources_default.select=2048:ncpus=2
set queue half_machine resources_max.ncpus=4096
set queue half_machine resources_min.ncpus=2049

create queue all_machine
set queue all_machine queue_type = Execution
```

```
set queue all_machine
resources_default.select=4096:ncpus=2
set queue all_machine resources_max.ncpus=8192
set queue all_machine resources_min.ncpus=4097
```

Users submit the job as follows:

```
qsub -q smalljobs <job.script>
qsub -q midplane <job.script>
qsub -q rack <job.script>
qsub -q half_machine <job.script>
qsub -q all_machine <job.script>
```

11.11.3.5 Restricting Small Jobs to Small Partitions

If a site wants to restrict small jobs to run only on small partitions (i.e. 64 CPUs or 256 CPUs), PBS should be configured so that certain queues are tied to specific vnodes. Example:

As root, edit the file `$PBS_HOME/server_priv/resourcedef` on the Blue Gene front-end node, and add a line:

```
Q       type=string_array       flag=h
```

As root, edit the file `$PBS_HOME/sched_priv/sched_config` and find the line beginning with "resources:". It will have a quoted string following the ":" with several resource names. Add the "Q" resource so it looks like:

```
resources: "ncpus, mem, arch, host, vnode, Q"
```

As root, restart the daemons on the front-end node:

```
/etc/init.d/pbs start
```

Create the following queue definitions:

```
create queue tinyjobs
set queue tinyjobs queue_type = Execution
set queue tinyjobs resources_default.select=32:ncpus=2
set queue tinyjobs resources_max.ncpus=64
```

```
set queue tinyjobs resources_min.ncpus=1
set queue tinyjobs resources_min.Q = tinyjobs
set queue tinyjobs resources_default.Q = tinyjobs
set queue tinyjobs default_chunk.Q = tinyjobs
set queue tinyjobs started= True
set queue tinyjobs enabled = True

create queue smalljobs
set queue smalljobs queue_type=Execution
set queue smalljobs resources_max.ncpus=256
set queue smalljobs resources_min.ncpus=65
set queue smalljobs resources_min.Q = smalljobs
set queue smalljobs resources_default.Q = smalljobs
set queue smalljobs default_chunk.Q = smalljobs
set queue smalljobs started= True
set queue smalljobs enabled = True
```

Add the following to the vnodes, so that the vnodes representing nodecards are assigned the small queues:

```
set node bgl_svc[R111] resources_available.Q = none
set node bgl_svc[R110] resources_available.Q = none
set node bgl_svc[R101] resources_available.Q = none
set node bgl_svc[R100#3#J216] resources_available.Q =
"smalljobs,tinyjobs"
set node bgl_svc[R100#3#J214] resources_available.Q =
"smalljobs,tinyjobs"
set node bgl_svc[R100#3#J212] resources_available.Q =
"smalljobs,tinyjobs"
set node bgl_svc[R100#3#J210] resources_available.Q =
"smalljobs,tinyjobs"
set node bgl_svc[R100#2#J209] resources_available.Q =
"smalljobs,tinyjobs"
set node bgl_svc[R100#2#J207] resources_available.Q =
"smalljobs,tinyjobs"
set node bgl_svc[R100#2#J205] resources_available.Q =
"smalljobs,tinyjobs"
set node bgl_svc[R100#2#J203] resources_available.Q =
"smalljobs,tinyjobs"
set node bgl_svc[R100#1#J117] resources_available.Q =
"smalljobs,tinyjobs"
```

```
set node bgl_svc[R100#1#J115] resources_available.Q =
"smalljobs,tinyjobs"
set node bgl_svc[R100#1#J113] resources_available.Q =
"smalljobs,tinyjobs"
set node bgl_svc[R100#1#J111] resources_available.Q =
"smalljobs,tinyjobs"
set node bgl_svc[R100#0#J108] resources_available.Q =
"smalljobs,tinyjobs"
set node bgl_svc[R100#0#J106] resources_available.Q =
"smalljobs,tinyjobs"
set node bgl_svc[R100#0#J104] resources_available.Q =
"smalljobs,tinyjobs"
set node bgl_svc[R100#0#J102] resources_available.Q =
"smalljobs,tinyjobs"
set node bgl_svc[R011] resources_available.Q = none
set node bgl_svc[R010] resources_available.Q = none
set node bgl_svc[R001] resources_available.Q = none
set node bgl_svc[R000] resources_available.Q = none
```

So if users submit the following jobs:

```
J1 qsub -q tinyjobs sleepjob
J2 qsub -q tinyjobs sleepjob
J3 qsub -q tinyjobs sleepjob
J4 qsub -q tinyjobs sleepjob
J5 qsub -q smalljobs sleepjob
J6 qsub -q smalljobs sleepjob
J7 qsub -q smalljobs sleepjob
J8 qsub -q smalljobs sleepjob
J9 qsub -q tinyjobs sleepjob
```

Once all the vnodes representing nodecards are used up, any remaining small jobs would wait until a small vnode becomes available.

11.11.3.6 Configuration Handled by PBS

The PBS MOM finds the partitions and reports them. PBS will set each vnode's "sharing" attribute to "force_excl", and will set each vnode's "resource_available.arch" to "blue-gene".

11.11.4 Hardware Changes, Starting MOM on Blue Gene

Any updates to hardware status require that the Blue Gene MOM be restarted To start or restart the Blue Gene MOM in the service mode, run the startup script:

```
/etc/init.d/pbs [start, restart]
```

11.11.5 Jobs on Blue Gene

Before a PBS job is started, pbs_mom checks the physical states of the vnodes making up the partition assigned to the job. The job will fail to run if any of the vnodes have a physical state of not “UP”. The server will be sent an updated list not containing the vnodes that are physically down. MOM constantly monitors these “downed” vnodes, and if there's a change of state, then the server will be informed.

Before a PBS job is started, pbs_mom checks the state of the partition assigned to the job.

1. It considers a partition available if it has a state of "RM_PARTITION_READY" (initialized), booting (RM_PARTITION_CONFIGURING), or RM_PARTITION_FREE (free).

2. If the partition does not have any of the states above, then the job will fail to run but will be flagged for a retry.

3. If the partition state is “READY”, meaning it has been booted, then PBS will attempt to reset the state back to “FREE”. This is needed since `mpirun` will complain if it gets a partition that is “READY” and owned by another user. If the operation of setting the state to FREE fails, then the job will fail to run and be flagged for a retry.

   NOTE: Setting the state of the partition back to FREE state will cause any Blue Gene job that is already running on the partition to be freed. This means that if there's any Blue Gene job that was spawned on that partition outside of PBS, then that Blue Gene job will automatically be killed regardless of whether or note `$restrict_user` has been set on the MOM.

4. If the (unexpected) state of the partition is "RM_PARTITION_ERROR", the vnodes encompassing the partition will be marked DOWN and the server will be made

aware of the new status of these vnodes.

Before a PBS job is started, `pbs_mom` checks the partition assigned to the job to see if a Blue Gene jobid has been instantiated on the partition, or on another overlapping partition, outside of PBS. If so and `$restrict_user` is true, the Blue Gene job is canceled before proceeding to run the PBS job. Case 4 above is an exception to this.

Even though this should not happen, pbs_mom handles the condition where two or more PBS jobs have been assigned the same partition. In this case, the second and succeeding PBS jobs will not run and eventually be held after several tries.

On Blue Gene, in the job's executing environment, the following environment variables are always set by PBS:

```
MPIRUN_PARTITION=<partition_name>
MPIRUN_PARTITION_SIZE=<# of ncpus>
```

where MPIRUN_PARTITION is the partition assigned to the PBS job, and MPIRUN_PARTITION_SIZE is the number of CPUs making up the assigned partition.

The suspend/resume feature of PBS jobs is not supported. Attempts to suspend a PBS job will return "No support for requested service".

The hold/release feature of PBS either through check_abort, restart action scripts, foregrounded or transmogrified, is supported.

On a hold request of a running job, the Blue Gene job associated with the PBS job is cancelled.

On a release request, the job is restarted with MPIRUN_PARTITION and MPIRUN_PARTITION_SIZE variables restored in its environment, pointing to an assigned partition.

When `pbs_mom` is killed with -9 (SIGKILL) and restarted with the `pbs_mom -p` option, then any Blue Gene jobs belonging to PBS jobs will not be canceled. If `pbs_mom` is restarted without the "-p" option, then the PBS job is killed with the associated Blue Gene job being canceled.

A kill -HUP of pbs_mom is a no-op on a Blue Gene. The config file is not re-read, and the vnodes list is not regenerated to be sent to server. This is to prevent any inconsistencies being introduced, especially when partitions change or disappear midway through their use by a PBS job.

The vnodes in a partition assigned to a job are allocated exclusively. Each job is run within a partition. If a job cannot statically fit in a partition, it will be treated like any job that can never run.

11.11.6 Not Supported

The suspend/resume feature of PBS jobs is not supported. Attempts to suspend a PBS job will return "No support for requested service".

The MPI integration-related utility pbs_attach is not supported.

Node grouping and placement sets are not supported.

If there is at least one Blue Gene vnode in a complex, then attempts to set node_group_enable will fail.

If a complex has no Blue Gene vnodes and has node_group_key set, then when a Blue Gene vnode is added, either no jobs will run on the Blue Gene vnode or that vnode will be marked offline.

If node_group_enable is set on a complex that does not have Blue Gene vnodes, then when a Blue Gene vnode is added to the complex, the scheduler will not schedule jobs on the Blue Gene vnodes. Further, PBS will mark the Blue Gene as offline. The server will set a comment on all affected Blue Gene vnodes explaining that you cannot have a Blue Gene in a complex with node_group_enable set to true.

If a job requests node grouping on a complex containing at least one Blue Gene vnode, the scheduler will print a log message and set a job comment saying “This job requests node grouping on a complex that contains a Blue Gene vnode and therefore will not run”.

In a heterogeneous complex containing one or more Blue Gene

vnodes and other non-Blue Gene components, if a job is submitted with a select specification requesting multiple vchunks, where one or more of the vchunks requests a Blue Gene vnode, and one or more of the vchunks requests a non-Blue Gene vnode, then the job will never run.

11.12 Support for NEC SX-8

PBS supports the following NEC features:

The NEC checkpoint facility provides the PBS job checkpointing feature.

The NEC job feature creates a NEC jobid for each PBS task. This jobid acts as an inescapable session on a single host. PBS can track MPI processes as long as they are all on one NEC machine.

PBS supports the NEC SX-8, except for the following:

Users cannot run interactive jobs.

No support for running the client commands: xpbs, xpbsmon, pbs_tclsh, or pbs_wish, directly on the SX-8. They can be used from other platforms to connect to an SX-8 system, just not directly run on the SX-8 itself.

Cycle harvesting based on load average and keyboard/mouse activity is not supported.

There is no vmem resource (NEC SX-8 machines do not use virtual memory.)

The pbs_probe command will work the same except for the following:

No files or directories related to Tcl/Tk will exist.

Permissions for PBS_EXEC and PBS_HOME will have the group write bit set.

11.13 SGI Job Container / Limits Support

PBS Professional supports 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.

For information on using Comprehensive System Accounting, see "Configuring MOM for Comprehensive System Accounting" on page 300.

11.14 Support for AIX

PBS Professional supports Large Page Mode on AIX. No additional steps are required from the PBS administrator. Certain applications (like many FEA Solvers) can benefit from using large page support. This allows programs to do considerably less page "thrashing".

Setting the PBS environment to request large page mode is not recommended because every process started by a job will use large page mode. It is better for the user to explicitly request large page mode for the processes that should use large page mode.

11.15 Job Prologue / Epilogue Programs

PBS provides the ability for the Administrator to 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 vnodes are allocated to a job, these scripts are run only by the "Mother Superior", the `pbs_mom` on the first vnode allocated. This is also where the job shell script is run. Note that both the `prologue` and `epilogue` are run under root (on UNIX) or an Admin-type account (on Windows), and neither is included in the job session, thus the `prologue` cannot be used to modify the job environment or change limits on the job.

The primary purpose of the prologue is to provide a site with some means of performing addition checking prior to starting a job. The prologue can return values to indicate:

(0)	allow the job to continue to run
(1)	abort the job and discard it
(>1)	prevent the job from starting and requeue it

Note that the prologue does not have access to the $PBS_NODEFILE environment variable.

11.15.1 Sequence of Events for Start of Job

This is the order in which events take place on an execution host at the start of a job:

1 Licenses are obtained
2 Any specified files are staged in
3 `$TMPDIR` is created
4 The job's cpusets are created
5 The prologue is executed
6 The job script is executed

11.15.2 Sequence of Events for End of Job

This is the order in which events generally take place at the end of a job:

1 The job script finishes
2 The job's cpusets are destroyed
3 The epilogue is run
4 The obit is sent to the server
5 Any specified file staging out takes place, including stdout and stderr
6 Files staged in or out are removed
7 Job files are deleted
8 FLEX licenses are returned to pool

If a `prologue` or `epilogue` script is not present, MOM continues in a normal manner. If present, the script is run with root/Administrator 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 exact 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.
- Under UNIX, the script must be owned by root, be readable and executable by root, and cannot be writable by anyone but root.
- Under Windows, the script's permissions must give "Full Access" to the local Administrators group on the local com-

puter.

The “script” may be either a shell script or an executable object file.

The `prologue` will be run immediately prior to executing the job. When job 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 one of the following special values:

-11 - Job was rerun
-12 - Job was checkpointed and aborted

11.15.3 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 includes the contents of the pbs_environment file and PBS_JOBDIR.
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. The default for this file is `/dev/null`.

output — The standard output and standard error of the scripts are connected to the files which contain the standard output and error of the job. (Under UNIX, there is one exception: 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. Interactive jobs are only supported on UNIX.)

Important: Under Windows and with some UNIX shells, accessing `arg[10]` in the epilogue requires a shift in positional parameters. The script must call the arguments with indices 0 through 8, then perform a shift /8, then access the last argument using %9%. For example:

```
cat epilogue
> #!/bin/bash
>
> echo "argv[0] = $0" > /tmp/epiargs
> echo "argv[1] = $1" >> /tmp/epiargs
> echo "argv[2] = $2" >> /tmp/epiargs
> echo "argv[3] = $3" >> /tmp/epiargs
> echo "argv[4] = $4" >> /tmp/epiargs
> echo "argv[5] = $5" >> /tmp/epiargs
> echo "argv[6] = $6" >> /tmp/epiargs
> echo "argv[7] = $7" >> /tmp/epiargs
> echo "argv[8] = $8" >> /tmp/epiargs
> echo "argv[9] = $9" >> /tmp/epiargs
> shift
> echo "argv[10] = $9" >> /tmp/epiargs
```

11.15.4 Prologue and Epilogue Time Out

When the scheduler runs a job it does not continue the cycle until the prologue has ended. To prevent an error condition within the `prologue` or `epilogue` from delaying PBS, MOM places an alarm around the script's/program's execution. The default value is 30 seconds. If the alarm sounds before the script has terminated, MOM will kill the script. The alarm value can be changed via `$prologalarm` MOM configuration parameter.

Beware that

11.15.5 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 code. 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
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. Under UNIX, which supports interactive-batch jobs (`qsub -I` option), such jobs cannot be requeued on a non-zero status, and will therefore 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. Neither prologue nor epilogue exit values are passed along as the job's exit value.

11.16 The Accounting Log

The PBS Server 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 event 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 upon daemon/service shutdown .and reopen it upon daemon/service startup.

On UNIX the Server will also close and reopen the account log file 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, 2005 the Server will be writing in the file 20050209. The following actions will cause the current accounting file to be renamed feb9 and the Server to close the file and start writing a new 20050209.

```
cd $PBS_HOME/server_priv/accounting
mv 20050209 feb9
kill -HUP 1234     (the Server's pid)
```

On Windows, to manually rotate the account log file, shut down the Server, move or rename the accounting file, and restart the Server. For example, to cause the current accounting file to be renamed feb9 and the Server to close the file and start writing a new 20050209:

```
cd "%PBS_HOME%\server_priv\accounting"
net stop pbs_server
move 20050209 feb9
net start pbs_server
```

11.16.1 Accounting Log Format

The PBS accounting file is a text file with each entry terminated by a newline. 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 record types are:

A Job was aborted by the server.

B Beginning of reservation period. If the log entry is for a reservation, the `message_text` field contains information describing the specified advance reservation. Possible information includes:

Table 26: Reservation Information

Attribute	Explanation
`owner=`ownername	Name of party who submitted the resource reservation request.
`name=`reservation_name	If submitter supplied a name string for the reservation.
`account=`account_string	If submitter supplied a string to be recorded in accounting.
`queue=`queue_name	The name of the instantiated reservation queue if this is a general resource 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 resource reservation was created; seconds since the epoch.

Table 26: Reservation Information

Attribute	Explanation
`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, in seconds since the epoch.
`duration=` reservation_duration	The duration specified or computed for the resource reservation, in seconds.
`exec_host=vnode_`list	List of each vnode with vnode-level, consumable resources allocated from that vnode. exec_host=vnodeA/P*C [+vnodeB/P * C] where P is a unique index and C is the number of CPUs assigned to the reservation, 1 if omitted.
`Authorized_Users=` user_list	The list of acl_users on the queue that is instantiated to service the reservation.
`Authorized_Groups=` group_list	If specified, the list of acl_groups on the queue that is instantiated to service the reservation.
`Authorized_Hosts=` host_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 `requester=user@host` to identify who deleted the job.

E Job ended (terminated execution). In this case, the `message_text` field contains information about the job. The end of job accounting record will not be written until all of the resources have been freed. The "end" entry in the job end

record will include the time to stage out files, delete files, and free the resources. This will not change the recorded "walltime" for the job. Possible information includes:

Table 27: PBS Job Information

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.
`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, i.e. was enqueued in an execution queue and was in the "Q" state. Reset when a job moves queues. Not affected by qaltering.
`start=`time	Time when job execution started.
`exec_host=vnode_`list	List of each vnode with vnode-level, consumable resources allocated from that vnode. exec_host=vnodeA/P*C [+vnodeB/P * C] where P is a unique index and C is the number of CPUs assigned to the job, 1 if omitted.

Table 27: PBS Job Information

Attribute	Explanation
`Resource_List.`resource=amount	List of resources requested by the reservation. Resources are listed individually as, for example: Resource_List.ncpus=16 Resource_List.mem=1048676.
`resources_used`	Resources used by the job as reported by MOM. Typically includes ncpus, mem, vmem, cput, walltime, cpupercent.
`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 SGI irix6cpuset MOM and the Altix ProPack 2.4 or 3.0 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 On Altix machines with ProPack 4, the alternate id will show the path to the job's cpuset, starting with /PBSPro/.
`end=`time	Time in seconds since epoch when this accounting record was written.
`Exit_status=`value	The exit status of the job. See "Job Exit Codes" on page 532.
`resources_used.`*`RES`*`=`*value*	Provides the aggregate amount (*value*) of specified resource *`RES`* used during the duration of the job.
accounting_id=jidvalue	CSA JID, job container ID

Table 27: PBS Job Information

Attribute	Explanation
`resource_assigned.RES=` value	**Not** a job attribute; simply a label for reporting job resource assignment. The value of `resources_assigned` reported in the Accounting records is the actual amount assigned to the job by PBS. All allocated consumable resources will be included in the "resource_assigned" entries, one resource per entry. Consumable resources include ncpus, mem and vmem by default, and any custom resource defined with the -n or -f flags. A resource will not be listed if the job does not request it directly or inherit it by default from queue or server settings. For example, if a job requests one CPU on an Altix that has four CPUs per blade/vnode and that vnode is allocated exclusively to the job, even though the job requested one CPU, it is assigned all 4 CPUs.

F Resource reservation period finished.

K Scheduler or server requested removal of the reservation. The `message_text` field contains: `requester=user@host` to identify who deleted the resource reservation.

k Resource reservation terminated by ordinary client - e.g. an owner issuing a `pbs_rdel` command. The `message_text` field contains: `requester=user@host` to identify who deleted the resource reservation.

L License information. This line in the log will contain the following fields:
Log date; record type; keyword; specification for floating license; hour; day; month; max
The following table explains each field:

Table 28: Licensing Info in Accounting Log

Field	Explanation
Log date	Date of event
record type	Indicates license info
keyword	license
specification for floating license	Indicates that this is floating license info
hour	Number of licenses used in the last hour
day	Number of licenses used in the last day
month	Number of licenses used in the last month
max	Maximum number of licenses ever used. Not dependent on server restarts.

Q Job entered a queue. For this kind of record type, 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.

Attribute	Explanation
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=vnode_list	List of each vnode with vnode-level, consumable resources allocated from that vnode. exec_host=vnodeA/P*C [+vnodeB/P * C] where P is the job number and C is the number of CPUs assigned to the job, 1 if omitted.
resource_assigned	**Not** a job attribute; instead simply a label for reporting resources assigned to a job. Consumable resources that were allocated to that job.
Resource_List.resource=amount	List of resources requested by the reservation. Resources are listed individually as, for example: Resource_List.ncpus=16 Resource_List.mem=1048676.
session=sessionID	Session number of job.
accounting_id=identifier_string	An identifier that is associated with system-generated accounting data. In the case where accounting is CSA on Altix, identifier_string is a job container identifier or JID created for the PBS job.

T Job was restarted from a checkpoint file.

U Created unconfirmed resources reservation on Server. The message_text field contains requester=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, corresponding to the resources requested and used, respectively.

Important: If a job ends between MOM poll cycles, `resources_used.`*`RES`* numbers will be slightly lower than they are in reality. For long-running jobs, the error percentage will be minor.

11.16.2 PBS Accounting and Windows

PBS will save information such as user name, group name, and account name in the accounting logs found in `PBS_HOME\server_priv\accounting`. Under Windows, these saved entities can contain space characters, thus PBS will put a quote around string values containing spaces. For example,

```
user=pbstest group=None account="Power Users"
```

Otherwise, one can specify the replacement for the space character by adding the `-s` option to the `pbs_server` command line option. This can be set as follows:

1. Bring up the `Start Menu->Settings->Control Panel->Administrative Tools->Services` dialog box (Windows 2000) or `Start Menu->Control Panel ->Performance and Maintenance->Administrative Tools->Services` dialog box (Windows XP).
2. Select `PBS_SERVER`.
3. Stop the Server
4. Specify in start parameters the option for example "`-s %20`".
5. Start the Server

This will replace space characters as "`%20`" in `user=, group=, account=` entries in accounting log file:

```
user=pbstest group=None account=Power%20Users
```

Important: If the first character of the replacement string argument to `-s` option appears in the input string itself, then that character will be replaced by its hex representation prefixed by %. For example, given:

```
account=Po%wer Users
```

Since % also appears the above entry and our replacement string is "`%20`", then replace this % with its hex representation (`%25`):

```
account="Po%25wer%20Users"
```

11.17 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 component (`pbs_server`, `pbs_mom`, and `pbs_sched`) and the PBS accounting log.

11.17.1 PBS Events

The amount of output in the PBS event logfiles depends on the specified log filters for each component. All three PBS components 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. (Note that this is opposite to the scheduler's log filters, which specify what to leave out.) The available events, and corresponding decimal and hexadecimal values are shown below. When these appear in the log file, they are tagged with the hexadecimal shown, without a preceding "0x".

Table 29: PBS Events

Value	Hex	Event Description
1	0001	Internal PBS errors.
2	0002	System (OS) errors, such as malloc failure.
4	0004	Administrator-controlled events, such as changing queue attributes.
8	0008	Job related events: submitted, ran, deleted, ...
16	0010	Job resource usage.

Table 29: PBS Events

Value	Hex	Event Description
32	0020	Security related, e.g. attempts to connect from an unknown host.
64	0040	When the Scheduler was called and why.
128	0080	Debug messages. Common messages..
256	0100	Debug level 2.
512	0200	Reservation related messages
1024	0400	Debug level 3. Most prolific debug messages

For example, if you want to log all events except those at levels 512 and 1024 (hex 0x200 and 0x400), you would use a log level of 511. This is 256 + 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1. If you want to log events at levels 1, 2, and 16, you would set the log level to 19.

The event logging mask is controlled differently for the different components. The following table shows the log event parameter for each, and page reference for details.

PBS	Attribute and Reference	Notes
Server	See "log_events" on page 187.	Takes effect immediately with `qmgr`
MOM	See "$logevent <mask>" on page 265.	Requires `SIGHUP` to MOM
Scheduler	See "log_filter" on page 319.	Requires `SIGHUP` to Scheduler

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.

11.17.1.1 Scheduler Commands

These commands provide the scheduler a hint as to why a scheduling cycle is being started. The following table shows commands from the server to the scheduler.

Table 30: Commands from Scheduler to Server

Value	Event Description
1	New job enqueued
2	Job terminated
3	Scheduler time interval reached
4	Cycle again after scheduling one job
5	Scheduling command from operator or manager
7	Configure
8	Quit (qterm -s)
9	Ruleset changed
10	Schedule first
11	Schedule a job reservation
12	Scheduler a job (qrun command has been given)

11.17.2 Event Logfiles

Each PBS component maintains separate event logfiles. The logfiles default to a file with the current date as the name in the *PBS_HOME/(component)_logs* directory. This location can be overridden with the "`-L pathname`" option where pathname must be an absolute path.

The log filters work differently: the server and MOM log filters specify what to put in the log file, and the scheduler's log filter specifies what to keep out of its log files.

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.

On UNIX, all components will close and reopen the same named log file on receipt of **SIGHUP**. The process identifier (PID) of the component 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 $PBS_HOME/component_logs
mv current archive
kill -HUP 'cat ../component_priv/component.lock'
```

On Windows, manual rotation of the event log files can be accomplished by stopping the particular PBS service component for which you want to rotate the logfile, moving the file, and then restarting that component. For example:

```
cd "%PBS_HOME%\component_logs"
net stop pbs_component
move current archive
net start pbs_component
```

Each daemon will write its version and build information to its event logfile each time it is started or restarted, and also when the logfile is automatically rotated out. The pbs_version information and build information will appear in individual records. These records will contain the substrings:

pbs_version = <PBSPro_stringX.stringY.stringZ.5-digit seq>
build = <status line from config.status, etc>

Example:

pbs_version = PBSPro_9.1.0.63106
build = '--set-cflags=-g -O0' --enable-security=KCRYPT ...

11.17.3 Event Logfile Format

Each component 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 a bitmask for the type of event which triggered the event logging. It corresponds to the bit position, 0 to n, of each log event in the event mask of the PBS component writing the event record. See section 11.17.1 “PBS Events” on page 480 for a description of the event mask.

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.

All messages are associated with an object_type, where the object_type is the type of object which the message is about. The following lists each possible object_type:

Svr	for server
Que	for queue
Job	for job
Req	for request
Fil	for file
Act	for accounting string
Node	for vnode or host
Resv	for reservation
Sched	for scheduler

The object_name is the name of the specific object. message_text field is the text of the log message.

PBS can log per-vnode cputime usage. The mother superior logs cputime in the format “hh:mm:ss” for each vnode of a multi-vnode job. The logging level of these messages is PBSEVENT_DEBUG2.

To append job usage to standard output for an interactive job, use a shell script for the epilogue which contains the following:

```
#!/bin/sh
tracejob -sl $1 | grep 'cput'
```

11.18 Using the UNIX syslog Facility

Each PBS component logs various levels of information about events in its own log file. While having the advantage of a concise location for the information from each component, the disadvantage is that in a complex, the logged information is scattered across each execution host. The UNIX syslog facility can be useful.

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 components:

`PBS_LOCALLOG`=x	Enables logging to local PBS log files. Only possible when logging via syslog feature is enabled. 0 = no local logging 1 = local logging enabled
`PBS_SYSLOG`=x	Controls the use of syslog and syslog "facility" under which the entries are logged. If x is: 0 - no syslogging 1 - logged via `LOG_DAEMON` facility 2 - logged via `LOG_LOCAL0` facility 3 - logged via `LOG_LOCAL1` facility ... 9 - logged via `LOG_LOCAL7` facility
`PBS_SYSLOGSEVR`=y	Controls the severity level of messages that are logged; see `/usr/include/sys/syslog.h`. If y is: 0 - only `LOG_EMERG` messages are logged 1 - messages up to `LOG_ALERT` are logged ... 7 - messages up to `LOG_DEBUG` are logged

Important: `PBS_SYSLOGSEVR` is used in addition to PBS's `log_events` mask which controls the class of events (job, vnode, ...) that are logged.

11.19 Managing Jobs

11.19.1 UNIX 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.

PBS passes 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 start with a `#!` line, the wrong shell may be used to interpret the script and thus produce 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.

11.19.2 Managing Jobs on Machines with 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. The cpusets are created with the name of the jobid for which they are created.

11.19.3 Job IDs

The largest possible job ID is the 7-digit number 9999999. After this has been reached, job IDs start again at zero.

11.19.4 Job States

Job states are abbreviated to one character.

Table 31: Job States

State	Description
B	Job arrays only: job array has started
E	Job is exiting after having run
H	Job is held. A job is put into a held state by the server or by a user or administrator. A job stays in a held state until it is released by a user or administrator.
Q	Job is queued, eligible to run or be routed
R	Job is running
S	Job is suspended by server. A job is put into the suspended state when a higher priority job needs the resources.
T	Job is in transition (being moved to a new location)
U	Job is suspended due to workstation becoming busy
W	Job is waiting for its requested execution time to be reached or job specified a stagein request which failed for some reason.
X	Subjobs only; subjob is finished (expired.)

11.19.4.1 Job Substates

Table 32: Job Substates

Substate Number	Substate Description
00	Transit in, prior to waiting for commit
01	Transit in, waiting for commit
02	transiting job outbound, not ready to commit
03	transiting outbound, ready to commit
10	job queued and ready for selection
11	job queued, has files to stage in
14	job staging in files before waiting
15	job staging in files before running
16	job stage in complete
20	job held - user or operator
22	job held - waiting on dependency
30	job waiting until user-specified execution time
37	job held - file stage in failed
41	job sent to MOM to run
42	Running
43	Suspended by Operator or Manager
44	job sent to run under Globus
45	Suspended by Scheduler
50	Server received job obit
51	Staging out stdout/err and other files
52	Deleting stdout/err files and staged-in files

Table 32: Job Substates

Substate Number	Substate Description
53	Mom releasing resources
54	job is being aborted by server
56	(Set by MOM) Mother Superior telling sisters to kill everything
57	(Set by MOM) job epilogue running
58	(Set by MOM) job obit notice sent
59	Waiting for site "job termination" action script
60	Job to be rerun, MOM sending stdout/stderr back to Server
61	Job to be rerun, staging out files
62	Job to be rerun, deleting files
63	Job to be rerun, freeing resources
70	Array job has begun
153	(Set by MOM) Mother Superior waiting for delete ACK from sisters

Chapter 12
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 identifies all the user commands, and the right column identifies all the administrator commands. (The user commands are described in detail in the **PBS Professional User's Guide**.)

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 11.7.7 "External Security" on page 427.)

Some of the PBS commands are intended to be used only 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 `pbs-report` command, you can add read access to the `server_priv/accounting` directory, enabling the command to report job-specific information. Be cautioned that all job information will then be available to all users. Likewise, opening access to the

accounting records will permit additional information to be printed by the `tracejob` command, which normal users would not have permissions to view. In either case, an administrator-type user (or UNIX root) always has read access to these data.

Most commands, when given the sole option "--version", will output the version of PBS for that command and exit. For example,

```
qmgr --version
```

will cause the `qmgr` command to output version information and exit. See each command's manual page.

Under Windows, use double quotes when specifying arguments to PBS commands.

Table 33: PBS Professional User and Manager Commands

User Commands		Administrator Commands	
Command	**Purpose**	**Command**	**Purpose**
`nqs2pbs`	Convert from NQS	`pbs-report`	Report job statistics
`pbs_rdel`	Delete Adv. Reservation		
`pbs_rstat`	Status Adv. Reservation	`pbs_hostn`	Report host name(s)
`pbs_password`	Update per user / per server password[1]	`pbs_migrate_users`	Migrate per user / per server passwords [1]
`pbs_rsub`	Submit Adv.Reservation	`pbs_probe`	PBS diagnostic tool
`pbsdsh`	PBS distributed shell	`pbs_rcp`	File transfer tool
`qalter`	Alter job	`pbs_tclsh`	TCL with PBS API
`qdel`	Delete job	`pbsfs`	Show fairshare usage
`qhold`	Hold a job	`pbsnodes`	Node manipulation
`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`	Manager interface
`qselect`	Select jobs by criteria	`qrerun`	Requeue running job

Table 33: PBS Professional User and Manager Commands

User Commands		Administrator Commands	
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
tracejob	Report job history	qterm	Shut down PBS
xpbs	Graphical User Interface	xpbsmon	GUI monitoring tool

Notes: 1 Available on Windows only.

12.1 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 and function properly. 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

12.2 The pbs_migrate_users Command

During a migration upgrade in Windows environments, if the Server attribute single_signon_password_enable is set to "true" in both the old Server and the new Server, the per-user/per-server passwords are not automatically transferred from an

old Server to the new Server. The pbs_migrate_users command is provided for migrating the passwords. (Note that users' passwords on the old Server are not deleted.) The command usage is:

```
pbs_migrate_users old_server[:port] new_server[:port]
```

The exit values and their meanings are:

0	success
-1	writing of passwords to files failed.
-2	communication failures between old Server and new Server
-3	single_signon_password_enable not set in either old Server or new Server.
-4	the current user is not authorized to migrate users

12.3 The pbs_rcp vs. scp Command

The **pbs_rcp** command is used internally by PBS as the default file delivery mechanism. PBS can be directed to use Secure Copy (**scp**) by so specifying in the PBS global configuration file. Specifically, to enable scp, set the PBS_SCP parameter to the full path of the local scp command, as described in the discussion of “pbs.conf” on page 403.) This should be set on all vnodes where there is or will be a PBS MOM running. MOMs already running will need to be stopped and restarted.

12.4 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 reads basic information from the pbs.conf file, and the values of any of the following environment variables 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.

Important: The pbs_probe command is currently only available on UNIX; in Windows environments, use the pbs_mkdirs command instead.

The pbs_probe command usage is:

```
pbs_probe [ -f | -v ]
```

If no options are specified, `pbs_probe` runs in "report" mode, in which it will report on any errors in the PBS infrastructure files that it detects. The problems are categorized, and a list of the problem messages in each category are output. Those categories which are empty do not show in the output.

The available options, and description of each, follows.

Option	Description
`-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.

12.5 The pbsfs (PBS Fairshare) Command

The **`pbsfs`** command allows the Administrator to display or manipulate PBS fairshare usage data. The `pbsfs` command can only be run as root (UNIX) or a user with Administrator privilege (Windows). If the command is to be run with options to alter/update the fairshare data, the Scheduler must not be running. If you terminate the Scheduler, be sure to restart it after using the `pbsfs` command.

For printing, the scheduler can be running, but the data may be stale. To make sure the data isn't stale when being printed, sending a kill -HUP to the scheduler will force the scheduler to write out its internal cache.

Important: If the Scheduler is killed, it will lose any new fairshare data since the last synchronization. For suggestions on minimizing or eliminating possible data loss, see section 9.15.11 "Viewing and Managing Fairshare Data" on page 361.

The command usage is:

```
pbsfs [ -d | -e | -p | -t ]
pbsfs [ -c entity1 entity2 ] [ -g entity ]
```

```
[ -s entity usage_value ]
```

The available options, and description of each, follows.

Option	Description	Scheduler: Up/Down
`-c` *`entity1 entity2`*	Compare two *entities* and print the most deserving entity.	Up
`-d`	Decay the fairshare tree (divide all values in half)	Down
`-e`	Trim fairshare tree to include only entries in `resource_group` file	Down
`-g` *`entity`*	Print all data for *entity* and path from the root of tree to node.	Up
`-p`	Print the fairshare tree in a flat format (default format).	Up
`-s` *`entity usage_value`*	Set *entity's* usage value to *usage_value*. Note that editing a non-leaf node is ignored. All non-leaf usage values are calculated each time the Scheduler is run or HUPed.	Down
`-t`	Print the fairshare tree in a hierarchical format.	Up

There are multiple parts to a fairshare node and you can print these data in different formats.The data displayed is:

Data	Description
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 leaves sum to 100%. If all of the nodes are summed, the result will be greater then 100%. Only the leaves of the tree are fairshare entities.

usage / perc	The value the Scheduler will use to 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 entities by starting at the root, and working toward the leaves, to determine which has the higher priority.
resource	Resource for which usage is accumulated for the fairshare calculations. Default is `cput` (cpu seconds) but can be changed in `sched_config` .

Whenever the fairshare usage database is changed, the original database is saved with the name "`usage.bak`". Only one backup will be made.

Subjobs are treated as regular jobs in the case of fairshare. Fairshare data may not be accurate for job arrays, because subjobs are typically shorter than the scheduler cycle, and data for them can be lost.

12.5.1 Trimming the Fairshare Data

Fairshare usage data may need to be trimmed because of the way the scheduler deals with unknown entities which have usage data. If the scheduler finds an entity which has usage data, but is not in the resource_group file, it will add it to the "unknown" group. This is sometimes the result of a typo. It will also be be what happens to accounts that are no longer in a group. Trimming the fairshare tree is a good way to get rid of these.

The recommended set of steps to use `pbsfs` to trim fairshare data are as follows:

UNIX:

First send a HUP signal to the Scheduler to force current fairshare usage data to be written, then terminate the Scheduler:

```
kill -HUP pbs_sched_PID
kill pbs_sched_PID
```

Windows:

```
net stop pbs_sched
```

Now you can modify the $*PBS_HOME*/sched_priv/resource_group file if needed. When satisfied with it, run the `pbsfs` command to trim the fairshare tree:

```
pbsfs -e
```

Lastly, restart the Scheduler:

UNIX:

```
$PBS_EXEC/sbin/pbs_sched
```

Windows:

```
net start pbs_sched
```

12.6 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 Professional external API library calls. This enables the user to write TCL scripts which utilize the PBS Professional API to query information. For usage see the pbs_tclapi(3B) manual page, and the **PBS Professional External Reference Specification.**

The pbs_tclsh command is supplied with the standard PBS binary. Users can make queries of MOM using this utility, for example:

```
% pbs_tclsh
tclsh> openrm <hostname>
<fd>
tclsh> addreq <fd> "loadave"
tclsh> getreq <fd>
5.0
tclsh> closereq <fd>
```

12.7 The pbsnodes Command

The **pbsnodes** command is used to query the status of hosts, or to mark hosts OFFLINE or FREE. The pbsnodes command obtains host information by sending a request to the PBS server.

To print the status of the specified host(s), run pbsnodes without options and with a list of hosts (and optionally the -s option.)

To print the command usage, run pbsnodes with no options and no arguments.

When the pbsnodes command is run with Manager or Operator privilege, it will output version information for each node specified by the command.

PBS Manager or Operator privilege is required to execute `pbsnodes` with the `-c`, `-o`, or `-r` options.

To remove a host from the scheduling pool, mark it OFFLINE. If a node has been marked DOWN, the server will mark it FREE the next time it can contact the MOM.

For hosts with multiple vnodes, `pbsnodes` operates on a host and all of its vnodes, where the hostname is resources_available.host. See the `-v` option.

To act on vnodes, use the `qmgr` command.

Syntax:

```
pbsnodes [ -c | -o | -r ] [-s server]
   hostname [hostname ...]

pbsnodes [ -l ] [-s server]

pbsnodes -a [ -v ] [-s server]
```

Options::

Option	Description
(no options)	If neither options nor a host list is given, the `pbsnodes` command prints usage syntax.

Option	Description
`-a`	Lists all hosts and all their attributes (available and used.) When listing a host with multiple vnodes: 1. The output for the jobs attribute lists all the jobs on all the vnodes on that host. Jobs that run on more than one vnode will appear once for each vnode they run on. 2. For consumable resources, the output for each resource is the sum of that resource across all vnodes on that host. 3. For all other resources, e.g. string and boolean, if the value of that resource is the same on all vnodes on that host, the value is returned. Otherwise the output is the literal string `"<various>"`.
`-c host list`	Clears OFFLINE and DOWN from listed hosts. The listed hosts will become FREE if they are online, or remain DOWN if they are not (for example, powered down.) Requires PBS Manager or Operator privilege.
`host list`	Prints information for the specified host(s).
`-l`	Lists all hosts marked as DOWN or OFFLINE. Each such host's state and comment attribute (if set) is listed. If a host also has state STATE-UNKNOWN, that will be listed. For hosts with multiple vnodes, only hosts where all vnodes are marked as DOWN or OFFLINE are listed.
`-o host list`	Marks listed hosts as OFFLINE even if currently in use. This is different from being marked DOWN. A host that is marked OFFLINE will continue to execute the jobs already on it, but will be removed from the scheduling pool (no more jobs will be scheduled on it.) Requires PBS Manager or Operator privilege.
`-r host list`	Clears OFFLINE from listed hosts.
`-s server`	Specifies the PBS *`server`* to which to connect.

Option	Description
`-v`	**Can only be used with the `-a` option.** Prints one entry for each vnode in the PBS complex. (Information for all hosts is displayed.) The output for the jobs attribute for each vnode lists the jobs executing on that vnode. The output for resources and attributes lists that for each vnode.

12.8 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.

12.9 The tracejob Command

PBS includes the **`tracejob`** utility to extract daemon/service logfile messages for a particular job (from all log files available on the local host) and print them sorted into chronological order.

Important: By default a normal user does not have access to the accounting records, and so information contained therein will not be displayed. However, if an administrator or UNIX root runs the tracejob command, this data will be included.

Usage for the `tracejob` command is:

```
tracejob [-a|s|l|m|v][-w cols][-p path][-n days][-f filter]
         [-c count] jobid
```

Note: for an array job, the job ID must be enclosed in double quotes.

The available options, and description of each, follows.

Option	Description
`-a`	Do not report accounting information.
`-c <count>`	Set excessive message limit to count. If a message is logged at least count times, only the most recent message is printed. Default for count is 15.
`-f <filter>`	Do not include logs of type filter. The -f option can be used more than once on the command line. filter: error, system, admin, job, job_usage, security, sched, debug, debug2
`-l`	Do not report scheduler information.
`-m`	Do not report MOM information.
`-n <days>`	Report information from up to days days in the past. Default is 1 = today.
`-p <path>`	Use path as path to PBS_HOME on machine being queried.
`-s`	Do not report server information.
`-w <cols>`	Width of current terminal. If not specified by the user, tracejob queries OS to get terminal width. If OS doesn't return anything, default is 80.
`-v`	Verbose. Report more of tracejob's errors than default.
`-z`	Disable excessive message limit. Excessive message limit is enabled by default.

For more information, see man(8) tracejob.

The following example requests all log messages for a particular job from today's (the default date) log file. Note that the 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 schedu**L**er log files).

```
tracejob 475
Job: 475.pluto.domain.com
03/10/2005 14:29:15 S enqueuing into workq, state 1 hop 1
03/10/2005 14:29:15 S Job Queued at request of james, owner=
     james@mars.domain.com, job name = STDIN
03/10/2005 15:06:30 S Job Modified at request of Scheduler
03/10/2005 15:06:30 L Considering job to run
03/10/2005 15:06:30 S Job Run at request of Scheduler
03/10/2005 15:06:32 L Job run on node mars
03/10/2005 15:06:32 M Started, pid = 25282
03/10/2005 15:06:32 M Terminated
03/10/2005 15:06:32 M task 1 terminated
03/10/2005 15:06:32 M kill_job
03/10/2005 15:06:32 S Obit received
03/10/2005 15:06:32 S dequeuing from workq, state 5
03/10/2005 15:06:32 A user=jwang group=mygroup 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
```

12.10 The qdisable Command

The `qdisable` command directs that the designated queue should no longer accept batch jobs. If the command is successful, the queue will no longer accept Queue Job requests which specified the now-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 ...
```

12.11 The qenable Command

The `qenable` command directs that the designated queue 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 now-enabled queue will accept Queue Job requests which specify the queue. The command usage is:

```
qenable destination ...
```

12.12 The qstart Command

The `qstart` command directs that the designated queue should process batch jobs. If the queue is an execution queue, the Server will begin to schedule jobs that reside in the queue for execution. If the designated queue is a routing queue, the Server will begin to route jobs from that queue. The command usage is:

```
qstart destination ...
```

12.13 The qstop Command

The `qstop` command directs that the designated queue should stop processing batch jobs. If the designated queue is an execution queue, the Server will cease scheduling jobs that reside in the queue for execution. If the queue is a routing queue, the Server will cease routing jobs from that queue. The command usage is:

```
qstop destination ...
```

12.14 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 rerunnable then the rerun request will fail for that job. (See also the discussion of the `-r` option to `qsub` in the **PBS Professional User's Guide**.) The command usage is:

```
qrerun [ -W force ] jobID [ jobID ...]
```

Note: for array jobs, the job IDs must be enclosed in double quotes.

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 vnode on which the job is executing is unreachable.

The qrerun command can be used on a job array, a subjob, or a range of subjobs. If the qrerun command is used on a job array, all of that array's currently running subjobs and all of its completed and deleted subjobs are requeued.

12.15 The qrun Command

The **`qrun`** command is used to force a Server to initiate the execution of a batch job. The job can be run regardless of scheduling position, resource requirements and availability, or state; see the -H option. You can overload CPUs using this command. The command usage is:

```
qrun [ -a ] [ -H host-spec ] jobID [ jobID ...]
```

Note: for array jobs, some shells require that job IDs be enclosed in double quotes.

The available options, and description of each, follows.

Option	Description
`-a`	Specifies that the `qrun` command will exit before the job actually starts execution.
`-H host-spec`	Specifies the vnode(s) within the complex on which the job(s) are to be run. The *host-spec* argument is a plus-separated list of vnode names, e.g. VnodeA+VnodeB+VnodeC. Resources can be specified in this fashion: VnodeA:mem=100kb:ncpus=1+VnodeB:mem=100kb:ncpus=2

See section 4.3.1 “Rules for Submitting Jobs” on page 31 of the **PBS Professional User’s Guide** for detailed information on requesting resources and placing jobs on vnodes.

No -H hosts option	If the operator issues a qrun request of a job without -H hosts, the server will make a request of the scheduler to run the job immediately. The scheduler will run the job if the job is otherwise runnable by the scheduler: The queue in which the job resides is an execution queue and is started. The job is in the queued state. Either the resources required by the job are available, or preemption is enabled and the required resources can be made available by preempting jobs that are running.
-H hosts option	If the -H hosts option is used, the Server will immediately run the job on the named hosts, regardless of current usage on those vnodes.
-H hosts option with list of vnodes	If a “+” separated list of hosts is specified in the Run Job request, e.g. VnodeA+VnodeB+... the Scheduler will apply one requested chunk from the select directive in round-robin fashion to each vnode in the list.
-H hosts option with list of vnodes and resource specification	If a “+” separated list of hosts is specified in the Run Job request, and resources are specified with vnode names, e.g. NodeA:mem=100kb:ncpus=1+vnodeB:mem=100kb:ncpus=2, the Scheduler will apply the specified allocations and the select directive will be ignored. Any single resource specification will result in the job’s select directive being ignored.

A qrun command issued with the -H option may oversubscribe resources on a vnode, but it will not override the exclusive/shared allocation of a vnode. If a job is already running and the vnode is allocated to that prior job exclusively due to an explicit request of the job or due to the vnode's "sharing" attribute setting, an attempt to qrun an additional job on that vnode will result in the qrun being rejected and the job being left in the Queued state.

The qrun command can be used on a subjob or a range of subjobs, but not on a job array. When it is used on a range of subjobs, the non-running subjobs in that range are run.

12.16 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 173.

12.17 The qterm Command

The **`qterm`** command is used to shut down PBS, and is discussed in detail earlier in this book, in section 11.4.8 "Stopping PBS" on page 417.

12.18 The pbs_wish Command

The **`pbs_wish`** command is a version of TK Window Shell linked with a wrapped versions of the PBS Professional external API library. For usage see the `pbs_tclapi(3B)` manual page, and the **PBS Professional External Reference Specification.**

12.19 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` options of the `qstat` command. (For details see "Displaying Job Comments" in the **PBS Professional User's Guide**.) Operators and managers may use the `-W` option of the `qalter` command, for example

```
qalter -W comment="some text" job_id
```

The qalter command can be used on job array objects, but not on subjobs or ranges of subjobs. Note also that when used on a job array, the job ID must be enclosed in double quotes. See "qalter: Altering a Job Array" on page 164 of the **PBS Professional User's Guide**.

12.20 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: The pbs-report command is not available on Windows.

Before first using pbs-report, the Administrator is advised to tune the pbs-report configuration to match the local site. This can be done by editing the file PBS_EXEC/lib/pm/PBS.pm.

Important: If job arrays are being used, the pbs-report command will produce errors including some about uninitialized variables. It will report on the job array object as well as on each subjob.

12.20.1 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.

--csv character
: Have the output be separated by the specified character. Currently only the "|" is supported. Character must be enclosed in double quotes.

--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).

--explainwait
: Print a reason for why jobs had to wait before running.

--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 all options 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`. Note that if specific queues are defined via the `@QUEUES` line in `PBS.pm`, then only those queues will be displayed. Leaving that parameter blank allows all queues to be displayed.

--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.

To calculate muda:

1. Runtime is job end - job start.
2. If suspend is greater than 10 seconds,
   then suspend is runtime - walltime.
   Otherwise suspend is set to zero.
3. If cput is not zero, then muda is suspend/cput.
   Otherwise muda is zero.

--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). Job arrays will not be displayed, but subjobs will be displayed.

--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, --wallmax, --wallmin --waitmax, --waitmin, — 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; it may be used by a custom job front-end which enforces 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 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 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.

12.20.2 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 Professional User's 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 complex. 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. 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.

12.20.3 pbs-report Complex 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.

12.21 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`.

12.21.1 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:

```
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: See the manual page for xpbs, xpbs(1B), for a complete description of all xpbs functions.

Running xpbs will initialize the X resource database in order from the following sources:

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: PBS_EXEC/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:

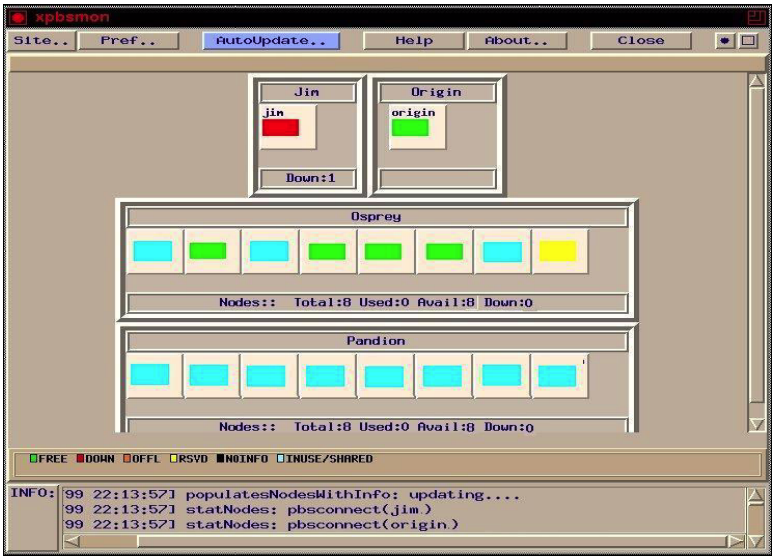
```
xpbsrc*backgroundColor: blue
*backgroundColor: green
```

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 Professional User's Guide**, Chapter 5, "Using the xpbs GUI".

12.22 The xpbsmon GUI Command

xpbsmon is the vnode 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 (vnodes).



The system Administrator needs to define the site's information in a global X resources file, *PBS_EXEC/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 vnodes (or execution hosts) where jobs on a particular Server run, and the list of queries that are communicated to each vnode's pbs_mom. If vnode queries have been specified, the host where xpbsmon is running must have been given explicit permission by the pbs_mom 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 *PBS_EXEC/lib/xpbsmon/* directory.

12.23 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] ... ]
```

Note that Under Windows, if the pbskill command is used to terminate the MOM service, it may leave job processes running, which if present, will prevent a restart of MOM (a "network is busy" message will be reported). This can be resolved by manually killing the errant job processes via the Windows task manager.

Chapter 13

Example Configurations

Up to this point in this manual, we have seen many examples of how to configure the individual PBS components, 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. Several configuration models are discussed, followed by several complex examples of specific features.

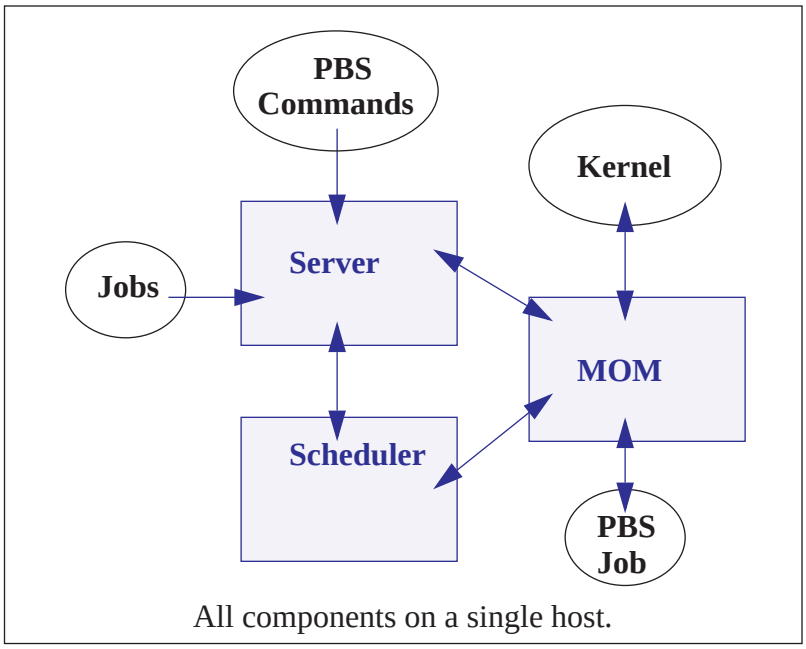
Single Vnode System
Single Vnode System with Separate PBS Server
Multi-vnode complex
Complex Multi-level Route Queues (including group ACLs)
Multiple User ACLs

For each of these possible configuration models, the following information is provided:

General description for the configuration model
Type of system for which the model is well suited
Contents of Server nodes file
Any required Server configuration
Any required MOM configuration
Any required Scheduler configuration

13.1 Single Vnode System

Running PBS on a single vnode/host as a standalone 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 components run on the same host, which is the same host on which jobs will be executed, as shown in the figure below.



All components on a single host.

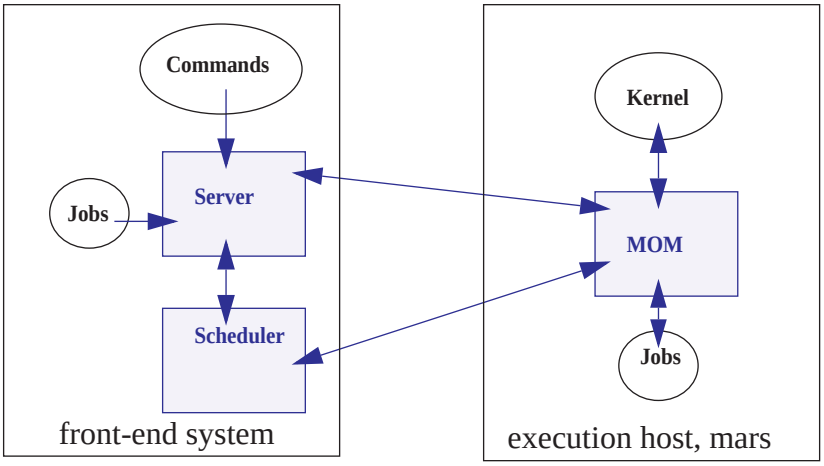
For this example, let's assume we have a 32-CPU server machine named "mars". We want users to log into mars and jobs will be run via PBS on mars.

In this configuration, the server's default `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`

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.

13.2 Separate Server and Execution Host

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 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 use `qmgr(1B)` to manipulate the contents to contain one vnode: `mars`. 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 host (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/sched_config` file, the following options would need to be set:

```
load_balancing: true all
```

13.3 Multiple Execution Hosts

The multi-vnode complex 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 execution hosts. The PBS Server and Scheduler are typically run on the front-end system, and a MOM is run on each of the execution hosts, as shown in the diagram to the right.

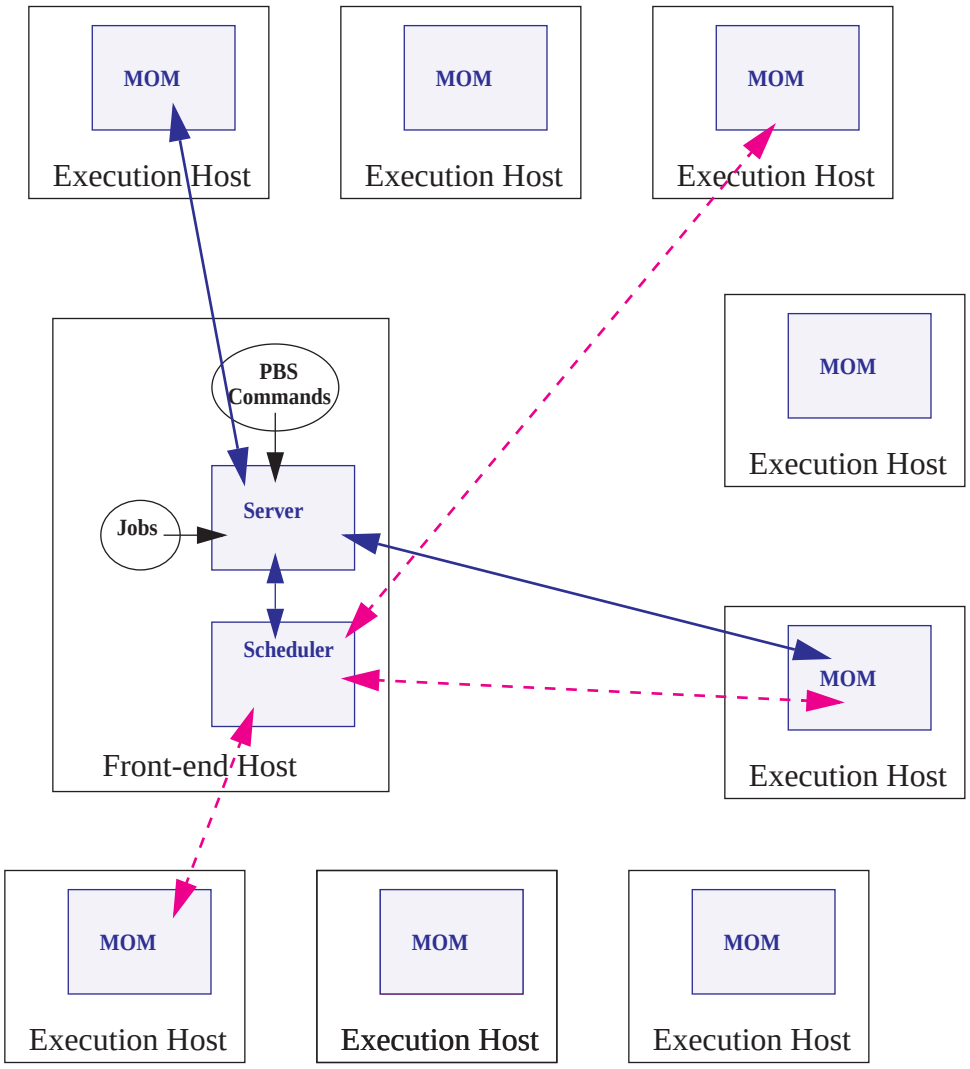
In this model, the server's `nodes` file will need to contain the list of all the vnodes in the complex.

The MOM `config` file on each vnode will need two static resources added, to specify the target load for each vnode. If we assume each of the vnodes in our "planets" cluster is a 32-processor system, then the following example shows what might be desirable ideal and maximum load 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 vnodes, making sure not to run two jobs in a row on the same vnode (round robin vnode 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-vnode complex configuration wherein the Scheduler and Server communicate with the MOMs on the execution hosts. 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.

13.4 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 group IDs. In other words, users request the architecture they want, and PBS finds 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 astronomy department, so only members of that group should be permitted into that queue. Given these requirements, let’s look at how we would set up such a 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 (which 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 select=arch=sv2:ncpus=4 testjob
```

13.5 External Software License Management

PBS Professional can be configured to schedule jobs based on externally-controlled licensed software. A detailed example is provided in section 10.7.4 "Example of Floating, Externally-managed License with Features" on page 392.

13.6 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 wants 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:
```

Chapter 14
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/UserArea. The last section in this chapter gives important information on how to get additional assistance from the PBS Support staff.

14.1 Finding PBS Version Information

Use the `qstat` command to find out what version of PBS Professional you have.

```
qstat -fB
```

14.2 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 12.4 "The pbs_probe Command" on page 494.

14.3 Job Exit Codes

The exit value of a job may fall in one of three ranges: X < 0, 0 <=X < 128, X >=128.

X < 0:
This is a PBS special return value indicating that the job could not be executed. These negative values are listed in the table below.

0 <= X < 128 (or 256):
This is the exit value of the top process in the job, typically the shell. This may be the exit value of the last command executed in the shell or the .logout script if the user has such a script (csh).

X >= 128 (or 256 depending on the system)
This means the job was killed with a signal. The signal is given by X modulo 128 (or 256). For example an exit value of 137 means the job's top process was killed with signal 9 (137 % 128 = 9).

	Name	Description
0	JOB_EXEC_OK	job exec successful
-1	JOB_EXEC_FAIL1	Job exec failed, before files, no retry
-2	JOB_EXEC_FAIL2	Job exec failed, after files, no retry
-3	JOB_EXEC_RETRY	Job execution failed, do retry
-4	JOB_EXEC_INITABT	Job aborted on MOM initialization
-5	JOB_EXEC_INITRST	Job aborted on MOM init, chkpt, no migrate
-6	JOB_EXEC_INITRMG	Job aborted on MOM init, chkpt, ok migrate
-7	JOB_EXEC_BADRESRT	Job restart failed
-8	JOB_EXEC_GLOBUS_INIT_RETRY	Init. globus job failed. do retry
-9	JOB_EXEC_GLOBUS_INIT_FAIL	Init. globus job failed. no retry
-10	JOB_EXEC_FAILUID	invalid uid/gid for job
-11	JOB_EXEC_RERUN	Job rerun
-12	JOB_EXEC_CHKP	Job was checkpointed and killed

	Name	Description
-13	JOB_EXEC_FAIL_PASSWORD	Job failed due to a bad password
-14	JOB_EXEC_RERUN_ON_SIS_FAIL	Job was requeued (if rerunnable) or deleted (if not) due to a communication failure between Mother Superior and a Sister

The PBS Server logs and accounting logs record the exit status of jobs. Zero or positive exit status is the status of the top level shell. 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 job 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.

14.4 Common Errors

14.4.1 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.

14.4.2 Vnodes Down

The PBS Server determines the state of vnodes (up or down), by communicating with MOM on the vnode. The state of vnodes may be listed by two commands: qmgr and pbsnodes.

```
qmgr
Qmgr: list node @active
pbsnodes -a
Node jupiter
        state = state-unknown, down
```

A vnode in PBS may be marked "down" in one of two substates. For example, the state above of vnode "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 vnode. 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 vnode is still marked "state-unknown, down" after 10+ minutes, either the vnode 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 vnode.

If the vnode is listed as

```
pbsnodes -a
Node jupiter
        state = down
```

then the Server has been able to ping MOM on the vnode in the past, but she has not responded recently. The Server will send a "ping" PBS message to every free vnode each ping cycle, 10 minutes. If a vnode does not acknowledge the ping before the next cycle, the Server will mark the vnode down.

14.4.3 Requeueing a Job "Stuck" on a Down Vnode

PBS Professional will detect if a vnode 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 rerunnable" (i.e. via the `qsub -r n` option), then the job will be deleted rather than requeued. If the affected vnode is vnode 0 (Mother Superior), the requeue will occur quickly. If it is another vnode 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 188), 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
```

14.4.4 File Stagein Failure

When stagein fails, the job is placed in a 30-minute wait to allow the user time to fix the problem. Typically this is a missing file or a network outage. Email is sent to the job owner when the problem is detected. Once the problem has been resolved, the job owner or the Operator may remove the wait by resetting the time after which the job is eligible to be run via the `-a` option to `qalter`. The server will update the job's comment with information about why the job was put in the wait state. The job's exec_host string is cleared so that it can run on any vnode(s) once it is eligible.

14.4.5 File Stageout Failure

When stageout encounters an error, there are three retries. PBS waits 1 second and tries again, then waits 11 seconds and tries a third time, then finally waits another 21 seconds and tries a fourth time. PBS sends the job's owner email if the stageout is unsuccessful. For each attempt, if PBS is using scp and that doesn't work, PBS will then try rcp.

14.4.6 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 (on UNIX) or "Full Control" for "Everyone" (on Windows).

See also the "Delivery of Output Files" section of the **PBS Professional User's Guide**.

14.4.7 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.

14.4.8 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 Professional User's Guide**.) If MOM finds there are no processes then she will force the job into the exiting state.

14.4.9 Job Held Due to Invalid Password

If a job fails to run due to an invalid password, then the job will be put on hold (hold type "p"), its comment field updated as to why it failed, and an email sent to user for remedy action. See also the `qhold` and `qrls` commands in the **PBS Professional User's Guide**.

14.4.10 SuSE 9.1 with mpirun and ssh

Use "`ssh -n`" instead of "`ssh`".

14.4.11 Jobs that Can Never Run

If backfilling is being used, the scheduler looks at the job being backfilled around and determines whether that job can never run.

If backfilling is turned on, the scheduler determines whether that job can or cannot run now, and if it can't run now, whether it can ever run. If the job can never run, the scheduler logs a message saying so.

The scheduler only considers the job being backfilled around. That is the only job for which it will log a message saying the job can never run.

This means that a job that can never run will sit in the queue until it becomes the most deserving job. Whenever this job is considered for having small jobs backfilled around it, the error message "resource request is impossible to solve: job will never run" is printed in the scheduler's log file. If backfilling is off, this message will not appear.

If backfilling is turned off, the scheduler determines only whether that job can or cannot run now. The scheduler won't determine if a job will ever run or not.

14.5 Common Errors on Windows

This section discusses errors often encountered under Windows.

14.5.1 Windows: Services Don't Start

In the case where the PBS daemons, the Active Directory database, and the domain controller are all on the same host, some PBS services may not start up immediately. If the Active Directory services are not running when the PBS daemons are started, the daemons

won't be able to talk to the domain controller. This can prevent the PBS daemons from starting. As a workaround, wait until the host is completely up, then retry starting the failing service.
Example: net start pbs_server

14.5.2 MOMs Won't Start

In a domained environment, if the pbsadmin account is a member of any group besides "Domain Users", the install program will fail to add pbsadmin to the local Administrators group on the install host. Make sure that pbsadmin is a member of only one group, "Domain Users" in a domained environment.

14.5.3 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. Use the following command to verify DNS reverse lookup is working

```
pbs_hostn -v hostname
```

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.

14.5.4 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 in all hostnames that resolve to your machine. See also section 11.7.5 "User Authorization" on page 426.

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.

14.5.5 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 resolving to a bad IP address. Perhaps, in `%WINDIR%\system32\drivers\etc\hosts`, localhost and *hostname* were mapped to 127.0.0.1.

14.5.6 Windows: Server Reports Error 10054

If the Server reports error 10054 `rp_request()`, this indicates that another process, probably `pbs_sched`, `pbs_mom`, or `pbs_send_job` is hung up causing the Server to report bad connections. If you desire to kill these services, then use Task Manager to find the Service's process id, and then issue the command:

```
pbskill process-id
```

14.5.7 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
```

14.5.8 Windows: Errors When Not Using Drive C:

If PBS is installed on a hard drive other than `C:`, it may not be able to locate the `pbs.conf` global configuration file. If this is the case, PBS will report the following message:

```
E:\Program Files\PBS Pro\exec\bin>qstat -
pbsconf error: pbs conf variables not found:
PBS_HOME PBS_EXEC
No such file or directory
```

```
qstat: cannot connect to server UNKNOWN (errno=0)
```

To correct this problem, set PBS_CONF_FILE to point pbs.conf to the right path. Normally, during PBS Windows installation, this would be set in system autoexec.bat which will be read after the Windows system has been restarted. Thus, after PBS Windows installation completes, be sure to reboot the Windows system in order for this variable to be read correctly.

14.5.9 Windows: Vnode Comment "ping: no stream"

If a vnode shows a "down" status in xpbsmon or "pbsnodes -a" and contains a vnode comment with the text "ping: no stream" and "write err", then attempt to restart the Server as follows to clear the error:

```
net stop pbs_server
net start pbs_server
```

14.5.10 Windows: Services Debugging Enabled

The PBS services, pbs_server, pbs_mom, pbs_sched, and pbs_rshd are compiled with debugging information enabled. Therefore you can use a debugging tool (such as Dr. Watson) to capture a crash dump log which will aid the developers in troubleshooting the problem. To configure and run Dr. Watson, execute drwtsn32 on the Windows command line, set its "Log Path" appropriately and click on the button that enables a popup window when Dr. Watson encounters an error. Then run a test that will cause one of the PBS services to crash and email to PBS support the generated output in Log_Path. Other debugging tools may be used as well.

14.6 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 Professional website, which has a number of ways to assist you in resolving problems with PBS, such as the Tips & Advice page.

The PBS Professional 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 Professional 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.)

14.7 Troubleshooting PBS Licenses

14.7.1 Unable to Connect to License Server

If PBS cannot contact the license server, the server will log a message:

"Unable to connect to license server at pbs_license_file_location=<X>"

If the license file location is incorrectly initialized (e.g. if the host name or port number is incorrect), PBS may not be able to pinpoint the misconfiguration as the cause of the failure to reach a license server.

If PBS cannot detect a license server host and port when it starts up, the server logs an error message:

"Did not find a license server host and port (pbs_license_file_location=<X>). No external license server will be contacted"

14.7.2 Unable to Run Job; Unable to Obtain Licenses

If the PBS scheduler cannot obtain the licenses to run or resume a job, the scheduler will log a message:

“Could not run job <job>; unable to obtain <N> CPU licenses. avail licenses=<Y>”
“Could not resume <job>; unable to obtain <N> CPU licenses. avail licenses=<Y>”

14.7.3 Reservation Job Fails to Run

A reservation job may not be able to run due to a shortage of licenses. The scheduler will log a message similar to the following:

"Could not run job <job>; unable to obtain <N> cpu licenses. avail_licenses=<Y>”

If the value of the `pbs_license_min` attribute is less than the number of CPUs in the PBS complex when a reservation is being confirmed, the server will log a warning:

“WARNING: reservation <resid> confirmed, but if reservation starts now, its jobs are not guaranteed to run as pbs_license_min=<X> < <Y> (# of CPUs in the complex)“

14.7.4 New Jobs Not Running

If PBS loses contact with the Altair License Server, any jobs currently running will not be interrupted or killed. The PBS server will continually attempt to reconnect to the license server, and re-license the assigned vnodes once the contact to the license server is restored.

No new jobs will run if PBS server loses contact with the License server.

14.7.5 Insufficient Minimum Licenses

If the PBS server cannot get the number of licenses specified in `pbs_license_min` from the FLEX server, the server will log a message:

"checked-out only <X> CPU licenses instead of pbs_license_min=<Y> from license server at host <H>, port <P>. Will try to get more later."

14.7.6 Wrong Type of License

If the PBS server encounters a proprietary license key that is of not type “T”, then the server will log the following message:

“license key #1 is invalid: invalid type or version".

14.7.7 User Error Messages

If a user's job could not be run due to unavailable licenses, the job will get a comment: "Could not run job <job>; unable to obtain <N> CPU licenses. avail_licenses=<Y>"

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

Error Name	Error Code	Description
PBSE_INTERNAL	15011	Internal Server error occurred
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	Can’t enable queue, lacking definition
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

Error Name	Error Code	Description
PBSE_EXCQRESC	15036	Job exceeds Queue resource limits
PBSE_QUENODFLT	15037	No Default Queue Defined
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

Error Name	Error Code	Description
PBSE_SISCOMM	15059	Sister could not communicate
PBSE_SVRDOWN	15060	Request rejected -server shutting down
PBSE_CKPSHORT	15061	Not all tasks could checkpoint
PBSE_UNKNODE	15062	Named vnode is not in the list
PBSE_UNKNODEATR	15063	Vnode attribute not recognized
PBSE_NONODES	15064	Server has no vnode list
PBSE_NODENBIG	15065	Node name is too big
PBSE_NODEEXIST	15066	Node name already exists
PBSE_BADNDATVAL	15067	Bad vnode attribute value
PBSE_MUTUALEX	15068	State values are mutually exclusive
PBSE_GMODERR	15069	Error(s) during global mod of vnodes
PBSE_NORELYMOM	15070	Could not contact MOM
PBSE_RESV_NO_WALLTIME	15075	Job reservation 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)

Error Name	Error Code	Description
PBSE_RESVMSG	15087	So reply_text can return a msg
PBSE_NOTRESV	15088	Not a reservation
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
PBSE_SSIGNON_UNSET_REJECT	15101	rejected if SVR_ssignon_enable not set
PBSE_SSIGNON_SET_REJECT	15102	rejected if SVR_ssignon_enable set
PBSE_SSIGNON_BAD_TRANSITION1	15103	bad attempt: true to false
PBSE_SSIGNON_BAD_TRANSITION2	15104	bad attempt: false to true
PBSE_SSIGNON_NOCONNECT_DEST	15105	couldn't connect to destination host during a user migration request
PBSE_SSIGNON_NO_PASSWORD	15106	no per-user/per-server password
Resource monitor specific error codes		
PBSE_RMUNKNOWN	15201	Resource unknown

Error Name	Error Code	Description
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

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
73	PBS_BATCH_UserCred
74	PBS_BATCH_UserMigrate

Appendix B: Request Codes

Appendix C: File Listing

The following table lists all the PBS files and directories; owner and permissions are specific to UNIX systems.

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-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

Directory / File	Owner	Permission	Average Size
PBS_HOME/server_priv/jobs	root	drwxr-x---	4096
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

Directory / File	Owner	Permission	Average Size
PBS_HOME/sched_priv/sched.lock	root	-rw-r--r--	4
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_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/au-nodeupdate.pl	root	-rw-r--r--	
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	-r--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

Directory / File	Owner	Permission	Average Size
PBS_EXEC/include/pbs_ifl.h	root	-r--r--r--	17424
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/MPI	root	drwxr-xr-x	4096
PBS_EXEC/lib/MPI/pbsrun.bgl.init.in	root	-rw-r--r--	11240
PBS_EXEC/lib/MPI/pbsrun.ch_gm.init.in	root	-rw-r--r--	9924
PBS_EXEC/lib/MPI/pbsrun.ch_mx.init.in	root	-rw-r--r--	9731
PBS_EXEC/lib/MPI/pbsrun.gm_mpd.init.in	root	-rw-r--r--	10767
PBS_EXEC/lib/MPI/pbsrun.intelmpi.init.in	root	-rw-r--r--	10634
PBS_EXEC/lib/MPI/pbsrun.mpich2.init.in	root	-rw-r--r--	10694
PBS_EXEC/lib/MPI/pbsrun.mx_mpd.init.in	root	-rw-r--r--	10770
PBS_EXEC/lib/MPI/sgiMPI.awk	root	-rw-r--r--	6564
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

Directory / File	Owner	Permission	Average Size
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
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

Directory / File	Owner	Permission	Average Size
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
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

Directory / File	Owner	Permission	Average Size
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
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

Directory / File	Owner	Permission	Average Size
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
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

Directory / File	Owner	Permission	Average Size
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
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/maxi-mize.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

Directory / File	Owner	Permission	Average Size
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
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

Directory / File	Owner	Permission	Average Size
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
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

Directory / File	Owner	Permission	Average Size
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_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
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

Directory / File	Owner	Permission	Average Size
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.7B	root	-rw-r--r--	22124
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/mpiexec.8B	root	-rw-r--r--	4701
PBS_EXEC/man/man8/pbs-report.8B	root	-rw-r--r--	19221
PBS_EXEC/man/man8/pbs_attach.8B	root	-rw-r--r--	3790
PBS_EXEC/man/man8/pbs_hostn.8B	root	-rw-r--r--	2781
PBS_EXEC/man/man8/pbs_idled.8B	root	-rw-r--r--	2628
PBS_EXEC/man/man8/pbs_lamboot.8B	root	-rw-r--r--	2739
PBS_EXEC/man/man8/pbs_migrate_users.8B	root	-rw-r--r--	2519
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_mpihp.8B	root	-rw-r--r--	4120
PBS_EXEC/man/man8/pbs_mpilam.8B	root	-rw-r--r--	2647
PBS_EXEC/man/man8/pbs_mpirun.8B	root	-rw-r--r--	3130
PBS_EXEC/man/man8/pbs_password.8B	root	-rw-r--r--	3382
PBS_EXEC/man/man8/pbs_poe.8B	root	-rw-r--r--	3973
PBS_EXEC/man/man8/pbs_probe.8B	root	-rw-r--r--	3344
PBS_EXEC/man/man8/pbs_sched_cc.8B	root	-rw-r--r--	6731
PBS_EXEC/man/man8/pbs_server.8B	root	-rw-r--r--	7914
PBS_EXEC/man/man8/pbs_tclsh.8B	root	-rw-r--r--	2475

Directory / File	Owner	Permission	Average Size
PBS_EXEC/man/man8/pbs_tmrsh.8B	root	-rw-r--r--	3556
PBS_EXEC/man/man8/pbs_wish.8B	root	-rw-r--r--	2123
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/pbsrun.8B	root	-rw-r--r--	20937
PBS_EXEC/man/man8/pbsrun_unwrap.8B	root	-rw-r--r--	2554
PBS_EXEC/man/man8/pbsrun_wrap.8B	root	-rw-r--r--	3855
PBS_EXEC/man/man8/printjob.8B	root	-rw-r--r--	2823
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/man/man8/tracejob.8B	root	-rw-r--r--	4664
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

Directory / File	Owner	Permission	Average Size
PBS_EXEC/sbin/pbs_mom.standard	root	-rwx------	0
PBS_EXEC/sbin/pbs_probe	root	-rwsr-xr-x	83108
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
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

Directory / File	Owner	Permission	Average Size
PBS_EXEC/tcltk/license.terms	root	-rw-r--r--	2233

Appendix C: File Listing

Appendix D: Log Messages

The server, scheduler and MOM all write messages to their log files. Which messages are written depends upon each daemon's event mask. See section 11.17.1, "PBS Events", on page 480, section 11.17.2, "Event Logfiles", on page 482. and section 11.17.3, "Event Logfile Format", on page 483.

A few log messages are listed here.

RPP Retries

Table 34: RPP Retries

	RPP Retries
Logs	Server, scheduler, MOM
Level	0002; DEBUG
Form	date; time;event type; reporting daemon; event class; rpp_stats; total (pkts=<packets>, retries=<retries>, fails=<fails>) last <number of seconds> secs (pkts=<packets>,retries=<retries>, fails=<fails>)
Example	03/22/2006 15:20:44;0002; pbs_mom; Svr;rpp_stats; total (pkts=4321, retries=25, fails=3) last 3621 secs (pkts=43, retries=2, fails=0)

Table 34: RPP Retries

	RPP Retries
Explanation	RPP packet retries, reported both for total number since daemon start ("total") and since last log message ("last <seconds> secs"). Logged at most once per hour unless this hour's retry count is 0. The number of seconds since the previous log message is shown in "last <seconds> secs". pkts: number of RPP packets sent. In "total" group, this is since daemon start (in example, 4321). In "last" group, this is since previous log message (in example, 43). retries: number of RPP data packet retries. In "total" group, this is since daemon start (in example, 25). In "last" group, this is since previous log message (in example, 2). fails: number of failures reported to the caller of the RPP function. In "total" group, this is since daemon start (in example, 3). In "last" group, this is since previous log message (in example, 0). No log message if the number of fails and retries are zero.

cput and mem Logged by Mother Superior

Table 35: cput and mem Logged by Mother Superior

	cput and mem
Logs	Mother Superior
Level	0100
Form	Date; Time; event class; reporting daemon; Job; Job ID; Hostname; cput; mem
Example	07/02/2007 19:47:14;0100;pbs_mom;Job;40.pepsi;pepsi cput= 0:00:00 mem=4756kb
Explanation	On job exit, Mother superior logs the amount of cput and mem used by this job on each node.

MOM Adds $clienthost Address

Table 36: MOM Adds $clienthost Address

	$clienthost Address
Logs	MOM
Level	Event level 0x2, PBSE_SYSTEM, event class Server
Form	Adding IP address XXX.XXX.XXX.XXX as authorized
Example	Adding IP address 127.0.0.1 as authorized
Explanation	When MOM starts up, she logs the addresses associated with a host listed in Mom's config file in $clienthost statements. When MOM receives the list from the Server, addresses associated with other MOMs in the PBS complex will be listed. This occurs as soon as MOM and the Server establish communication and again whenever a node goes down and comes back up, or there is a change to the list of execution hosts (node added to or deleted from the complex). That event and the associated logging may occur at any time.

Scheduler: Job is Invalid

Table 37: Scheduler: Job is Invalid

	Invalid Job
Logs	Scheduler
Level	DEBUG, which is in the default set of levels
Form	Job is invalid - ignoring for this cycle
Example	Job is invalid - ignoring for this cycle
Explanation	Job failed a validity check such as 1) no egroup, euser, select, place, 2) in peer scheduling, pulling server is not a manager for furnishing server, 3) internal scheduler memory failure

Scheduler: Can’t find subjob in simulated universe

Table 38: Scheduler: Can’t find new subjob in simulated universe

	Scheduler Simulation
Logs	Scheduler
Level	DEBUG
Form	can't find new subjob in simulated universe
Example	can't find new subjob in simulated universe
Explanation	This means that when backfilling around a job array, we can run into an error case. The error case we're handling here is that we have simulated the future in a simulated universe. In the simulated universe, we've spawned and run a subjob. Now we're trying to find it so we can do the same thing in the real universe. The simulated subjob can't be found.

Scheduler: Message Indicating Whether It Is Prime Time

Table 39: Scheduler: Message Indicating Whether It Is Prime Time

	Is It Prime Time Now?
Logs	Scheduler
Level	DEBUG2 (256)
Form	“It is *P*. It will end in XX seconds at MM/DD/YYYY HH:MM:SS”
Example	“It is prime time. It will end in 29 seconds at 03/10/2007 09:29:31”
Explanation	The scheduler is declaring whether the current time is prime time or non-prime time. The scheduler is stating when this period of prime time or non-prime time will end.

Jobs that can never run

Table 40: Jobs that can never run

Logs	Scheduler
Level	DEBUG
Form	"resource request is impossible to solve: job will never run"
Example	"resource request is impossible to solve: job will never run"
Explanation	The "most deserving" job can never run. Only printed when backfilling is on.

Appendix D: Log Messages

Appendix E: License Agreement

Altair Engineering, Inc.
Software License Agreement

This License Agreement is a legal agreement between Altair Engineering, Inc. ("Altair") and you ("Licensee") governing the terms of use of the Altair Software. Before you may download or use the Software, your consent to the following terms and conditions is required by clicking on the 'I Accept" button. If you do not have the authority to bind your organization to these terms and conditions, you must click on the button that states "I do not accept" and then have an authorized party in your organization consent to these terms. In the event that your organization and Altair have a master software license agreement, mutually agreed upon in writing, in place at the time of your execution of this agreement, the terms of the master agreement shall govern.

1. **DEFINITIONS**. In addition to terms defined elsewhere in this Agreement, the following terms shall have the meanings defined below for purposes of this Agreement:

Documentation. Documentation provided by Altair on any media for use with the Software.

Execute. To load Software into a computer's RAM or other primary memory for execution by the computer.

Global Zone: Software is licensed based on three Global Zones: the Americas, Europe and Asia-Pacific. When Licensee has Licensed Workstations located in multiple Global Zones, which are connected to a single License (Network) Server, a premium is applied to the standard Software License pricing for a single Global Zone.

License Log File. A computer file providing usage information on the Software as gathered by the Software.

License Management System. The license management system that accompanies the Software and limits its use in accordance with the usage permitted under this Agreement, and which includes a License Log File.

L**icense (Network) Server**. A network file server that Licensee owns or leases located on Licensee's premises and identified by machine serial number on the Order Form.

License Units. A parameter used by the License Management System to determine the

usage of the Software permitted under this Agreement at any one time.

Licensed Workstations. Single-user computers located in the same Global Zone(s) that Licensee owns or leases that are connected to the License (Network) Server via local area network or Licensee's private wide-area network.

Maintenance Release. Any release of the Software made generally available by Altair to its Licensees with annual leases, or those with perpetual licenses who have an active maintenance agreement in effect, that corrects programming errors or makes other minor changes to the Software. The fees for maintenance and support services are included in the annual license fee but perpetual licenses require a separate fee.

Order Form. Altair's standard form in either hard copy or electronic format that contains the specific parameters (such as identifying Licensee's contracting office, License Fees, Software, Support, and License (Network) Servers) of the transaction governed by this Agreement.

Proprietary Rights Notices. Patent, copyright, trademark or other proprietary rights notices applied to the Software, Documentation or the packaging or media of same.

Software. The software identified in the Order Form and any Updates or Maintenance Releases.

Suppliers. Any person, corporation or other legal entity which may provide software or documents which are included in the Software.

Support. The maintenance and support services provided by Altair pursuant to this Agreement.

Templates. Human readable ASCII files containing machine-interpretable commands for use with the Software.

Term. The initial term of this Agreement or any renewal term. Annual licenses shall have a 12-month term of use. Paid-up, or perpetual licenses, shall have a term of twenty-five years.

Update. A new version of the Software made generally available by Altair to its Licensee that includes additional features or functionalities but is substantially the same computer code as the existing Software.

2. **PAYMENT**. Licensee shall pay in full the fee for licensed Software and Support within thirty (30) days of receipt of the invoice. Past due fees shall bear interest at the maximum legal rate. Altair may condition its delivery of any Maintenance Release or Update to Licensee on Licensee's having paid all amounts then owed to Altair. Fees do not include taxes or duties and Licensee is responsible for paying (or for reimbursing Altair if Altair is required to pay) any federal, state or local taxes, or duties imposed on this License or the possession or use by Licensee of the Software excluding, however, all taxes on or measured by Altair's net income. Altair shall be entitled to its reasonable costs of collection (including attorneys fees and interest) if license fees are not paid to it on a timely basis.

3. **TERM**. Unless terminated earlier in accordance with the provisions of this Agreement, this Agreement will be in force for a period as stated on the Order Form. For

annual licenses or Support provided for perpetual licenses, renewal shall be automatic for a successive year ("Renewal Term"), upon mutual written execution of a new Order Form. All charges and fees for each Renewal Term shall be set forth in the Order Form executed for each Renewal Term. All Software procured by Licensee may be made coterminous at the request of Licensee and the consent of Altair.

4. **LICENSE GRANT**. Subject to the terms and conditions set forth in this Agreement, Altair hereby grants Licensee, and Licensee hereby accepts, a limited, non-exclusive, non-transferable license to: a) install the Software on the License (Network) Server(s) identified on the Order Form for use only at the sites identified on the Order Form; b) execute the Software on Licensed Workstations in accordance with the License Management System for use solely by Licensee's employees or its onsite Contractors who have agreed to be bound by the terms of this Agreement, for Licensee's internal business use on Licensed Workstations within the Global Zone(s) as identified on the Order Form and for the term identified on the Order Form; c) make backup copies of the Software, provided that Altair's Proprietary Rights Notices are reproduced on each such backup copy; d) freely modify and use Templates, provided that such modifications shall not be subject to Altair's warranties, indemnities, support or other Altair obligations under this Agreement; and e) copy and distribute Documentation inside Licensee's organization exclusively for use by Licensee's employees. A copy of the License Log File shall be made available to Altair automatically on no less than a monthly basis. In the event that Licensee uses a third party vendor to provide itself with information technology (IT) support, the IT company shall be permitted to access the Software only upon its agreement to abide by the terms of this Agreement. Licensee shall indemnify, defend and hold harmless Altair for the actions of its IT vendor(s).

5. **RESTRICTIONS ON USE**. Notwithstanding the foregoing license grant, Licensee shall not do (or allow others to do) any of the following: a) install, use, copy, modify, merge, or transfer copies of the Software or Documentation, except as expressly authorized in this Agreement; b) use any back-up copies of the Software for any purpose other than to replace the original copy provided by Altair in the event it is destroyed or damaged; c) disassemble, decompile or "unlock", reverse translate, reverse engineer, or in any manner decode the Software for any reason; d) sublicense, sell, lend, assign, rent, distribute, publicly display or publicly perform the Software or Documentation or Licensee's rights under this Agreement; e) allow use outside the Global Zone(s) or User Sites identified on the Order Form; f) allow third parties to access or use the Software, such as through a service bureau, wide area network, Internet location or time-sharing arrangement except as expressly provided in Section 4(b); g) remove any Proprietary Rights Notices from the Software; h) disable or circumvent the License Management System provided with the Software; or (i) develop, test or support software of Licensee or third parties.

6. **OWNERSHIP AND CONFIDENTIALITY**. Licensee acknowledges that all

applicable rights in patents, copyrights, trademarks, service marks, and trade secrets embodied in the Software and Documentation are owned by Altair and/or its Suppliers. Licensee further acknowledges that the Software and Documentation, and all copies thereof, are and shall remain the sole and exclusive property of Altair and/or its Suppliers. This Agreement is a license and not a sale of the Software. Altair retains all rights in the Software and Documentation not expressly granted to Licensee herein. Licensee acknowledges that the Software and accompanying Documentation are confidential and constitute valuable assets and trade secrets of Altair and/or its Suppliers. Licensee agrees to take the precautions necessary to protect and maintain the confidentiality of the Software and Documentation, and shall not disclose or make them available to any person or entity except as expressly provided in this Agreement. Licensee shall promptly notify Altair in the event any unauthorized person obtains access to the Software. If Licensee is required by any governmental authority or court of law to disclose Altair's confidential information, then Licensee shall immediately notify Altair before making such disclosure so that Altair may seek a protective order or other appropriate relief. Licensee's obligations set forth in Section 5 and Section 6 of this Agreement shall survive termination of this Agreement for any reason. Altair's Suppliers, as third party beneficiaries, shall be entitled to enforce the terms of this Agreement directly against Licensee as necessary to protect Supplier's intellectual property or other rights. Altair shall keep confidential all Licensee information provided to Altair in order that Altair may provide Support to Licensee shall be kept confidential and used only for the purpose of assisting Licensee in its use of the licensed Software.

7. **MAINTENANCE AND SUPPORT**. Maintenance. Altair will provide Licensee at no additional charge for annual licenses, and for a fee for paid-up licenses, with any Maintenance Releases and Updates of the Software or Documentation that are generally released by Altair during the term of this Agreement, except that this shall not apply to any Renewal Term for which full payment has not been received. Altair does not promise that there will be a certain number of Updates (or any Updates) during a particular year. If there is any question or dispute as to whether a particular release is a Maintenance Release, an Update or a new product, the categorization of the release as determined by Altair shall be final. Licensee must install Maintenance Releases and Updates promptly after receipt from Altair. Maintenance Releases and Updates are Software subject to this Agreement. Altair shall only be obligated to provide support and maintenance for the most current release of the Software and its most recent prior release Support. Altair will provide support via telephone and email to Licensee at the fees, if any, as listed on the Order Form.. If Support has not been procured for any period of time for paid-up licenses, a reinstatement fee shall apply. Support consists of responses to questions from Licensee's personnel related to the use of the then-current and most recent prior release version of the Software. Licensee agrees to provide Altair will sufficient information to resolve technical issues as may be reasonably requested by Altair. Licensee agrees to the best of its abilities to read, comprehend and follow operating instructions and procedures as specified in,

but not limited to, Altair's Documentation and other correspondence related to the Software, and to follow procedures and recommendations provided by Altair in an effort to correct problems. Licensee also agrees to notify Altair of a programming error, malfunction and other problems in accordance with Altair's then current problem reporting procedure. If Altair believes that a problem reported by Licensee may not be due to an error in the Software, Altair will so notify Licensee. Questions must be directed to Altair's specially designated telephone support numbers and email addresses. Support will also be available via email at Internet addresses designated by Altair. Support is available Monday through Friday (excluding holidays) from 8:00 a.m. to 5:00 p.m local time in the Global Zone where licensed. Exclusions. Altair shall have no obligation to maintain or support (a) altered, damaged or Licensee-modified Software, or any portion of the Software incorporated with or into other software; (b) any version of the Software other than the current version of the Software or the immediately previous version; (c) Software problems causes by Licensee's negligence, abuse or misapplication of Software other than as specified in the Documentation, or other causes beyond the reasonable control of Altair; or (d) Software installed on any hardware, operating system version or network environment that is not supported by Altair. Support also excludes configuration of hardware, non Altair Software, and networking services; consulting services; general solution provider related services; and general computer system maintenance.

8. **WARRANTY AND DISCLAIMER**. Altair warrants for a period of ninety (90) days after Licensee initially receives the Software that the Software will perform under normal use substantially as described in then current Documentation and this Agreement. Supplier software included in the Software and provided to Licensee shall be warranted as stated by the Supplier. Copies of the Suppliers' terms and conditions for software are available on the Altair Support website. Support services shall be provided in a workmanlike and professional manner, in accordance with the prevailing standard of care for consulting support engineers at the time and place the services are performed.

ALTAIR DOES NOT REPRESENT OR WARRANT THAT THE SOFTWARE WILL MEET LICENSEE'S REQUIREMENTS OR THAT ITS OPERATION WILL BE UNINTERRUPTED OR ERROR-FREE, OR THAT IT WILL BE COMPATIBLE WITH ANY PARTICULAR HARDWARE OR SOFTWARE. ALTAIR EXCLUDES AND DIS**CLAIMS ALL EXPRESS AND IMPLIED WARRANTIES NOT STATED HEREIN, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT. THE ENTIRE RISK FOR THE PERFORMANCE, NON-PERFORMANCE OR RESULTS OBTAINED FROM USE OF THE SOFTWARE RESTS WITH LICENSEE AND NOT ALTAIR. ALTAIR MAKES NO WARRANTIES WITH RESPECT TO THE ACCURACY, COMPLETENESS, FUNCTIONALITY, SAFETY, PERFORMANCE, OR ANY OTHER ASPECT OF ANY DESIGN, PROTOTYPE OR FINAL PRODUCT DEVELOPED BY LICENSEE USING THE**

SOFTWARE.

9. INDEMNITY. Altair will defend, at its expense, any claim made against Licensee based on an allegation that the Software infringes a patent or copyright ("Claim"); provided, however, that this indemnification does not include claims based on Supplier software, and that Licensee has not materially breached the terms of this Agreement, Licensee notifies Altair in writing within ten (10) days after Licensee first learns of the Claim; and Licensee cooperates fully in the defense of the claim. Altair shall have sole control over such defense; provided, however, that it may not enter into any settlement license binding upon Licensee without Licensee's consent, which shall not be unreasonably withheld. If a Claim is made, Altair may modify the Software to avoid the alleged infringement, provided, however, that such modifications do not materially diminish the Software's functionality. If such modifications are not commercially reasonably or technically possible, Altair may terminate this Agreement and refund to Licensee the prorated license fee that Licensee paid for the then current Term. Perpetual licenses shall be pro-rated over a 36-month term. Altair shall have no obligation under this Section 9, however, if the alleged infringement arises from Altair's compliance with specifications or instructions prescribed by Licensee, modification of the Software by Licensee, use of the Software in combination with other software not provided by Altair and which use is not specifically described in the Documentation and if Licenses is not using the most current version of the Software, if such alleged infringement would not have occurred except for such exclusions listed here. This section 9 states Altair's entire liability to Licensee in the event a Claim is made.

10. LIMITATION OF REMEDIES AND LIABILITY. Licensee's exclusive remedy (and Altair's sole liability) for Software that does not meet the warranty set forth in Section 8 shall be, at Altair's option, either (i) to correct the nonconforming Software within a reasonable time so that it conforms to the warranty; or (ii) to terminate this Agreement and refund to Licensee the license fees that Licensee has paid for the then current Term for the nonconforming Software; provided, however that Licensee notifies Altair of the problem in writing within the applicable Warranty Period when the problem first occurs. Any corrected Software shall be warranted in accordance with Section 8 for ninety (90) days after delivery to Licensee. The warranties hereunder are void if the Software has been misused or improperly installed, or if Licensee has violated the terms of this Agreement.

Altair's entire liability for all claims arising under or related in any way to this Agreement (regardless of legal theory), except as provided in Section 9, shall be limited to direct damages, and shall not exceed, in the aggregate for all claims, the license and maintenance fees paid under this Agreement by Licensee in the 12 months prior to the claim on a prorated basis. ALTAIR AND ITS SUPPLIERS SHALL NOT BE LIABLE TO LICENSEE OR ANYONE ELSE FOR INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING HEREUNDER (INCLUDING LOSS OF PROFITS OR DATA, DEFECTS IN DESIGN OR PRODUCTS CREATED USING THE SOFTWARE, OR ANY INJURY OR DAMAGE RESULTING FROM SUCH DEFECTS, SUFFERED

BY LICENSEE OR ANY THIRD PARTY) EVEN IF ALTAIR OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Licensee acknowledges that it is solely responsible for the adequacy and accuracy of the input of data, including the output generated from such data, and agrees to defend, indemnify, and hold harmless Altair and its Suppliers from any and all claims, including reasonable attorney's fees, resulting from, or in connection with Licensee's use of the Software. No action, regardless of form, arising out of the transactions under this Agreement may be brought by either party against the other more than two (2) years after the cause of action has accrued, except for actions related to unpaid fees.

11. TERMINATION. Either party may terminate this Agreement upon thirty (30) days prior written notice upon the occurrence of a default or material breach by the other party of its obligations under this Agreement (except for a breach by Altair of the warranty set forth in Section 8 for which a remedy is provided under Section 10; or a breach by Licensee of Section 5 or Section 6 for which no cure period is provided and Altair may terminate this Agreement immediately) if such default or breach continues for more than thirty (30) days after receipt of such notice. Upon termination of this Agreement, Licensee must cease using the Software and, at Altair's option, return all copies to Altair, or certify it has destroyed all such copies of the Software and Documentation.

12. UNITED STATES GOVERNMENT RESTRICTED RIGHTS. This section applies to all acquisitions of the Software by or for the United States government. By accepting delivery of the Software, the government hereby agrees that the Software qualifies as "commercial" computer software as that term is used in the acquisition regulations applicable to this procurement and that the government's use and disclosure of the Software is controlled by the terms and conditions of this Agreement to the maximum extent possible. This Agreement supersedes any contrary terms or conditions in any statement of work, contract, or other document that are not required by statute or regulation. If any provision of this Agreement is unacceptable to the government, Vendor may be contacted at Altair Engineering, Inc., 1820 E. Big Beaver Road, Troy, MI 48083-2031; telephone (248) 614-2400. If any provision of this Agreement violates applicable federal law or does not meet the government's actual, minimum needs, the government agrees to return the Software for a full refund.

For procurements governed by DFARS Part 227.72 (OCT 1998), HyperWorks Software is provided with only those rights specified in this Agreement in accordance with the Rights in Commercial Computer Software or Commercial Computer Software Documentation policy at DFARS 227.7202-3(a) (OCT 1998). For procurements other than for the Department of Defense, use, reproduction, or disclosure of the Software is subject to the restrictions set forth in this Agreement and in the Commercial Computer Software - Restricted Rights FAR clause 52.227-19 (June 1987) and any restrictions in successor regulations thereto.

Portions of Altair's PBS Professional Software and Documentation are provided with

RESTRICTED RIGHTS. Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subdivision(c)(1)(ii) of the rights in the Technical Data and Computer Software clause in DFARS 252.227-7013, or in subdivision (c)(1) and (2) of the Commercial Computer Software-Restricted Rights clause at 48 CFR52.227-19, as applicable.

13. CHOICE OF LAW AND VENUE. This Agreement shall be governed by and construed under the laws of the state of Michigan, without regard to that state's conflict of laws principles except if the state of Michigan adopts the Uniform Computer Information Transactions Act drafted by the National Conference of Commissioners of Uniform State Laws as revised or amended as of June 30, 2002 ("UCITA") which is specifically excluded. This Agreement shall not be governed by the United Nations Convention on Contracts for the International Sale of Goods, the application of which is expressly excluded. Each Party waives its right to a jury trial in the event of any dispute arising under or relating to this Agreement. Each party agrees that money damages may not be an adequate remedy for breach of the provisions of this Agreement, and in the event of such breach, the aggrieved party shall be entitled to seek specific performance and/or injunctive relief (without posting a bond or other security) in order to enforce or prevent any violation of this Agreement.

14. GENERAL PROVISIONS. **Export Controls**. Licensee acknowledges that the Software may be subject to the export control laws and regulations of the United States and any amendments thereof. Licensee agrees that Licensee will not directly or indirectly export the Software into any country or use the Software in any manner except in compliance with all applicable U.S. export laws and regulations. **Notice**. All notices given by one party to the other under this Agreement shall be sent by certified mail, return receipt requested, or by overnight courier, to the respective addresses set forth in this Agreement or to such other address either party has specified in writing to the other. All notices shall be deemed given when actually received. **Assignment**. Neither party shall assign this Agreement without the prior written consent of other party, which shall not be unreasonably withheld. All terms and conditions of this Agreement shall be binding upon and inure to the benefit of the parties hereto and their respective successors and permitted assigns. **Waiver**. The failure of a party to enforce at any time any of the provisions of this Agreement shall not be construed to be a waiver of the right of the party thereafter to enforce any such provisions. **Severability**. If any provision of this Agreement is found void and unenforceable, such provision shall be interpreted so as to best accomplish the intent of the parties within the limits of applicable law, and all remaining provisions shall continue to be valid and enforceable. **Headings**. The section headings contained in this Agreement are for convenience only and shall not be of any effect in constructing the meanings of the Sections. **Modification**. No change or modification of this Agreement will be valid unless it is in writing and is signed by a duly authorized representative of each party. **Conflict**. In the event of any conflict between the terms of this Agreement and any terms and conditions on a Purchase Order or comparable document, the terms of this Agreement

shall prevail. Moreover, each party agrees any additional terms on any Purchase Order other than the transaction items of (a) item(s) ordered; (b) pricing; (c) quantity; (d) delivery instructions and (e) invoicing directions, are not binding on the parties. **Entire Agreement**. This Agreement and the Order Form(s) constitute the entire understanding between the parties related to the subject matter hereto, and supersedes all proposals or prior agreements, whether written or oral, and all other communications between the parties with respect to such subject matter. This Agreement may be executed in one or more counterparts, all of which together shall constitute one and the same instrument.

Appendix E: License Agreement

Index

B

C

D

E

F

N

O

P

Q

R

S

T

U

V

W

X