PBS Professional® 12.0

Reference Guide

Altair PBS Professional 12 Reference Guide, updated: 1/25/13

For documentation and the PBS Works forums, go to:
  Web: www.pbsworks.com
For more information, contact Altair at:
  Email: pbssales@altair.com

Technical Support

Location	Telephone	e-mail
North America	+1 248 614 2425	pbssupport@altair.com
China	+86 (0)21 6117 1666	es@altair.com.cn
France	+33 (0)1 4133 0992	francesupport@altair.com
Germany	+49 (0)7031 6208 22	hwsupport@altair.de
India	+91 80 66 29 4500	pbs-support@india.altair.com
Italy	+39 0832 315573 +39 800 905595	support@altairengineering.it
Japan	+81 3 5396 2881	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)1926 468 600	pbssupport@uk.altair.com

Table of Contents

Table of Contents

Table of Contents

Table of Contents

About PBS Documentation

Where to Keep the Documentation

To make cross-references work, put all of the PBS guides in the same directory.

What is PBS Professional?

PBS is a workload management system that provides a unified batch queuing and job management interface to a set of computing resources.

The PBS Professional Documentation

The documentation for PBS Professional includes the following:

PBS Professional Administrator's Guide:

Provides the PBS administrator with the information required to configure and manage PBS Professional (PBS).

PBS Professional Quick Start Guide:

Provides a quick overview of PBS Professional installation and license file generation.

PBS Professional Installation & Upgrade Guide:

Contains information on installing and upgrading PBS Professional.

PBS Professional User's Guide:

Covers user commands and how to submit, monitor, track, delete, and manipulate jobs.

PBS Professional Programmer's Guide:

Discusses the PBS application programming interface (API).

PBS Professional Reference Guide:

Contains PBS reference material.

PBS Manual Pages:

Describe PBS commands, resources, attributes, APIs

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 book.

Document Conventions

PBS documentation uses the following typographic conventions:

<u>abbr</u>eviation

The shortest acceptable abbreviation of a command or subcommand is underlined.

command

Commands such as `qmgr` and `scp`

input

Command-line instructions

manpage(x)

File and path names. Manual page references include the section number in parentheses appended to the manual page name.

formats

Formats

Attributes

Attributes, parameters, objects, variable names, resources, types

Values

Keywords, instances, states, values, labels

Definitions

Terms being defined

```
Output
```

Output or example code

File contents

Chapter 1

Terminology

This chapter describes the terms used in PBS Professional documentation.

Accept an action (Hooks)

A hook *accepts* an action when the hook allows the action to take place.

Access control list, ACL

An *ACL*, or *Access Control List*, is a list of users, groups, or hosts from which users or groups may be attempting to gain access. This list defines who or what is allowed or denied access to parts of PBS such as the Server, queues, or reservations. A Server ACL applies to access to the Server, and therefore all of PBS. A queue's ACL applies only to that particular queue. A reservation's ACL applies only to that particular reservation. See section 9.3.4, "ACLs", on page 643.

Access to a queue

Applies to users, groups, and hosts. Being able to submit jobs to the queue, move jobs into the queue, being able to perform operations on jobs in the queue, and being able to get the status of the queue.

Access to a reservation

Applies to users, groups, and hosts. Being able to place jobs in the reservation, whether by submitting jobs to the reservation or moving jobs into the reservation. It also means being able to delete the reservation, and being able to operate on the jobs in the reservation.

Access to the server

Applies to users, groups, and hosts. Being able to run PBS commands to submit jobs and perform operations on them such as altering, selecting, and querying status. It also means being able to get the status of the Server and queues.

Terminology

Account

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

Action (Hooks)

> A PBS operation or state transition. The actions that hooks can affect are submitting a job, altering a job, running a job, making a reservation, and moving a job to another queue.

Active (Failover)

> A server daemon is active when it is managing user requests and communicating with the scheduler and MOMs.

Active Directory (Windows)

> *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.

Admin (Windows)

> A user logged in from an account that is either:

> **1.** A member of a group having full control over the local computer and the domain controller

> **2.** Allowed to make domain and schema changes to the Active Directory.

Administrator, PBS Administrator

> A person who administers PBS, performing functions such as downloading, installing, upgrading, configuring, or managing PBS. A PBS administrator must have an account with Manager privilege and an account with root privilege. Administrator is distinguished from "site administrator", although often these are the same person.

> The term *PBS Administrator* is used in PBS documentation to mean the above, rather than to indicate a user role recognized by PBS Professional.

Administrators (Windows)

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

Advance reservation

> A reservation for a specific set of resources for a specified start time and duration in the future. Advance reservations are created by users to reserve resources for jobs. The reservation is available only to the creator of the reservation and any users or groups specified by the creator.

AOE, Application operating environment

> The environment on a vnode. This may be one that results from provisioning that vnode, or one that is already in place

API

> PBS provides an *Application Programming Interface,* or *API,* which is used by the commands to communicate with the Server. This API is described in the PBS Professional Programmer's Guide. A site may make use of the API to implement new commands if so desired.

Application Checkpoint

> The application performs its own checkpointing when it receives the appropriate signal etc.

Array job

> See "Job array".

Attribute

> An *attribute* is a data item belonging to an object. The attribute's value affects the behavior of or provides information about the object. A job's owner can set the attributes of a job, and the administrator can set attributes of queues and vnodes.

Backfilling

> A scheduling policy where

> **1.** High-priority jobs are scheduled for execution

> **2.** Lower-priority jobs are run if the following conditions are true:

>> Resources (that cannot be used by the high-priority jobs) are available

>> The lower-priority jobs will not delay the higher-priority jobs

> Lower-priority jobs selected for execution are those next in priority order that will fit in the available resources.

Batch, Batch processing

> Allowing jobs to be run outside of the interactive login environment.

Borrowing vnode

> The vnode where a shared vnode resource is available, but not managed.

Built-in resource

> A resource that is defined in PBS Professional as shipped. Examples of built-in resources are ncpus, which tracks the number of CPUs, and mem, which tracks memory. See section 5.4.1, "Built-in and Custom Resources" on page 287 in the PBS Professional Administrator's Guide.

Terminology

Checkpoint/Restart

> Allows jobs to be checkpointed and restarted. Uses OS-provided or third-party checkpoint/restart facility.

Checkpoint and Abort, checkpoint_abort

> The checkpoint script or tool writes a restart file, then PBS kills and requeues the job. The job resumes from the start file when it is executed again.

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

> A relatively homogeneous set of systems that are used as if they are a single machine.

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.

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. A complex can be a heterogeneous mix of system architectures, and can include one or more clusters.

Consumable resource

> A consumable resource is a resource that is reduced or taken up by being used. Examples of consumable resources are memory or CPUs. See section 5.4.3, "Consumable and Non-consumable Resources" on page 288 in the PBS Professional Administrator's Guide.

CPU

Has two meanings, one from a hardware viewpoint, and one from a software viewpoint:

1. A core. The part of a processor that carries out computational tasks. Some systems present virtual cores, for example in hyperthreading.

2. Resource required to execute a program thread. PBS schedules jobs according, in part, to the number of threads, giving each thread a core on which to execute. The resource used by PBS to track CPUs is called "*ncpus*". The number of CPUs available for use defaults to the number of cores reported by the OS. When a job requests one CPU, it is requesting one core on which to run.

Custom resource

A resource that is not defined in PBS as shipped. Custom resources are created by the PBS administrator or by PBS for some systems. See section 5.4.1, "Built-in and Custom Resources" on page 287 in the PBS Professional Administrator's Guide.

Degraded reservation

An advance reservation for which one or more associated vnodes are unavailable.

A standing reservation for which one or more vnodes associated with any occurrence are unavailable.

Delegation (Windows)

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, delegation allows giving the account read access only to all domain users and groups information. This is done via the Delegation wizard.

Destination, destination queue

A queue where a job is sent. A destination may be at the same PBS server or at another server.

Destination identifier

The string that names a destination queue. It is composed of two parts and has the following format:

queue@server
where

server

> The name of a PBS Server

queue

> The string identifying a queue on that Server.

Directive

A means by which the user specifies to PBS the value of a job submission variable such as number of CPUs, the name of the job, etc. The default start of a directive is "*#PBS*". PBS directives either specify resource requirements or attribute values. See page section 3.11, "Changing the Job's PBS Directive", on page 59 of the PBS Professional User's Guide.

Domain Admin Account (Windows)

A domain account on Windows that is a member of the "Domain Admins" group.

Domain Admins (Windows)

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 User Account (Windows)

A domain account on Windows that is a member of the Domain Users group.

Domain Users (Windows)

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.

Enterprise Admins (Windows)

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.

Entity, PBS entity

A user, group, or host.

Entity share

Setting job execution and/or preemption priority according to how much of the fairshare tree is assigned to each job's owner.

Execution host

A computer which runs PBS jobs. An *execution host* is a system with a single operating system (OS) image, a unified virtual memory space, one or

more CPUs and one or more IP addresses. Systems like Linux clusters, which contain separate computational units each with their own OS, are collections of hosts. Systems such as the SGI ICE are also collections of hosts. The SGI Altix 4700 is a single execution host.

An execution host can be comprised of one or more vnodes. For example, the SGI Altix 4700, while being a single execution host, can contain multiple vnodes, where each vnode is a blade. On the SGI ICE, each blade is treated as a vnode. See "Vnode".

Execution queue

A queue from which a job can be executed.

Failover

The PBS complex can run a backup server. If the primary server fails, the secondary takes over without an interruption in service.

Fairshare

A scheduling policy that prioritizes jobs according to how much of a specified resource is being used by, and has recently been used by, job submitters. Job submitters can be organized into groups and subgroups, so that jobs can also be prioritized according to those groups' resource usage. Users and groups can each be allotted a percentage of total resource usage. See section 4.8.18, "Using Fairshare" on page 165 in the PBS Professional Administrator's Guide.

File staging

File staging is the transfer of files between a specified storage location and the execution host. See "Stage in" and "Stage out".

Finished jobs

Jobs whose execution is done, for any reason:

- Jobs which finished execution successfully and exited

- Jobs terminated by PBS while running

- Jobs whose execution failed because of system or network failure

- Jobs which were deleted before they could start execution

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).

Furnishing queue

In peer scheduling, the queue from which jobs are pulled to be run at another complex

Generic group limit

> A limit that applies separately to groups at the server or a queue. This is the limit for groups which have no individual limit specified. A limit for generic groups is applied to the usage across the entire group. A separate limit can be specified at the server and each queue.

Generic project limit

> Applies separately to projects at the server or a queue. The limit for projects which have no individual limit specified. A limit for generic projects is applied to the usage across the entire project. A separate limit can be specified at the server and each queue.

Generic user limit

> A limit that applies separately to users at the server or a queue. This is the limit for users who have no individual limit specified. A separate limit for generic users can be specified at the server and at each queue.

Global resource

> A global resource is defined in a resources_available attribute, at the server, a queue, or a host. Global resources can be operated on via the qmgr command and are visible via the qstat and pbsnodes commands. See section 5.4.5, "Global and Local Resources" on page 289 in the PBS Professional Administrator's Guide.

Group

> A collection of system users. A user must be a member of at least one group, and can be a member of more than one group.

Group access, Access by group

> Refers to access to PBS objects, such as the server, queues, and reservations. A user in the specified group is allowed access at the Server, queues, and reservations

Group ID (GID)

> Unique numeric identifier assigned to each group. See "Group".

Group limit

> Refers to configurable limits on resources and jobs. This is a limit applied to the total used by a group, whether the limit is a generic group limit or an individual group limit.

History jobs

> Jobs which will no longer execute at this server:
>
> • Moved jobs
>
> • Finished jobs

Hold

A restriction which prevents a job from being executed. When a job has a hold applied to it, it is in the *Held* (*H*) state. See section 2.46, "qhold", on page 149.

Hook

Hooks are custom executables that can be run at specific points in the execution of PBS. They accept, reject, or modify the upcoming action. This provides job filtering, patches or workarounds, and extends the capabilities of PBS, without the need to modify source code.

Host

A machine running an operating system. A host can be made up of one or more vnodes. All vnodes of a host share the same value for resources_available.host.

Host access, Access by host

Refers to user access at the Server, queues, and reservations from the specified host

HPC Basic Profile (HPCBP)

Proposed standard web services specification for basic job execution capabilities defined by the OGSA High Performance Computing Profile Working Group.

HPC Basic Profile Job, HPCBP Job

Generic job that can run either on vnodes managed by PBS or on nodes managed by HPC Basic Profile Server.

HPC Basic Profile Server

Service that executes jobs from any HPC Basic Profile compliant client

HPCBP MOM

MOM that sends jobs for execution to an HPC Basic Profile Server. This MOM is a client-side implementation of the HPC Basic Profile Specification, and acts as a proxy for and interface to an HPC Basic Profile compliant server.

Idle

A server daemon is idle when it is running, but only accepting handshake messages, not performing workload management.

Indirect resource

A shared vnode resource at vnode(s) where the resource is not defined, but which share the resource.

Individual group limit

Applies separately to groups at the server or a queue. This is the limit for a group which has its own individual limit specified. An individual group limit overrides the generic group limit, but only in the same context, for example, at a particular queue. The limit is applied to the usage across the entire group. A separate limit can be specified at the server and each queue.

Individual project limit

Applies separately to projects at the server or a queue. Limit for a project which has its own individual limit specified. An individual project limit overrides the generic project limit, but only in the same context, for example, at a particular queue. The limit is applied to the usage across the entire project. A separate limit can be specified at the server and each queue.

Individual user limit

Applies separately to users at the server or a queue. This is the limit for users who have their own individual limit specified. A limit for an individual user overrides the generic user limit, but only in the same context, for example, at a particular queue. A separate limit can be specified at the server and each queue.

Installation account

The account used by the administrator when installing PBS. Not the *pbsadmin* account used by PBS.

Interactive job

A job where standard input and output are connected to the terminal from which the job was submitted.

Job or Batch job

A unit of work managed by PBS. A *job* is a related set of tasks, created and submitted by the user. The user specifies the resources required by the job, and the processes that make up the job. When the user submits a job to PBS, the user is handing off these tasks to PBS to manage. PBS then schedules the job to be run, and manages the running of the job, treating the tasks as parts of a whole. A job is usually composed of a set of directives and a shell script.

Job array

A *job array* is a container for a collection of similar jobs submitted under a single job ID. It can be submitted, queried, modified and displayed as a unit. The jobs in the collection are called subjobs. For more on job arrays, see section , "Job Arrays", on page 235 of the PBS Professional User's Guide.

Job array identifier

> The identifier returned upon success when submitting a job array. The format is
>
> *sequence_number[]*
>
> or
>
> *sequence_number[].server.domain.com.*
>
> Note that some shells require you to enclose a job array identifier in double quotes.

Job array range

> A specification for a set of subjobs within a job array. When specifying a range, indices used must be valid members of the job array's indices. Format:
>
> *sequence_number[<first>-<last>:<step>][.server][@new server]*
>
> *first* is the first index of the subjobs.
>
> *last* is the last index of the subjobs.
>
> *step* is the stepping factor.

Job ID, Job identifier

> When a job is successfully submitted to PBS, PBS returns a unique identifier for the job. Format:
>
> *sequence_number[.server][@new server]*

Job state

> A job exists in one of the possible states throughout its existence within the PBS system. For example, a job can be queued, running, or exiting. See "States" on page 411.

Job Submission Description Language (JSDL)

> Language for describing the resource requirements of jobs.

Job-wide resource, Server resource

> A server resource, also called a server-level or job-wide resource, is a resource that is available at the server. A server resource is available to be consumed or matched at the server if you set the server's resources_available.<resource name> attribute to the available or matching value. For example, you can define a custom resource called *FloatingLicenses* and set the server's resources_available.FloatingLicenses attribute to the number of available floating licenses.
>
> A server resource is a job-wide resource. This means that a job can request this resource for the entire job, but not for individual chunks.

An example of a job-wide resource is shared scratch space, or any custom resource that is defined at the server and queue level.

Job-wide resource request

A job-wide resource request is for resource(s) at the server or queue level. This resource must be a server-level or queue-level resource. A job-wide resource is designed to be used by the entire job, and is available to the complex, not just one execution host. Job-wide resources are requested outside of a selection statement, in this form:

-l keyword=value[,keyword=value ...]

where *keyword* identifies either a consumable resource or a time-based resource such as walltime.

A resource request "outside of a selection statement" means that the resource request comes after "`-l`", but not after "`-lselect=`".

Kill a job

To terminate the execution of a job.

License Manager Daemon (`lmx-serv-altair`)

Daemon that functions as the license server.

License server

Manages licenses for PBS jobs.

License Server List Configuration

One form of redundant license server configuration. A collection of license server files, or "<port>@<host>" settings, pointing to license servers managing Altair licenses. Each server on the list is tried in turn. 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. The first running server is the only server queried.

Limit

A maximum that can be applied in various situations:

• The maximum amount of a resource that can be consumed at any time by running jobs

• The maximum amount of a resource that can be allocated to queued jobs,

• The maximum number of jobs that can be running

• The maximum number of jobs that can be queued

Load balance

> Scheduling policy wherein jobs are distributed across multiple hosts to even out the workload on each host.

Local resource

> A local resource is defined in a Version 1 MOM configuration file. Local resources cannot be operated on via the qmgr command and are not visible via the qstat and pbsnodes commands. Local resources can be used by the scheduler. See section 5.4.5, "Global and Local Resources" on page 289 in the PBS Professional Administrator's Guide.

Manager

> A person who has been granted Manager privilege by being listed in the Server's managers attribute. A Manager is authorized to use all restricted capabilities of PBS. A PBS Manager may act upon the Server, queues, or jobs. See section 9.2.2.3, "Manager" on page 640 in the PBS Professional Administrator's Guide.

Managing vnode

> The vnode where a shared vnode resource is defined, and which manages the resource.

Master provisioning script, Master script (Hooks)

> The script that makes up the provisioning hook.

Memory-only vnode

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

MOM

> The daemon which runs on an execution host, managing the jobs on that host. *MOM* is the informal name for the process called pbs_mom. One MOM runs on each execution host.

> MOM runs each job when it receives a copy of the job from the Server. MOM creates a new session that is as identical to the user's login session as possible. For example under UNIX, if the user's login shell is csh, then MOM creates a session in which .login is run as well as .cshrc. MOM returns the job's output to the user when directed to do so by the Server.

> MOM is a reverse-engineered acronym that stands for "Machine Oriented Mini-server".

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.

Mother Superior

Mother Superior is the MOM on the head or first host of a multihost job. Mother Superior controls the job, communicates with the Server, and controls and consolidates resource usage information. When a job is to run on more than one execution host, the job is sent to the MOM on the primary execution host, which then starts the job.

Moved jobs

Jobs which were moved to another server

Node

No longer used. See "Execution host".

Non-consumable resource

A non-consumable resource is a resource that is not reduced or taken up by being used. Examples of non-consumable resources are Boolean resources and walltime. See section 5.4.3, "Consumable and Non-consumable Resources" on page 288 in the PBS Professional Administrator's Guide.

Object, PBS object

An element of PBS such as the Server, a queue, or a reservation

Occurrence of a standing reservation

An instance of the standing reservation.

An occurrence of a standing reservation behaves like an advance reservation, with the following exceptions:

- While a job can be submitted to a specific advance reservation, it can only be submitted to the standing reservation as a whole, not to a specific occurrence. You can only specify *when* the job is eligible to run. See the qsub(1B) man page.

- When an advance reservation ends, it and all of its jobs, running or queued, are deleted, but when an occurrence ends, only its running jobs are deleted.

Each occurrence of a standing reservation has reserved resources which satisfy the resource request, but each occurrence may have its resources drawn from a different source. A query for the resources assigned to a standing reservation will return the resources assigned to the soonest occurrence, shown in the resv_nodes attribute reported by pbs_rstat.

Operator

This term means a person who has been granted Operator privilege by being listed in the Server's operators attribute. An Operator can use some but not all of the restricted capabilities of PBS. See .

Overall limit

Limit on the total usage. In the context of server limits, this is the limit for usage at the PBS complex. In the context of queue limits, this is the limit for usage at the queue. An overall limit is applied to the total usage at the specified location. Separate overall limits can be specified at the server and each queue.

Owner, Job owner

The user who submitted a specific job to PBS.

Parameter

A *parameter* specifies an element of the behavior of a component of PBS. For example, MOMs have parameters specifying which events to log, or what the maximum load should be. Parameters are specified by editing the component's configuration files.

PBS Entity

A user, group, or host

pbs Module

The *pbs module* is an interface to PBS and the hook environment. The interface is made up of Python objects, which have attributes and methods. You can operate on these objects using Python code.

PBS Object

An element of PBS such as the Server, a queue, or a reservation

pbsadmin (Windows)

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. This must be "*pbsadmin*".

PBS_HOME

The path containing PBS files. The path under which PBS files are installed on the local system.

PBS_EXEC

The path containing PBS executables. The path under which PBS executables are installed on the local system.

Terminology

PBS Professional

A workload management system consisting of a Server, a Scheduler, and any number of execution hosts each managed by a MOM. PBS accepts batch jobs from users, and schedules them on execution hosts according to the policy chosen by the site. PBS manages the jobs and their output according to site-specified policy.

Peer scheduling

A feature allowing different PBS complexes to automatically run each others' jobs. This way jobs can be dynamically load-balanced across the complexes. Each complex involved in peer scheduling is called a *peer*.

Placement set

A set of vnodes on which jobs can be run, selected so that the job will run as efficiently as possible. Placement sets are used to improve task placement (optimizing to provide a "good fit") by exposing information on system configuration and topology. See section 4.8.32, "Placement Sets" on page 205 in the PBS Professional Administrator's Guide.

Placement set series

The set of placement sets defined by a resource, where each set has the same value for the resource. If the resource takes on N values, there are N placement sets in the series. See section 4.8.32, "Placement Sets" on page 205 in the PBS Professional Administrator's Guide.

Placement pool

All of the placement sets defined at a PBS object. Each queue can have its own placement pool, and the server can have its own placement pool. See section 4.8.32, "Placement Sets" on page 205 in the PBS Professional Administrator's Guide.

Policy, Scheduling policy

The set of rules by which the scheduler selects jobs for execution.

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.

Preempt

Stop one or more running jobs in order to start a higher-priority job.

Preemption level

Job characteristic used to determine whether a job may preempt another or may be preempted, such as being in an express queue, starving, having an

owner who is over a soft limit, being a normal job, or having an owner who is over a fairshare allotment.

Preemption method

The method by which a job is preempted. This can be checkpointing, suspension, or requeueing.

Primary Scheduler

The PBS Professional scheduler daemon which is running during normal operation.

Primary Server

The PBS Professional server daemon which is running during normal operation.

Project

In PBS, a project is a way to group jobs independently of users and groups. A project is a tag that identifies a set of jobs. Each job's project attribute specifies the job's project.

Project limit

This is a limit applied to the total used by a project, whether the limit is a generic project limit or an individual project limit.

Provision

To install an OS or application, or to run a script which performs installation and/or setup

Provisioned vnode

A vnode which, through the process of provisioning, has an OS or application that was installed, or which has had a script run on it

Provisioning hook

The hook which performs the provisioning, either by calling other scripts or by running commands

Provisioning tool

A tool that performs the actual provisioning, e.g. SGI Tempo.

Pulling queue

In peer scheduling, the queue into which jobs are pulled, and from which they are run

Queue

A *queue* is a named container for jobs at a Server. There are two types of queues in PBS: routing queues and execution queues. A *routing queue* is a queue used to move jobs to other queues including those that exist on other

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*).

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.

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.

Reject an action (Hooks)

An action is *rejected* when a hook prevents the action from taking place.

Requeue

The process of stopping a running job and putting it back into the *queued* ("*Q*") state.

Rerunnable

If a running PBS job can be terminated and then restarted from the beginning without harmful side effects, the job is rerunnable. The job's Rerunnable attribute must be set to *y* in order for PBS to consider a job to be rerunnable.

Reservation Degradation

PBS attempts to ensure that reservations run by finding usable vnodes when reservation vnodes become unavailable.

Resource

A *resource* can be something used by a job, such as CPUs, memory, high-speed switches, scratch space, licenses, or time, or it can be an arbitrary item defined for another purpose. PBS has built-in resources, and allows custom-defined resources. See section , "PBS Resources" on page 281 in the PBS Professional Administrator's Guide.

Restart

A job that was stopped after being checkpointed while previously executing is executed again, starting from the point where it was checkpointed.

Restart File

> The job-specific file that is written by the checkpoint script or tool. This file contains any information needed to restart the job from where it was when it was checkpointed.

Restart Script

> The script that MOM runs to restart a job. This script is common to all jobs, and so must use the information in a job's restart file to restart the job.

Route a job

> When PBS moves a job between queues. PBS provides a mechanism whereby a job is automatically moved from a routing queue to another queue. This is performed by PBS. The resource request for each job in a routing queue is examined, and the job is placed in a destination queue which matches the resource request. The destination queue can be an execution queue or another routing queue.

Routing queue

> A queue that serves as a temporary holding place for jobs, before they are moved to another queue. Jobs cannot run from routing queues.

Scheduler

> The *scheduler* is the daemon which implements the site's job scheduling policy controlling when and where each job is run. The scheduler is the process called `pbs_sched`.

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).

Scheduling policy

> Scheduling policy determines when each job runs, and and how much of each resource it can use. Scheduling policy consists of a system for determining the priority of each job, combined with a set of limits on how many jobs can be run, and/or how much of each resource can be used.

Schema Admins (Windows)

> 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.

Secondary Scheduler

> The PBS Professional scheduler daemon which takes over when the primary scheduler is not available.

Secondary Server

The PBS Professional server daemon which takes over when the primary server fails.

Sequence_number

The numeric part of a job or job array identifier, e.g. **1234.**

Server

The central PBS daemon, which does the following:

- Handles PBS commands

- Receives and creates batch jobs

- Sends jobs for execution

The server is the process called `pbs_server`.

Each PBS complex has one primary server, and if the complex is configured for failover, a secondary server.

The server contains a licensing client which communicates with the licensing server for licensing PBS jobs.

Shared resource

A vnode resource defined and managed at one vnode, but available for use at others.

Shrink-to-fit job

A job that requests the min_walltime resource. A shrink-to-fit job requests a running time in a specified range, where min_walltime is required, and max_walltime is not. PBS computes the actual walltime.

Sister

Any MOM that is not on the head or first host of a multihost job. A sister is directed by the Mother Superior. Also called a *subordinate MOM*.

Sisterhood

All of the MOMs involved in running a particular job.

Snapshot Checkpoint

The checkpoint script or tool writes a restart file, and the job continues to execute. The job resumes from this start file if the system experiences a problem during the job's subsequent execution.

Soonest occurrence of a standing reservation

The occurrence which is currently active, or if none is active, then it is the next occurrence.

Stage in

        The process of moving one or more job-related files from a storage location to the execution host before running the job.

Stage out

        The process of moving one or more job-related files from the execution host to a storage location after running the job.

Staging and execution directory

        The *staging and execution directory* is a directory on the execution host where the following happens:

- Files are staged into this directory before execution

- The job runs in this directory

- Files are staged out from this directory after execution

A job-specific staging and execution directory can be created for each job, or PBS can use a specified directory, or a default directory. See section 12.14.1, "The Job's Staging and Execution Directories" on page 835 in the PBS Professional Administrator's Guide.

Standing reservation

        An advance reservation which recurs at specified times. For example, the user can reserve 8 CPUs and 10GB every Wednesday and Thursday from 5pm to 8pm, for the next three months.

State

        The PBS server, vnodes, reservations, and jobs can be in various states, depending on what PBS is doing. For example the server can be *idle* or *scheduling*, vnodes can be *busy* or *free*, and jobs can be *queued* or *running*, among other states. For a complete description of states, see "States" on page 411.

Strict ordering

        A scheduling policy where jobs are run according to policy order. If the site-specified policy dictates a particular priority ordering for jobs, that is the order in which they are run. Strict ordering can be modified by backfilling in order to increase throughput. See "Backfilling".

Subject

        A process belonging to a job run by an authorized, unprivileged user (a job submitter.)

Subjob

        One of the jobs in a job array, e.g. `1234[7]`, where `1234[]` is the job array itself, and `7` is the index. Queued subjobs are not individually listed

in the queue; only their job array is listed. Running subjobs are individually listed.

Subjob index

The unique index which differentiates one subjob from another. This must be a non-negative integer.

Subordinate MOM

Any MOM that is not on the head or first host of a multihost job.   A subordinate MOM is directed by the Mother Superior. Also called a *sister*.

Task

A process belonging to a job. 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.

Three-server Configuration

One form of redundant license 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

Token

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

User

Has two meanings:

1. A person who submits jobs to PBS, as differentiated from Operators, Managers and administrators. See section 9.2.2.1, "User" on page 639 in the PBS Professional Administrator's Guide.

2. A system user, identified by a unique character string (the user name) and by a unique number (the user ID). Any person using the system has a username and user ID.

User access, Access by user

The specified user is allowed access at the Server, queues, and reservations .

User ID, UID

A unique numeric identifier assigned to each user.

User limit

Refers to configurable limits on resources and jobs. A limit placed on one or more users, whether generic or individual.

vchunk

The part of a chunk that is supplied by one vnode. If a chunk is broken up across multiple vnodes, each vnode supplies a vchunk.

Version 1 configuration file

MOM configuration file containing MOM configuration parameters. See Chapter 3, "MOM Parameters", on page 267.

Version 2 configuration file

Vnode configuration file containing vnode attribute settings. Created using `pbs_mom -s insert` command.

Virtual processor, VP

PBS can treat a vnode as if it has more processors available than the number of physical processors. When resources_available.ncpus is set to a number higher than the actual number of physical processors, the vnode can be said to have virtual processors. Also called logical processors.

Vnode

A virtual node, or *vnode*, is an abstract object representing a set of resources which form a usable part of an execution host. 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.

Chapter 2

PBS Commands

This chapter contains a description of each PBS command. Each description includes any options to the command.

2.1   Windows Requirements

Under Windows, use double quotes when specifying arguments to PBS commands.

2.2   Requirements for Commands

Some PBS commands require root privilege or PBS Operator or Manager privilege in order to run. Some can be executed by anyone, but the output depends upon the privilege of the user.

Most PBS commands require that the Server be running; some require that MOMs be running.

PBS Commands

The following table lists the commands, and indicates the permissions required to use each, and whether the server or MOM must be running.

Table 2-1: Permission and Daemon Requirements for Commands

Command	Permissions Required	Server Must Be Running?	MOM Must Be Running?
"mpiexec"	Any	No	No
"nqs2pbs"	Any	No	No
"pbs"	Users can only get status	No	No
"pbs-report"	Root	No	No
"pbsdsh"	Any	No	Yes
"pbsfs"	Any	No	No
"pbsnodes"	Result depends on permission	Yes	No
"pbsrun"	Can be run by root or PBS administrator only	No	No
"pbsrun_unwrap"	Can be run by root only	No	No
"pbsrun_wrap"	Can be run by root only	No	No
"pbs_account"	Admin on Windows	No	No
"pbs_attach"	Any	Yes	Yes
"pbs_dataservice"	Root on UNIX; Admin on Windows	No	No
"pbs_ds_password"	Root on UNIX; Admin on Windows	No	No
"pbs_hostn"	Any	No	No
"pbs_idled"	Can be run only by root or PBS administrator	No	No

Table 2-1: Permission and Daemon Requirements for Commands

Command	Permissions Required	Server Must Be Running ?	MOM Must Be Running ?
"pbs_lamboot"	Any	No	No
"pbs_migrate_users"	Any	Yes	No
"pbs_mkdirs"	Can be run by PBS administrator only	No	No
"pbs_mom"	Can be run only by root or PBS administrator	No	No
"pbs_mom_globus"	Can be run only by root or PBS administrator	No	No
"pbs_mpihp"	Any	Yes	Yes
"pbs_mpilam"	Any	Yes	Yes
"pbs_mpirun"	Any	Yes	Yes
"pbs_password"	Any	Yes	No
"pbs_probe"	Can be run only by root or PBS administrator	No	No
"pbs_python"	Only to call pbs.server()	No	No
"pbs_rdel"	Any	Yes	No
"pbs_renew"	Any	No	No
"pbs_rstat"	Any	Yes	No
"pbs_rsub"	Any	Yes	No
"pbs_sched"	Can be run only by root or PBS administrator	No	No
"pbs_server"	Can be run only by root or PBS administrator	No	No

PBS Commands

Table 2-1: Permission and Daemon Requirements for Commands

Command	Permissions Required	Server Must Be Running ?	MOM Must Be Running ?
`"pbs_tclsh"`	Any	No	No
`"pbs_tmrsh"`	Any	No	Yes
`"pbs_topologyinfo"`	Root	No	No
`"pbs_wish"`	Any	No	No
`"printjob"`	Can be run only by root or PBS administrator	No	No
`"qalter"`	Any	Yes	No
`"qdel"`	Any	Yes	No
`"qdisable"`	Requires Manager or Operator privilege to run	Yes	No
`"qenable"`	Requires Manager or Operator privilege to run	Yes	No
`"qhold"`	Some holds can be set by root or administrator only	Yes	No
`"qmgr"`	Any	Yes	No
`"qmove"`	Any	Yes	No
`"qmsg"`	Any	Yes	No
`"qorder"`	Any	Yes	No
`"qrerun"`	Any	Yes	No
`"qrls"`	Some holds can be released by root or administrator only	Yes	No

Table 2-1: Permission and Daemon Requirements for Commands

Command	Permissions Required	Server Must Be Running ?	MOM Must Be Running ?
`"qrun"`	Can be run only by Operator or Manager	Yes	No
`"qselect"`	Any	Yes	No
`"qsig"`	Any	Yes	No
`"qstart"`	Can be run only by Operator or Manager	Yes	No
`"qstat"`	Result depends on permission	Yes	No
`"qstop"`	Can be run only by Operator or Manager	Yes	No
`"qsub"`		Yes	No
`"qterm"`	Can be run only by Operator or Manager	Yes	No
`"tracejob"`	Can be run only by root or PBS administrator	No	No
`"xpbs"`		Yes	No
`"xpbsmon"`		Yes	No

2.3   mpiexec

Runs MPI programs under PBS on Linux

2.3.1 Synopsis

mpiexec

mpiexec --version

2.3.2 Description

The PBS `mpiexec` command provides the standard `mpiexec` interface on an SGI Altix, ICE or UV running supported versions of ProPack or Performance Suite. If executed on a non-Altix, non-ICE, and non-UV system, it will assume it was invoked by mistake. In this case it will use the value of PBS_O_PATH to search for the correct `mpiexec`. If one is found, the PBS `mpiexec` will exec it.

The PBS `mpiexec` calls the SGI `mpirun(1)`. The name of the array to use when invoking `mpirun` is user-specifiable via the PBS_MPI_SGIARRAY environment variable.

It is transparent to the user; MPI jobs submitted outside of PBS will run as they would normally. MPI jobs can be launched across multiple SGI machines. PBS will manage, track, and cleanly terminate multi-host MPI jobs. PBS users can run MPI jobs within specific partitions.

If the PBS_MPI_DEBUG environment variable's value has a nonzero length, PBS will write debugging information to standard output.

2.3.3 Usage

The PBS `mpiexec` command presents the `mpiexec` interface described in section "4.1 Portable MPI Process Startup" of the "MPI-2: Extensions to the Message-Passing Interface" document in `http://www.mpiforum.org/docs/mpi-20-html/node42.htm`

2.3.4 Options

--version

> The `mpiexec` command returns its PBS version information and exits. This option can only be used alone.

2.3.5 Requirements

Altix, ICE, or UV running a supported version of ProPack or Performance Suite.

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()` on each node of a multinode MPI job must be the same as it is on the mother superior node.

In order to run multihost jobs, the SGI Array Services must be correctly configured. SGI systems communicating via SGI's Array Services must all use the same version of the `sgi-arraysvcs` package. SGI 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.

2.3.6   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.

2.3.7   Path

PBS' `mpiexec` is located in `PBS_EXEC/bin/mpiexec`.

2.3.8   See Also

The PBS Professional Administrator's Guide

SGI man pages: SGI's `mpirun(1)`, SGI's `mpiexec_mpt(1)`, SGI's `array_services(5)`

PBS man pages: `pbs_attach(8B)`

2.4   nqs2pbs

Converts NQS job scripts to PBS format.

2.4.1 Synopsis

nqs2pbs nqs_script [pbs_script]

nqs2pbs --version

2.4.2 Description

This utility converts an existing NQS job script to work with PBS and NQS. The existing script is copied and PBS directives, #PBS , are inserted prior to each NQS directive #QSUB or #@$, in the original script.

Certain NQS date specification and options are not supported by PBS. A warning message will be displayed indicating the problem and the line of the script on which it occurred.

If any unrecognizable NQS directives are encountered, an error message is displayed. The new PBS script will be deleted if any errors occur.

2.4.3 Options

--version
> The `nqs2pbs` command returns its PBS version information and exits. This option must be used alone.

2.4.4 Operands

nqs_script
> Specifies the file name of the NQS script to convert. This file is not changed.

pbs_script
> If specified, it is the name of the new PBS script. If not specified, the new file name is nqs_script.new .

NOTES

Converting NQS date specifications to the PBS form may result in a warning message and an incompletely converted date. PBS does not support date specifications of "*today*", "*tomorrow*", or the name of the days of the week such as "*Monday*". If any of these are encountered in a script, the PBS specification will contain only the time portion of the NQS specification, i.e. #PBS -a hhmm[.ss]. It is suggested that you specify the execution time on the `qsub` command line rather than in the script.

Note that PBS will interpret a time specification without a date in the following way:

• If the time specified has not yet been reached, the job will become eligible to run at that time today.

• If the specified time has already passed when the job is submitted, the job will become eligible to run at that time tomorrow.

This command does not support time zone identifiers. All times are taken as local time.

2.4.5 See Also

`qsub(1B)`

2.5 pbs-report

Prints PBS job statistics.

2.5.1 Synopsis

pbs-report [--age seconds[:offset]] [--account account] [--begin -b yyyymmdd[:hhmm[ss]]] [--count -c] [--cpumax seconds] [--cpumin seconds] [--csv character] [--dept department] [--end -e yyyymmdd[:hhmm[ss]]] [--exit -x integer] [--explainwait] [--group UNIX group] [--help] [--host hostname] [--inclusive] [--index key] [--man] [--negate option] [--package solver] [--point yyyymmdd[:hhmm[ss]]] [--queue PBS queue] [--range span] [--reslist] [--sched] [--sort field] [--time option] [--user username] [--verbose] [--vsort field] [--waitmax seconds] [--waitmin seconds] [--wall] [--wallmax seconds] [--wallmin seconds]

pbs-report --version

2.5.2 Description

Allows the PBS Administrator to generate a report of job statistics from the PBS accounting logfiles. Options to the `pbs-report` command control how jobs are selected for reporting and how reports are generated.

The `pbs-report` command is not available on Windows.

Before first using `pbs-report`, the Administrator is advised to tune the configuration to match the local site by editing the file PBS_EXEC/lib/pm/PBS.pm.

2.5.2.1 Permissions

This command can be run only by root.

2.5.2.2 Selecting Jobs For Reporting

2.5.2.2.i Filtering Jobs by Dates or Times

--begin, --end, --range, --age, --point

--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.

Jobs are listed by start time only, regardless of whether end time is specified via --end or --inclusive.

--range

A short-hand 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.

2.5.2.2.ii Filtering Jobs by Attribute

--cpumax, --cpumin, --waitmax, --waitmin, --wallmax, --wallmin

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.

2.5.2.2.iii Filtering Jobs by User or Department

--dept, --group, --user

--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.

2.5.2.2.iv   Filtering Jobs by Job Property

--host, --exit, --package, --queue

--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.

2.5.2.2.v   Filtering Jobs by Account String

--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.

2.5.2.2.vi   Negating Filters

--negate

> The --negate option 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.

2.5.2.3   Generating Reports

Several report types can be generated, each indexed and sorted according to the user's needs.

--verbose

> Generates a wide tabular output with detail for every job selected. It can be used to generate output for import to a spreadsheet. Verbose reports may be sorted on any field using the --vsort option. Default: summary report only.

--reslist

> Generates a tabular output with detail on resources requested for every job selected. Resource list reports may be sorted on any field using the --vsort option. Default: summary report only.

--inclusive

> Allows a user to require that the job's start time also falls within the date range.
>
> Jobs are listed by start time only, regardless of whether end time is specified via --end or --inclusive.

--index

> Allows specification of the field on which data in the summary should be grouped. Fields listed in the option description are mutually exclusive. The selected field will be 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

> 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 how the two options interact.

--vsort

>   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).

2.5.3   Options to `pbs-report`

--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. --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.

>   Jobs are listed by start time only, regardless of whether end time is specified via --end or --inclusive.

--count -c

>   Display a numeric count of matching jobs. Currently only valid with --cpumax for use in monitoring rapidly-exiting jobs.

--cpumax seconds

>   Filter out any jobs which have more than the specified number of CPU seconds.

--cpumin seconds

>   Filter out any jobs which have less than the specified number of CPU seconds.

--dept -d department

>   Limit results to those jobs whose owners are in the indicated department . Default: any. This option only works in conjunction with an LDAP server which supplies department codes. See also the --group option. Multiple values may be concatenated with colons or specified with multiple instances of --dept.

--end -e yyyymmdd[:hhmm[ss]]
Report end date and optional time. Default: most recent log data. --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.

Jobs are listed by start time only, regardless of whether end time is specified via --end or --inclusive.

--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. Group is defined by the operating system. Multiple values may be concatenated with colons or specified with multiple instances of --group.

--help -h
Prints a brief help message and exits.

--host -m execution host
Limit results to the specified execution host. Multiple values may be concatenated with colons or specified with multiple instances of --host.

--inclusive key
Limit results to jobs which had start times in the range.

Jobs are listed by start time only, regardless of whether end time is specified via --end or --inclusive.

--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
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 --pack-
>   age. Valid values are can be seen by running a report with the --index
>   package option. This option keys on custom resources requested at job sub-
>   mission time. Sites not using such custom resources will have all jobs
>   reported under the catchall 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 period

>   Time period used is period before now. For example, if the period given is
>   "week", pbs-report looks at all jobs which have finished and which
>   were running any time from a week ago to now. Default: all. Valid values
>   for period are *today, week, month, quarter*, and *year*. This option
>   silently supersedes --begin and --end, and is superseded by --age.

--reslist

>   Include resource requests for all matching jobs. This option is mutually
>   exclusive with --verbose.

--sched -t

>   Generate a brief statistical analysis of Scheduler cycle times. No other data
>   on jobs is reported.

--sort -s field

>   Field by which to sort reports. Default: user. Valid values are *cpu, date,
>   dept, host, jobs, package, queue, suspend* (aka muda), *wait*, and *wall*.

--time option

>   Valid values: "*full*", "*partial*". Used to indicate how time should be
>   accounted. The default of "*full*" means that entire job's CPU and wall time
>   is counted in the report if the job ended during the report's date range. With
>   the "*partial*" option, only CPU and wall time during the report's date range
>   are counted.
>
>   By default, time is credited at the point when the job ended. This can be
>   changed using the --inclusive option. 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. See --inclusive.

--user -u username
Limit results to the specified username. Multiple values may be concatenated with colons or specified with multiple instances of --user.

--verbose -v
Include attributes for reported jobs. Subjobs are shown, but not job arrays. Default: no attributes.

--version
The `pbs-report` command returns its PBS version information and exits. This option can only be used alone.

--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.

2.5.4 Examples

"How much in the way of resources did every job this month waiting more than 10 minutes request?"

```
pbs-report --range month --waitmin 600 --reslist
```

This information might be valuable to determine if some simple resource additions (e.g. more memory or more disk) might increase overall throughput of the cluster.

2.5.4.1 Statistical Analysis

At the bottom of the summary statistics, prior to the job set summary, is a statistical break-down of the values in each column. Example:

| | # of | Total | Total | | Average |
Date	jobs	CPU Time	Wall Time	Efcy.	Wait Time
TOTAL	1900	10482613	17636290	0.594	1270
Minimum	4	4715	13276	0.054	221
Maximum	162	1399894	2370006	1.782	49284
Mean	76	419304	705451	0.645	2943
Deviation	41	369271	616196	0.408	9606
Median	80	242685	436724	0.556	465

This summary should be read in column format. The values each represent a statistical data point in the column. For instance, while the minimum number of jobs run in one day was 4 and the maximum 162, these values do not correlate to the 4715 and 1399894 CPU seconds listed as minima and maxima.

In the Job Set Summary section, the values should be read in rows, as shown here:

	Minimum	Maximum	Mean	Standard 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.

2.5.4.2 Cluster Monitoring

The --count and --cpumax functions are intended to allow an administrator to periodically run this script 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.

2.5.5 Standard Error

The pbs-report command will write a diagnostic message to standard error for each error occurrence.

2.5.6 Exit Status

Zero upon successful processing of all operands.

Greater than zero if the pbs-report command fails to process any operand.

2.5.7 See Also

The PBS Professional Administrator's Guide, pbs_server(8B), pbs_sched(8B), pbs_mom(8B)

2.6 pbs_account

Manage PBS service account

2.6.1 Synopsis

pbs_account [-a PBS service account name] [-c] [-s] [-p password] [--reg service_path] [--unreg service_path] [-o output_path] [--ci]

2.6.2 Description

The `pbs_account` command is used to manage the PBS service account. It is used to create the account, set or validate the account password, add privileges to the account, and register or unregister the account with the SCM.

2.6.3 Permissions

This command can be run by administrators only.

2.6.4 Platforms

This command is available on Windows only.

2.6.5 Options

-a <account name>
> Specifies account name.

-c [<password>]

- • If specified account does not exist, creates the account with the password.
- • If specified account exists, validates password against it.

> If password is not specified, user is prompted for password.

-o <output path>
> Prints `stdout` and `stderr` messages in specified output path

-p [<password >]
> Updates the PBS service account password. If no password is specified, the user is prompted for a password.

-s

> Adds necessary privileges to the PBS service account. Grants the "Create Token Object", "Replace Process Level Token", "Log On as a Service", and "Act as Part of the Operating System" privileges to PBS service account.

--reg-server <path to server>   [<password>]

--reg-mom   <path to MOM>   [<password>]

--reg-sched <path to scheduler> [<password>]

--reg-rshd   <path to rshd>   [<password>]

> Registers the PBS service with the SCM, instructing it to run the services under the PBS service account and supplied password. <path> must be in double quotes.

--unreg-server <path to server>

--unreg-mom   <path to MOM>

--unreg-sched <path to scheduler>

--unreg-rshd   <path to rshd>

> Unregisters the PBS service with the SCM. <path> must be in double quotes.

--ci

> Prints actions taken by `pbs_account` while creating PBS service account when -c option is used

<no options>

> Prints name of PBS service account, if it exists. Exit value is 0.

2.6.6   Examples

- To create the PBS service account:

  pbs_account –c –s –p password

- To change the PBS service account:

  pbs_account [--reg service_path] [- a PBS service account name]

- To register the server, scheduler, MOM, and rshd services:

  pbs_account --reg "\Program Files\PBS Pro\exec\sbin\pbs_server.exe" [-p <password>]

  pbs_account --reg "\Program Files\PBS Pro\exec\sbin\pbs_mom.exe" [-p <password>]

  pbs_account --reg "\Program Files\PBS Pro\exec\sbin\pbs_sched.exe" [-p <password>]

  pbs_account --reg "\Program Files\PBS Pro\exec\sbin\pbs_rshd.exe" [-p <password>]

2.7 pbs_attach

Attaches a session ID to a PBS job

2.7.1 Synopsis

pbs_attach [-j jobid] [-m port] -p pid

pbs_attach [-j jobid] [-m port] [-P] [-s] cmd [arg ...]

pbs_attach --version

2.7.2 Description

The `pbs_attach` command associates the processes in a session with a PBS job by attaching the session ID to the job. This allows PBS MOM to monitor and control those processes.

MOM uses process IDs to determine session IDs, which are put into MOM's task list (attached to the job.) All process IDs in a session are then associated with the job.

When a command *cmd* is given as an operand, the `pbs_attach` process becomes the parent process of *cmd*, and the session ID of `pbs_attach` is attached to the job.

The -p option cannot be used with the -P or -s options or the *cmd* operand.

This command is not supported under Windows.

2.7.3 Options to pbs_attach

-j jobid

> The job ID to which the session ID is to be attached. If jobid is not specified, a best effort will be made to determine the job to which to attach the session.

-m port

> The port at which to contact MOM. Default: value of *PBS_MANAGER_SERVICE_PORT* from `pbs.conf`

-p pid

> Process ID whose session ID will be attached to the job. Default: process ID of `pbs_attach`.

-P

> Attach sessions of both `pbs_attach` and the parent of `pbs_attach` to job. When used with -s option, this means the sessions of the new fork() ed

pbs_attach and its parent, which is pbs_attach, are attached to the job.

-s

Starts a new session by fork()-ing pbs_attach. The session ID of the new pbs_attach is attached to the job.

--version

The pbs_attach command returns its PBS version information and exits. This option can only be used alone.

2.7.4   Operands

cmd

Name of command whose process ID is to be associated with the job.

2.7.5   Exit Status

0

Success

1

Any error following successful command line processing. A message is printed to standard error.

If *cmd* is specified, pbs_attach waits for *cmd* to exit, then exits with the exit value of *cmd*.

If *cmd* is not specified, pbs_attach exits after attaching the session ID(s) to the job.

2.7.6   See Also

The PBS Professional Administrator's Guide, pbs_mom(8B), pbs_tmrsh(8B), tm(3)

2.8   pbs_dataservice

Start, stop, or check the status of PBS data service

2.8.1   Synopsis

pbs_dataservice [start|stop|status]

2.8.2 Description

The `pbs_dataservice` command starts, stops or gets the status of the PBS data service.

2.8.3 Permission

On UNIX, root privilege is required to use this command. On Windows, Admin privilege is required.

2.8.4 Arguments

start

Starts the PBS data service.

stop

Stops the PBS data service.

Can be used only when PBS server is not running.

status

Displays the status of the PBS data service, as follows:

- Data service running

  `PBS Data Service running`

- Data service not running

  `PBS Data Service not running`

2.8.5 Exit Status

Zero for success

Non-zero for failure

2.9 pbs_ds_password

Sets or changes data service user account or its password

2.9.1 Synopsis

pbs_ds_password [-r] [-C username]

2.9.2 Description

You can use this command to change the user account or account password for the data service. Blank passwords are not allowed.

2.9.3 Permissions

On UNIX, root privilege is required to use this command. On Windows, Admin privilege is required.

2.9.4 Restrictions

Do not run this command if failover is configured. It is important not to inadvertently start two separate instances of the data service on two machines, thus potentially corrupting the database.

2.9.5 Options to `pbs_ds_password`

-C <username>

Changes user account for data service to specified account. Specified user account must already exist.

On UNIX-based systems, the specified user account must not be root.

On Windows, the specified user account must match the PBS service account (which can be any user account.)

This option cannot be used while the data service is running.

Can be used with the -r option to automatically generate a password for the new account.

-r

Generates a random password. The data service is updated with the new password.

Can be used with the -C option.

<no option>

Asks the user to enter a new password twice. Entries must match. Updates data service with new password.

2.9.6 Exit Status

Zero on success.

Non-zero on failure.

2.10 pbs_hostn

Reports hostname and network address(es)

2.10.1 Synopsis

pbs_hostn [-v] hostname

pbs_hostn --version

2.10.2 Description

The `pbs_hostn` command takes a hostname, and reports the results of both `gethostbyname(3)` and `gethostbyaddr(3)` system calls. Both forward and reverse lookup of hostname and network addresses need to succeed in order for PBS to authenticate a host.

Running this command can assist in troubleshooting problems related to incorrect or non-standard network configuration, especially within clusters.

2.10.3 Options

-v

> Turns on verbose mode.

--version

> The `pbs_hostn` command returns its PBS version information and exits. This option can only be used alone.

2.10.4 Operands

The `pbs_hostn` command accepts a hostname operand either in short name form, or in fully qualified domain name (FQDN) form.

2.10.5   Standard Error

The `pbs_hostn` command will write a diagnostic message to standard error for each error occurrence.

2.10.6   Exit Status

Zero upon successful processing of all the operands presented to the `pbs_hostn` command.

Greater than zero if the `pbs_hostn` command fails to process any operand.

2.10.7   See Also

The PBS Professional Administrator's Guide and the following manual page: `pbs_server(8B)`

2.11   pbs_idled

PBS daemon to watch the console and inform `pbs_mom` of idle time

2.11.1   UNIX/Linux Synopsis

pbs_idled [-w wait_time] [-f idle_file] [-D display] [-r reconnect_delay]
pbs_idled --version

2.11.2   Windows Synopsis

pbs_idled [start | stop]
pbs_idled --version

2.11.3   UNIX/Linux Description

On UNIX/Linux, the `pbs_idled` program sits and watches an X windows display and communicates the idle time of the display back to PBS. If the mouse is moved or a key is touched, PBS is informed that the node is busy.

This program should be run out of the system-wide Xsession file. It should be run in the background before the window manager is run. If this program is run outside of the Xsession, it will need to be able to make a connection to the X display. See the xhost or xauth man pages for a description of X security.

2.11.4 Windows Description

On Windows, `pbs_idled` reads its polling interval from a file called `idle_poll_time` which is created by MoM. The process monitors keyboard, mouse, and console activity, and updates a file called `idle_touch` when it finds user activity. The `idle_touch` file is created by MoM.

2.11.5 UNIX/Linux Options to `pbs_idled`

-w <wait_time>
>
> Granularity between when the daemon checks for events or pointer movement.

-f <idle_file>
>
> Update file times on <idle_file>. PBS will not monitor any other than the default.

-D <display>
>
> The display to connect to and monitor.

-r <reconnect_delay>
>
> The amount of time to try and reconnect to the X display if the previous attempt was unsuccessful.

--version
>
> The `pbs_idled` command returns its PBS version information and exits. This option can only be used alone.

2.11.6 Windows Options to `pbs_idled`

start
>
> Starts the `pbs_idled` process.

stop
>
> Stops the `pbs_idled` process.

--version
>
> The `pbs_idled` process returns its PBS version information and exits. This option can only be used alone.

2.11.7   See Also

The PBS Professional Administrator's Guide and the following manual pages:
`pbs_mom(8B), xhost(1), xauth(1)`

2.12   pbs

Start, stop, restart, or get the PIDs of PBS daemons

2.12.1   Synopsis

pbs [start|stop|restart|status]

2.12.2   Description

The `pbs` command starts, stops or restarts all PBS daemons on the local machine. Does not affect other hosts. It also reports the PIDs of all daemons when given the status argument.

2.12.2.1   Caveats

This command operates only on daemons that are marked as active in `pbs.conf`. For example, if PBS_START_MOM is set to *0* in the local `pbs.conf`, this command will not operate on `pbs_mom`, and will not start, stop, or restart `pbs_mom`.

2.12.2.2   Privilege

PBS Manager privilege is required to use this command.

2.12.3   Arguments

start

>Each daemon on the local machine is started. PBS reports the number and type of licenses available, as well as the name of the license server. Any running jobs are killed.

stop

>Each daemon on the local machine is stopped, and its PID is reported.

restart

        All daemons on the local machine are stopped, then they are restarted. PBS reports the name of the license server and the number and type of licenses available.

status

        PBS reports the PID of each daemon on the local machine.

2.12.4   See Also

The PBS Professional Administrator's Guide, `pbs_mom(8B)`, `pbs_server(8B)`, `pbs_sched(3)`

2.13   pbs_interactive

Register, unregister, or get the version of PBS_INTERACTIVE service

2.13.1   Synopsis

pbs_interactive [R | U]

pbs_interactive --version

2.13.2   Description

The `pbs_interactive` process is used to register or unregister the Windows PBS_INTERACTIVE service. The service must be registered manually; the installer does not register it.

On Windows, the PBS_INTERACTIVE service itself monitors logging in and out by users, starts a `pbs_idled` process for each user logging in, and stops the `pbs_idled` process of each user logging out.

2.13.2.1   Privilege

Administrator privilege is required to use this command.

2.13.3 Arguments

R
> Registers the PBS_INTERACTIVE service.

U
> Unregisters the PBS_INTERACTIVE service.

--version
> The `pbs_interactive` command returns its PBS version information and exits. This option can only be used alone.

2.14 pbs_lamboot

PBS front end to LAM's `lamboot` program

2.14.1 Synopsis

pbs_lamboot

pbs_lamboot --version

2.14.2 Description

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 running Linux 2.4 or higher.

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 nodes 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*.

2.14.3   Options

--version

> The pbs_lamboot command returns its PBS version information and exits. This option can only be used alone.

2.14.4   Operands

The operands for pbs_lamboot are the same as for lamboot.

2.14.5   Environment Variables and Path

The PATH on remote machines must contain PBS_EXEC/bin.

2.14.6   See Also

The PBS Professional Administrator's Guide, lamboot(1), tm(3)

2.15   pbs_migrate_users

Transfers per-user or per-server passwords between PBS servers during a migration upgrade

2.15.1   Synopsis

pbs_migrate_users old_server new_server

pbs_migrate_users --version

2.15.2   Description

The pbs_migrate_users command is used to transfer the per-user or per-server password of a PBS user from one server to another during a migration upgrade.

Users' passwords on the old server are not deleted.

Available on Windows and supported Linux x86 and x86_64 platforms only.

2.15.3　　Options

--version

        The `pbs_migrate_users` command returns its PBS version information and exits. This option can only be used alone.

2.15.4　　Operands

The format of *old_server* and *new_server* is

hostname[:port_number]

2.15.5　　Exit Status

0

        Success

-1

        Writing out passwords to files failed.

-2

        Communication failure between *old_server* and **new_server**.

-3

        Single_signon_password_enable not set in either old_server or new_server

-4

        User running `pbs_migrate_users` not authorized to migrate users.

2.15.6　　See Also

`pbs_password(8B)`

2.16　　**pbs_mkdirs**

Create, or fix the permissions of, the directories and files used by PBS

2.16.1   Synopsis

pbs_mkdirs

pbs_mkdirs [mom | sched | server]

2.16.2   Description

Runs on Windows only. If the directories and files used by PBS exist, the `pbs_mkdirs`
command fixes their permissions. If the directories and/or files do not exist, the
`pbs_mkdirs` command creates them, with the correct permissions. The `pbs_mkdirs`
command always examines the following directories and files:

```
pbs.conf
PBS_EXEC
PBS_HOME/spool
PBS_HOME/undelivered
PBS_HOME/pbs_environment
```

2.16.3   Options

mom
> The `pbs_mkdirs` command examines the following additional items:
> > `PBS_HOME/mom_priv`
> >
> > `PBS_HOME/mom_logs`

sched
> The `pbs_mkdirs` command examines the following additional items:
> > `PBS_HOME/sched_priv`
> >
> > `PBS_HOME/sched_logs`

server
> The `pbs_mkdirs` command examines the following additional items:
> > `PBS_HOME/server_priv`
> >
> > `PBS_HOME/server_logs`

(no options)
> The `pbs_mkdirs` command examines all of the files and directories specified for each of the mom, server, and sched options.

2.16.4   See Also

The PBS Professional Administrator's Guide, `pbs_probe(8B)`

2.17   pbs_mom

The PBS job monitoring and execution daemon

2.17.1   Synopsis

pbs_mom [-a alarm_timeout] [-C checkpoint_directory] [-c config_file] [-d home_directory]
*   [-L logfile] [-M TCP_port] [-n nice_val] [-N] [-p\|-r] [-R UDP_port] [-S server_port] [-*
*   s script_options]*

pbs_mom --version

2.17.2   Description

The `pbs_mom` command starts the PBS job monitoring and execution daemon, called *MOM*.
A special HPC Basic Profile MOM manages jobs for an HPC Basic Profile Server. This
MOM is called the HPCBP MOM; the standard MOM is called the MOM. See HPCBP
MOM below.

The standard MOM 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 vir-
tual nodes, in which case one MOM would manage all of the host's vnodes. See the PBS Pro-
fessional Administrator's Guide.

The MOM's 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 MOM will write events to its log file. The MOM writes its PBS version and build
information to the logfile whenever it starts up or the logfile is rolled to a new file.

The executable for `pbs_mom` is in `PBS_EXEC/sbin`, and can be run only by root.

2.17.2.1   HPCBP MOM

The HPCBP MOM acts as an intermediary between the PBS complex and the HPCBP Server. The HPCBP MOM does the following:

- Converts between formats used by PBS and HPCBP for information going both to and from the HPC Basic Profile Server.

- Takes job requests from the PBS complex, converts them to JSDL, and submits these converted jobs to the HPC Basic Profile Server.

- Retrieves job and node status information from the HPCBP Server and hands this status information to the PBS Server.

One HPCBP MOM is required for each HPCBP Server.

The HPCBP MOM is created by setting the HPCBP-specific vnode attributes on the vnode that will host the HPCBP MOM. These are the attributes prefixed by "`hpcbp_`". See the `pbs_node_attributes(7B)` man page.

Many of the configuration variables for `pbs_mom` behave differently in the HPCBP MOM. See the PBS Professional Administrator's Guide.

2.17.2.2   Cpusets

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.

Run parallel jobs exclusively within a cpuset for repeatability of performance. SGI states, "Using cpusets on an Altix system improves cached locality and memory access times and can substantially improve an application's performance and runtime repeatability."

The `CPUSET_CPU_EXCLUSIVE` flag will prevent CPU 0 from being used by the MOM in the creation of job cpusets. This flag is set by default, so this is the default behavior.

To find out which cpuset is assigned to a running job, use `qstat -f` to see the cpuset field in the job's altid attribute.

2.17.2.2.i   Altix Running Supported Versions of ProPack or Performance Suite

The cpusets created for jobs are marked cpu-exclusive.

MOM does not use any CPU which was in use at startup.

A PBS job can run across multiple Altixes that run supported versions of ProPack or Performance Suite.

PBS can run using SGI's MPI (MPT) over InfiniBand. See the PBS Professional Administrator's Guide.

2.17.2.3　Effect on Jobs of Starting MOM

When MOM is started or restarted, her default behavior is to leave any running processes running, but to tell the PBS server to requeue the jobs she manages. MOM tracks the process ID of jobs across restarts.

In order to have all jobs killed and requeued, use the r option when starting or restarting MOM.

In order to leave any running processes running, and not to requeue any jobs, use the p option when starting or restarting MOM.

2.17.3　Options to `pbs_mom`

-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.

Format: Integer

Default: *10 seconds*

-C checkpoint_directory

Specifies the path to the directory where MOM creates job-specific subdirectories used to hold each job's restart files. MOM passes this path to checkpoint and restart scripts. Overrides other checkpoint path specification methods. Any directory specified with the -C option must be owned, readable, writable, and executable by root only (*rwx,---,---*, or *0700*), to protect the security of the restart files. See the -d option to pbs_mom and section 10.3.6.5, "Specifying Checkpoint Path" on page 720 in the PBS Professional Administrator's Guide.

Format: *String*

Default: `PBS_HOME/checkpoint`

-c config_file

MOM will read this alternate 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 PBS_HOME/mom_priv/config. 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 given by $PBS_HOME.

Format: String

-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.

Format: integer port number.

Default: *15002*.

-n nice_val

Specifies the priority for the pbs_mom daemon.

Format: integer.

-N

Specifies that when starting, MOM should not detach from the current session.

-p

Specifies that when starting, MOM should allow any running jobs to continue running, and not have them requeued. This option can be used for single-host jobs only; multi-host jobs cannot be preserved. Cannot be used with the -r option. MOM is not the parent of these jobs.

Altix running ProPack or Performance Suite

The cpuset-enabled pbs_mom will, if given the -p flag, use the existing CPU and memory allocations for the /PBSPro 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 requeue any rerunnable jobs and kill any non-rerunnable jobs that she was tracking, and mark the jobs as terminated. Cannot be used with the -p option. MOM is not the parent of these jobs.

It is not recommended to 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.

Format: integer port number

Default: *15003*

-S server_port

Specifies the number of the TCP port on which pbs_mom initially contact the server.

Format: integer port number

Default: *15001*

-s script_options

This option provides an interface that allows the administrator to add, delete, and display MOM's configuration files. See CONFIGURATION FILES. script_options are used this way:

-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. Example:

```
pbs_mom –s insert <scriptname> <inputfile>
```

-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. Example:

```
pbs_mom –s remove <scriptname>
```

-s show <scriptname>

    Causes the contents of the named script to be printed to standard output. If scriptname does not exist, the operation fails, a diagnostic is presented, and `pbs_mom` exits with a nonzero status. Example:

        pbs_mom –s show <scriptname>

-s list

    Causes `pbs_mom` to list the set of PBS-prefixed and site-defined configuration files in the order in which they will be executed. Example:

        pbs_mom –s list

WINDOWS:

    Under Windows, the -N option must be used, so that `pbs_mom` will start up as a standalone program. For example:

        pbs_mom –N –s insert <scriptname> <inputfile>

    or

        pbs_mom –N –s list

--version

    The `pbs_mom` command returns its PBS version information and exits. This option can only be used alone.

2.17.4   Files and Directories

$PBS_HOME/mom_priv

    Default directory for default configuration files.

$PBS_HOME/mom_priv/config

    MOM's default configuration file.

$PBS_HOME/mom_logs

    Default directory for log files written by MOM.

$PBS_HOME/mom_priv/prologue

    File containing administrative script to be run before job execution.

$PBS_HOME/mom_priv/epilogue

    File containing administrative script to be run after job execution.

2.17.5   Signal Handling

pbs_mom handles the following signals:

SIGHUP

      The `pbs_mom` daemon will reread its configuration files, close and reopen the log file, and reinitialize resource structures.

SIGALRM

      MOM writes a log file entry. See the -a alarm_timeout option.

SIGINT

      The `pbs_mom` daemon exits, leaving all running jobs still running. See the -p option.

SIGKILL

      This signal is not caught. The `pbs_mom` daemon exits immediately.

SIGTERM, SIGXCPU, SIGXFSZ, SIGCPULIM, SIGSHUTDN

      The `pbs_mom` daemon terminates all running children and exits.

SIGPIPE, SIGUSR1, SIGUSR2, SIGINFO

      These are ignored.

All other signals have their default behavior installed.

2.17.6   Exit Status

- Greater than zero if the `pbs_mom` daemon fails to start, if the -s insert option is used with an existing scriptname, or if the administrator attempts to add a script whose name begins with "PBS".

- Greater than zero if the administrator attempts to use the -s remove option on a nonexistent configuration file, or on a configuration file whose name begins with "PBS".

- Greater than zero if the administrator attempts to use the -s show option on a nonexistent script.

2.17.7   See Also

The PBS Professional Administrator's Guide, `pbs_server(8B)`, `pbs_sched(8B)`, `qstat(1B)`, SGI's Altix documentation

2.18   pbs_mom_globus

PBS no longer supports Globus. The Globus functionality has been **removed** from PBS.

2.19   pbs_mpihp

Runs an MPI application in a PBS job with HP MPI

2.19.1   Synopsis

pbs_mpihp [-np #] [-h host] [other HP mpirun options] program [args]

pbs_mpihp [HP mpirun options] -f appfile [-- [<extra_args>]]

pbs_mpihp --version

2.19.2   Description

The PBS command `pbs_mpihp` replaces the standard `mpirun` command in a PBS HP MPI job, for executing programs.

`pbs_mpihp` is a front end to the HP MPI version of `mpirun`. It is for PBS jobs running under Linux 2.4 and higher. `pbs_mpihp` has the same usage as `mpirun`. When `pbs_mpihp` is invoked from a PBS job, it will process the command line arguments, then call standard HP `mpirun` to actually start the MPI ranks. The ranks created will be mapped onto cpus on the nodes allocated to the PBS job. The environment variable `MPI_REMSH` will be set to `$PBS_EXEC/bin/pbs_tmrsh`. This will cause the processes that are created to become part of the PBS job.

The path to standard HP `mpirun` is found by checking to see if a link exists with the name `PBS_EXEC/etc/pbs_mpihp`. If this link exists, it will point to standard HP `mpirun`. If it does not exist, a call to `mpirun -version` will be made to determine if it is HP `mpirun`. If so, the call will be made to "`mpirun`" without an absolute path. If HP `mpirun` cannot be found, an error will be output, all temp files will be cleaned up and the script will exit with value 127.

If `pbs_mpihp` is invoked from outside a PBS job, it will pass all of its arguments directly to standard HP `mpirun` without further processing.

The first form above allows one executable to be specified. The second form above uses an appfile to list multiple executables. The format is described in the HP `mpirun` man page. If this form is used from inside a PBS job, the file will be read to determine what executables are to be run and how many processes will be started for each.

When HP MPI is wrapped with `pbs_mpihp`, "`rsh`" is the default used to start the mpids. If you wish to use "`ssh`" or something else, be sure to set the following in `$PBS_HOME/pbs_environment`:

    PBS_RSHCOMMAND=ssh

or put the following in the job script:

```
export PBS_RSHCOMMAND=<rsh_cmd>
```

Executing pbs_mpihp with the -client option is not supported under PBS.

2.19.3   Usage

Usage is the same as for HP mpirun.

2.19.4   Options to `pbs_mpihp`

All options except the following are passed directly to HP mpirun with no modification.

-client

> Not supported.

-np number

> Specifies the number of processes to run on the PBS nodes.

-h host

> Ignored by pbs_mpihp.

-l user

> Ignored by pbs_mpihp.

-f appfile

> The specified appfile is read by pbs_mpihp.

--version

> The pbs_mpihp command returns its PBS version information and exits. This option can only be used alone.

2.19.5   See Also

The PBS Professional Administrator's Guide

mpirun(1)

2.20   pbs_mpilam

Runs MPI programs under PBS with LAM MPI

2.20.1   Synopsis

pbs_mpilam [options]

pbs_mpilam --version

2.20.2   Description

The PBS command `pbs_mpilam` replaces the standard `mpirun` command in a PBS LAM MPI job, for executing programs under Linux 2.4 or higher.

Usage is the same as for LAM `mpirun`. All options are passed directly to `mpirun`. If used to run a single program, PBS tracks resource usage and controls all user processes spawned by the program. If used to run multiple programs as specified in an application file (no <where> argument and no -np/-c option), then PBS does not manage the spawned user processes of each program.

If the where argument is not specified, then `pbs_mpilam` will try to run the user's program on all available CPUs using the C keyword.

2.20.3   Options to `pbs_mpilam`

(options)
> The `pbs_mpilam` command uses the same options as `mpirun`.

--version
> The `pbs_mpilam` command returns its PBS version information and exits. This option can only be used alone.

2.20.4   Path

The PATH on remote machines must contain PBS_EXEC/bin.

2.20.5   See Also

The PBS Professional Administrator's Guide

`mpirun(1)`

2.21   pbs_mpirun

Runs MPI programs under PBS with MPICH

2.21.1 Synopsis

pbs_mpirun [options]

pbs_mpirun --version

2.21.2 Description

The PBS command `pbs_mpirun` replaces the standard `mpirun` command in a PBS MPICH job using P4 running under Linux 2.4 and higher. Usage is the same as for `mpirun`, except for the -machinefile option. All other options are passed directly to `mpirun`.

On Windows, this command cannot be used to start job processes or track a job's resource usage.

2.21.3 Options to `pbs_mpirun`

(options)

The options to `pbs_mpirun` are the same as for `mpirun`, except for the -machinefile option. This is generated by `pbs_mpirun`. The user should not attempt to specify -machinefile.

The value for -machinefile is a temporary file created from PBS_NODEFILE in the format:

        hostname-1[:number of processors]

        hostname-2[:number of processors]

        hostname-n[:number of processors]

where if the number of processors is not specified, it is 1. An attempt by the user to specify the -machinefile option will result in a warning saying "`Warning, -machinefile value replaced by PBS`".

The default value for the -np option is the number of entries in PBS_NODEFILE.

--version

The `pbs_mpirun` command returns its PBS version information and exits. This option can only be used alone.

2.21.4 Environment Variables

`pbs_mpirun` modifies P4_RSHCOMMAND and PBS_RSHCOMMAND. Users should not edit these. `pbs_mpirun` copies the value of P4_RSHCOMMAND into PBS_RSHCOMMAND.

2.21.5 Path

The PATH on remote machines must contain `PBS_EXEC/bin`.

2.21.6 See Also

The PBS Professional Administrator's Guide

`mpirun(1)`

2.22 pbs_password

Sets or updates password of a PBS user

2.22.1 Synopsis

pbs_password [-r] [-s server] [-d] [user]

pbs_password --version

2.22.2 Description

The `pbs_password` command is used to set or update the password of a PBS user. The user does not have to have any jobs on the system.

When no options are given to `pbs_password`, the password credential on the default PBS server for the current user, i.e. the user who executes the command, is updated to the prompted password. Any user jobs previously held due to an invalid password are not released.

Available on Windows and supported Linux x86 and x86_64 platforms only.

The `pbs_password` command has no effect on running jobs. Queued jobs use the new password.

The `pbs_password` command does not change the user's password on the current host, only the password that is cached in PBS.

Note that `pbs_password` encrypts the password obtained from the user before sending it to the PBS Server.

2.22.3   Options to `pbs_password`

(no options)

> The user is prompted for a new password. The password credential on the default PBS server for the current user, i.e. the user who executes the command, is updated to the prompted password. Any user jobs previously held due to an invalid password are not released.

-r

> Any user jobs previously held due to an invalid password are released.

-s server

> Allows user to specify server where password will be changed.

-d

> Deletes the password.

user

> The password credential of user user is updated to the prompted password. If user is not the current user, this action is only allowed if one of the following is true:
>
> • The current user is root or admin.
>
> • User user has given the current user explicit access via the ruserok() mechanism, i.e. the hostname of the machine from which the current user is logged in appears in the server's `hosts.equiv` file, or the current user has an entry in user's `HOMEDIR\.rhosts` file.

--version

> The `pbs_password` command returns its PBS version information and exits. This option can only be used alone.

2.22.4   Exit Status

Table 2-2: Exit Status

Exit Status	Meaning
0	Success
-1	single_signon_password_enable not set
-2	Password of user on server failed to be created/updated
-3	Failed to release jobs held due to bad password owned by user on server
-4	Failed to delete password of user on server
-5	Current user not authorized to change password of user

2.22.5   See Also

`qhold(1B), qrls(1B), qselect(1B), ruserok()`

2.23   pbs_probe

Reports PBS diagnostic information

2.23.1   Synopsis

pbs_probe [-f | -v]

pbs_probe --version

2.23.2   Description

The `pbs_probe` command reports post-installation information that is useful for PBS diagnostics. Aside from the direct information that is supplied on the command line, `pbs_probe` uses as the source for basic information the file `/etc/pbs.conf` and the values of any of the following environment variable that may be set in the environment in which `pbs_probe` is run: PBS_CONF_FILE, PBS_HOME, PBS_EXEC, PBS_START_SERVER, PBS_START_MOM, and PBS_START_SCHED.

In order to execute `pbs_probe`, the user must have PBS Operator or Manager privilege.

Used without options, the `pbs_probe` runs in "report" mode. In this mode `pbs_probe` reports on any errors in the PBS infrastructure files that it detects. The problems are categorized, and a list of the problem messages placed in each category are output. Those categories which are empty do not show in the output.

2.23.3 Options to `pbs_probe`

-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.

--version

> The `pbs_probe` command returns its PBS version information and exits. This option can only be used alone.

2.23.4 Standard Error

The `pbs_probe` command will write a diagnostic message to standard error for each error occurrence.

2.23.5 Files

    /etc/pbs.conf /etc/init.d/pbs

2.23.6 See Also

The PBS Professional Administrator's Guide and the following manual pages:
`pbs_server(8B), pbs_sched(8B), pbs_mom(8B).`

2.24   pbs_python

Python interpreter for debugging a hook script from the command line

2.24.1   Synopsys

pbs_python --hook [-e <log_event_mask>] [-i <input_file>] [-L <log_dir>] [-l <log_file>]
    [-o <output_file>] [-r <resourcedef_file>] [<python_script>]

pbs_python --version

2.24.2   Description

The PBS Python interpreter is a wrapper for Python. You can use the PBS Python interpreter for debugging a hook script at the command line by specifying the ––hook option. You can use the ––hook option to get access to the PBS Python module and perform interactive or non-interactive debugging.

Without the ––hook option, you can use pbs_python to run the Python interpreter that is distributed with PBS Professional.

2.24.3   Options to `pbs_python`

--hook

> This option is a switch. When you use this option, you can use the PBS Python module (via "import pbs"), and the other options described here are available. When you use this option, you cannot use the standard Python options. This option is useful for debugging.

> When you do not use this option, you cannot use the other options listed here, but you can use the standard Python options.

-i <input_file>

> Text file containing data to populate pbs.event() objects. Each line specifies an attribute value or a resource value. Syntax of each input line is one of the following:

> *<object_name>.<attribute_name>=<attribute_value>*
> *<object_name>.<resource_list>[<resource_name>]=<resource_value>*
> Where

<object_name> is a PBS object name which can refer to its sub-objects. Examples: *"pbs.event()", "pbs.event().job",* *"pbs.event().vnode_list["<vnode name>"]"*.

Example input file:

```
pbs.event().hook_name=proto

pbs.event().hook_type=site

pbs.event().type=queuejob

pbs.event().requestor=user1

pbs.event().requestor_host=host1

pbs.event().hook_alarm=40

pbs.event().job.id=72

pbs.event().job.Job_Name=job1

pbs.event().job.Resource_List[ncpus]=5

pbs.event().job.Resource_List[mem]=6mb

pbs.event().vnode_list["host1"].resources_available["ncpus"] = 5

pbs.event().vnode_list["host1"].resources_available["mem"] =
    300gb
```

Available only when --hook option is used.

-e <log_event_mask>
> Sets the mask that determines which event types are logged by pbs_python. To see only debug messages, set the value to *0xd80*. To see all messages, set the value to *0xffff*. The pbs_python interpreter uses the same set of mask values that are used for the $logevent <mask> entry in the pbs_mom configuration file. See <u>section 2.17, "pbs_mom", on page 58</u>. Available only when --hook option is used.

-L <log_dir>
> Directory holding the log file where pbs.logmsg() and pbs.logjobmsg() write their output. Default is current working directory where pbs_python is executed. Available only when --hook option is used.

-l <log_file>
> Log file where pbs.logmsg() and pbs.logjobmsg() write their output. Default file name is current date in *yyyymmdd* format. Available only when --hook option is used.

-o <output_file>

    The output file contains the changes made after executing the hook script, such as the attributes and resources set in any pbs.event() jobs and reservations, whether to accept or reject an action, and any pbs.reject() messages.

    Example output file:

```
pbs.event().job.Job_Name=job2
pbs.event().job.Resource_List[file]=60gb
pbs.event().job.Resource_List[ncpus]=5
pbs.event().job.Resource_List[mem]=20gb
pbs.event().job.Account_Name=account2
pbs.event().reject=True
pbs.event().reject_msg=No way!
```

    Without this option, output goes to stdout. Available only when --hook option is used.

-r <resourcedef>

    File/path name containing a resource definition specifying a custom resource whose Python type is pbs.resource. Format:

    <resource_name>type=<typename> [flag=<value>]

    This file has the same format as the PBS_HOME/server_priv/resourcedef file. Available only when --hook option is used.

--version

    The pbs_python command prints its version information and exits. This option can only be used alone.

2.24.4   Arguments

<python_script>

    The hook script to execute. We recommend importing the PBS Python module at the start of the script:

```
import pbs
```

    If you do not specify *<python script>*, you can perform interactive debugging. If you type the following:

    % pbs_python --hook -i hook.input

    The interpreter displays a prompt:

    >>

You can type your Python lines at the prompt:

```
>>import pbs
>> e=pbs.event().job
>> print e.id
<job-id>
...
```

2.25   **pbs_rdel**

Deletes a PBS advance or standing reservation

2.25.1   **Synopsis**

pbs_rdel reservation_identifier[,reservation_identifier...]

pbs_rdel --version

2.25.2   **Description**

The `pbs_rdel` command deletes reservations in the order in which their reservation identifiers are presented to the command.

A reservation may be deleted by its owner, the PBS Operator, or the PBS Manager.

2.25.3   **Options**

--version

> The `pbs_rdel` command returns its PBS version information and exits. This option can only be used alone.

2.25.4   **Operands**

The `pbs_rdel` command accepts one or more reservation_identifier operands.

For an advance reservation this has the form:

[R]sequence_number[.server_name][@remote_server]

For a standing reservation this has the form:

[S]sequence_number[.server_name][@remote_server]

@remote_server is used to specify a reservation at a server other than the default server.

2.25.5   Exit Status

Zero upon success.

Greater than zero upon failure to process any operand.

2.25.6   See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide, `pbs_rsub(1B)`, `pbs_rstat(1B)`, `pbs_resv_attributes(7B)`

2.26   pbs_renew

Renews Kerberos credential

2.26.1   Synopsis

pbs_renew [-d] program [arg(s)]

pbs_renew --version

2.26.2   Description

The `pbs_renew` command is used internally by PBS when a job has a Kerberos credential. The program is run as a child process with any arguments passed to the command line of program. The `pbs_renew` process runs periodically to renew any Kerberos credential. It will wait for the child process to return, clean up any Kerberos credential and exit when the child process is done.

2.26.3   Options

-d
: Debug messages are printed to stderr.

--version

>>> The `pbs_renew` command returns its PBS version information and exits. This option can only be used alone.

2.26.4   See Also

The PBS Professional Administrator's Guide, `qsub(1B)`

2.27   pbs_rstat

Shows status of PBS advance or standing reservations

2.27.1   Synopsis

pbs_rstat [-F][-B][-S] [reservation_id...]

pbs_rstat --version

2.27.2   Description

The `pbs_rstat` command is used to show the status of all reservations on the PBS Server. Denied reservations are not displayed.

This command has three different output formats: brief (B), short (S), and full (F). This command can be used with any level of PBS privilege.

See the pbs_resv_attributes(7B) man page for information about reservation attributes.

2.27.3   Options to `pbs_rstat`

-B Brief

>>> Displays each reservation identifier only.

-S Short

>>> Displays a table showing the name, queue, owner, state, start time, duration, and end time of each reservation.

-F Full

>>> Displays all reservation attributes that are not set to the default value. Users without manager or operator privilege cannot print custom resources which were created to be invisible to users.

--version

The pbs_rstat command returns its PBS version information and exits. This option can only be used alone.

2.27.4   Output

See <u>section 8.6, "Reservation States", on page 419</u>.

2.27.5   Operands

The pbs_rstat command accepts one or more reservation_identifier operands.

2.27.5.1   Reservations at the default server

For an advance reservation, the reservation_identifier has the form:

[R]sequence_number[.server_name]

For a standing reservation, the reservation_identifier has the form:

[S]sequence_number[.server_name]

2.27.5.2   Reservations at a server other than the default server:

Specify the remote server's name using @remote_server.

For an advance reservation:

[R]sequence_number[.server_name][@remote_server]

For a standing reservation:

[S]sequence_number[.server_name][@remote_server]

2.27.6   See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide, pbs_rsub(1B), pbs_rdel(1B), pbs_resv_attributes(7B)

2.28   pbs_rsub

Creates a PBS advance or standing reservation

2.28.1   Synopsis

pbs_rsub [-D duration] [-E end_time] [-g group_list] [-G auth_group_list] [-H auth_host_list] [-I seconds] [-m mail_points] [-M mail_list] [-N reservation_name] [-q destination] [-r recurrence_rule] [-R start_time] [-u user_list] [-U auth_user_list] [-W attribute_value_list] -l resource_request [-l placement]

pbs_rsub --version

2.28.2   Description

The `pbs_rsub` command is used to create an advance or standing reservation. An advance reservation reserves specific resources for the requested time period, and a standing reservation reserves specific resources for recurring time periods. When a reservation is created, it has an associated queue.

After the reservation is requested, it is either confirmed or denied. Once the reservation has been confirmed, authorized users submit jobs to the reservation's queue via `qsub` and `qmove`.

A confirmed reservation will accept jobs at any time. The jobs in its queue can run only during the reservation period, whether during a single advance reservation or during the occurrences of a standing reservation.

When an advance reservation ends, all of its jobs are deleted, whether running or queued. When an occurrence of a standing reservation ends, only its running jobs are deleted; those jobs still in the queue are not deleted.

To get information about a reservation, use the `pbs_rstat` command.

To delete a reservation, use the `pbs_rdel` command. Do not use the `qdel` command.

The behavior of the `pbs_rsub` command may be affected by any site hooks. Site hooks can modify the reservation's attributes.

2.28.3   Requirements

When using `pbs_rsub` to request a reservation, the user must specify two of the following options: -R, -E, and -D. The resource request -l walltime can be used instead of the -D option.

2.28.4 Options to `pbs_rsub`

-D duration

Specifies reservation duration. If the start time and end time are the only times specified, this duration time is calculated.

Format: *Duration*

Default: none

-E end_time

Specifies the reservation end time. If start time and duration are the only times specified, the end time value is calculated.

Format: *Datetime.*

Default: none

-g group_list

The group_list is a comma-separated list of group names. The server uses entries on this list, along with an ordered set of rules, to associate a group name with the reservation.

Format: *group@hostname[,group@hostname ...]*

-G auth_group_list

Comma-separated list of names of groups who can or cannot submit jobs to this reservation. Group names are interpreted in the context of the server's host, not the context of the host from which the job is submitted.

This list becomes the acl_groups list for the reservation's queue. More specific entries should be listed before more general, because the list is read left-to-right, and the first match determines access.

If both Authorized_Users and Authorized_Groups are set, a user must belong to both in order to be able to submit jobs to this reservation. If the reservation creator specifies this list, the creator's group is not automatically added to the list.

Refer to the Authorized_Groups reservation attribute on the `pbs_resv_attributes(7B)` man page.

Format: *[+|-]group_name[,[+|-]group_name ...]*

Default: All groups are authorized to submit jobs.

-H auth_host_list

Comma-separated list of hosts from which jobs can and cannot be submitted to this reservation. This list becomes the acl_hosts list for the reservation's queue. More specific entries should be listed before more general, because the list is read left-to-right, and the first match determines access.

If the reservation creator specifies this list, the creator's host is not automatically added to the list.

See the Authorized_Hosts reservation attribute on the
`pbs_resv_attributes(7B)` man page.

Format: *[+|-]hostname[,[+|-]hostname ...]*

Default: All hosts are authorized to submit jobs.

-l block_time

Specifies interactive mode. The `pbs_rsub` command will block, up to block_time seconds, while waiting for the reservation request to be confirmed or denied.

If block_time is positive, and the reservation isn't confirmed or denied in the specified time, the ID string for the reservation is returned with the status "UNCONFIRMED".

If block_time is negative, and the scheduler doesn't confirm or deny the reservation in the specified time, the reservation is deleted.

Format: *Integer.*

Default: *Not interactive.*

-l placement

The placement specifies how vnodes are reserved. The place statement has this form:

-l place=[arrangement][: sharing][: grouping]

where

arrangement is one of *free | pack | scatter | vscatter*

sharing is one of *excl | share | exclhost*

grouping can have only one instance of *group=resource*

and where

free:

Use any vnode(s) for reservation.

pack:

All chunks are taken from one host.

scatter:

Only one chunk with any MPI processes will be taken from a host. A chunk with no MPI processes may be taken from the same node as another chunk.

vscatter:

Only one chunk is used in any vnode.

excl:

Only this reservation uses the vnodes chosen.

exclhost:

The entire host is allocated to the reservation.

share:

This reservation can share the chosen vnodes.

group=resource:

Chunks will be grouped according to a resource. All nodes in the group must have a common value for the resource, which can be either the built-in resource host or a site-defined node-level resource.

Note that nodes can have sharing attributes that override reservation placement requests.

See the `pbs_node_attributes(7B)` man page.

-l resource_request

The resource_request specifies the resources required for the reservation. These resources will be used for the limits on the queue that is dynamically created for the reservation. The aggregate amount of resources for currently running jobs from this queue will not exceed these resource limits. Jobs in the queue that request more of a resource than the queue limit for that resource are not allowed to run. Also, the queue inherits the value of any resource limit set on the server, and these are used for the job if the reservation request itself is silent about that resource. A non-privileged user cannot submit a reservation requesting a custom resource which has been created to be invisible or read-only for users.

Resources are requested by using the -l option, either in chunks inside of selection statements, or in job-wide requests using *resource_name=value* pairs. The selection statement is of the form:

-l select=[N:]chunk[+[N:]chunk ...]

where *N* specifies how many of that chunk, and a chunk is of the form:

resource_name=value[:resource_name=value ...]

Job-wide *resource_name=value* requests are of the form:

-l resource_name=value[,resource_name=value ...]

-m mail_points

Specifies the set of events that cause mail to be sent to the list of users specified in the -M mail_list option.

Format: string consisting of 1) any combination of "a", "b", "c" or "e", or 2) the single character "n".

Table 2-3: Suboptions to -m Option

Character	Meaning
a	Notify if the reservation is terminated for whatever reason
b	Notify when the reservation period begins
c	Notify when the reservation is confirmed
e	Notify when the reservation period ends
n	Send no mail. Cannot be used with any of a, b, c or e.

Default: "*ac*".

-M mail_list

The list of users to whom mail is sent whenever the reservation transitions to one of the states specified in the -m mail_points option.

Format: *user[@hostname][,user[@hostname]...]*

Default: Reservation owner.

-N reservation_name

This specifies a name for the reservation.

Format: String up to 15 characters in length. It must consist of printable, non-white space characters with the first character alphabetic.

Default: None.

-q destination

Specifies the destination server at which to create the reservation.

Default: The default server is used if this option is not selected.

-r recurrence_rule

Specifies rule for recurrence of standing reservations. Rule must conform to iCalendar syntax, and is specified using a subset of parameters from RFC 2445.

Valid syntax for the recurrence_rule takes one of two forms:

FREQ= freq_spec; COUNT= count_spec; interval_spec

or

FREQ= freq_spec; UNTIL= until_spec; interval_spec

where

freq_spec

> Frequency with which the standing reservation repeats. Valid values are:

> *WEEKLY|DAILY|HOURLY*

count_spec

> The exact number of occurrences. Number up to 4 digits in length.

> Integer.

interval_spec

> Specifies interval.

> Format is one or both of:

> *BYDAY = MO|TU|WE|TH|FR|SA|SU*

> or

> *BYHOUR = 0|1|2|...|23*

> When using both, separate them with a semicolon.

> Elements specified in the recurrence rule override those specified in the arguments to the -R and -E options. ,For example, the BYHOUR specification overrides the hourly part of the -R option. For example, `-R 0730 -E 0830 ... BYHOUR=9` results in a reservation that starts at 9:30 and runs for 1 hour.

until_spec

> Occurrences will start up to but not after date and time specified. Format:

> *YYYYMMDD[THHMMSS]*

> Note that the year-month-day section is separated from the hour-minute-second section by a capital T.

Requirements:

- The recurrence rule must be on one unbroken line and must be enclosed in double quotes.

- A start and end date must be used when specifying a recurrence rule. See the R and E options.

- The PBS_TZID environment variable must be set at the submission host. The format for PBS_TZID is a timezone location. Examples: `America/Los_Angeles, America/Detroit, Europe/`

Berlin, Asia/Calcutta. See the PBS Professional User's
Guide.

Examples of Standing Reservations

For a reservation that runs every day from 8am to 10am, for a total of 10
occurrences:

pbs_rsub -R 0800 -E 1000 -r "FREQ=DAILY;COUNT=10"

Every weekday from 6am to 6pm until December 10 2008

**pbs_rsub -R 0600 -E 1800 -r "FREQ=WEEKLY; BYDAY=MO,TU,WE,TH,FR;
    UNTIL=20081210"**

Every week from 3pm to 5pm on Monday, Wednesday, and Friday, for 9
occurrences, i.e., for three weeks:

**pbs_rsub -R 1500 -E 1700 -r "FREQ=WEEKLY;BYDAY=MO,WE,FR;
    COUNT=3"**

-R start_time

Specifies reservation starting time. If the reservation's end time and dura-
tion are the only times specified, this start time is calculated.

If the day, DD , is not specified, it defaults to today if the time hhmm is in
the future. Otherwise, the day is set to tomorrow. For example, if you sub-
mit a reservation with the specification -R 1110 at 11:15 a.m., it is inter-
preted as being for 11:10am tomorrow. If the month portion, MM , is not
specified, it defaults to the current month, provided that the specified day
DD , is in the future. Otherwise, the month is set to next month. Similar
rules apply to the two other optional, left-side components.

Format: *Datetime*

-u user_list

Comma-separated list of user names. Not used. Refer to the User_List
reservation attribute on the pbs_resv_attributes(7B) man page.

Format: *user[@host][,user[@host] ...]*

Default: None.

-U auth_user_list

Comma-separated list of users who are and are not allowed to submit jobs
to this reservation. This list becomes the acl_users attribute for the reser-
vation's queue. More specific entries should be listed before more general,
because the list is read left-to-right, and the first match determines access.

If both Authorized_Users and Authorized_Groups are set, a user must
belong to both in order to be able to submit jobs to this reservation. The

reservation creator's username is automatically added to this list, whether or not the reservation creator specifies this list.

Refer to the Authorized_Users reservation attribute on the `pbs_resv_attributes(7B)` man page.

Format: *[+|-]user@host[,[+|-]user@host...]*

Default: Job owner only.

-W attribute_value_list

This allows you to define other attributes for the reservation.

Supported attributes:

qmove=jobid

Converts a normal job designated by *jobid* into a reservation job that will run as soon as possible. Creates the reservation with its queue and moves the job into the reservation's queue. Uses the resources requested by the job to create the reservation.

When the reservation is created, it inherits its resources from the job, not from the resources requested through the `pbs_rsub` command.

If the qmove option is used and the reservation is not confirmed within the timeout period, the reservation is deleted. The default timeout period is 10 seconds. There is no option for this kind of reservation to be unconfirmed.

To specify the timeout, give a negative value for the −I option. For example, to specify a timeout of 300 seconds:

    `pbs_rsub −Wqmove=<job ID> −I −300`

The -R and -E options to `pbs_rsub` are disabled when using the qmove=jobid attribute.

Some shells require that you enclose a job array ID in double quotes.

Timeout must be specified with a negative number.

--version

The `pbs_rsub` command returns its PBS version information and exits. This option can only be used alone.

2.28.5   Output

The `pbs_rsub` command returns the reservation name.

For an advance reservation, this has the form

RNNNN.server

where *NNNN* is a unique integer. The associated queue's name is the prefix, *RNNNN*.

For a standing reservation, this has the form

SNNNN.server

where *NNNN* is a unique integer. The associated queue's name is the prefix, *SNNNN*.

2.28.6   See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide,
`pbs_resv_attributes(7B), pbs_rdel(1B), pbs_rstat(1B),`
`qmove(1B),  qsub(1B)`

2.29   pbs_sched

Runs the PBS scheduler

2.29.1   Synopsis

pbs_sched [-a alarm] [-c clientsfile] [-d home] [-L logfile] [-n] [-N] [-p file] [-R port] [-S
   port]

pbs_sched --version

2.29.2   Description

`pbs_sched` is the PBS scheduling daemon. It schedules PBS jobs.

`pbs_sched` must be executed with root permission.

2.29.3   Options to `pbs_sched`

> -a alarm
>> Deprecated. Will overwrite value of sched_cycle_length scheduler
>> attribute. Time in seconds to wait for a scheduling cycle to finish.
>>
>> Format: Time, in seconds.
>
> -c clientsfile
>> Add clients to the scheduler's list of known clients. The clientsfile contains
>> single-line entries of the form
>>
>>> `$clienthost <hostname>`

Each hostname is added to the list of hosts allowed to connect to the scheduler. If clientsfile cannot be opened, the scheduler aborts. Path can be absolute or relative. If relative, it is relative to `PBS_HOME/sched_priv/`.

-d home

The directory in which the scheduler will run.

Default: `PBS_HOME/sched_priv`.

-L logfile

The absolute path and filename of the log file. The scheduler writes its PBS version and build information to the logfile whenever it starts up or the logfile is rolled to a new file.

See the -d option.

Default: The scheduler will open a file named for the current date in the `PBS_HOME/sched_logs` directory.

-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 starting.

-N

Instructs the scheduler not to detach itself from the current session.

-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`.

See the -d option.

Default: `PBS_HOME/sched_priv/sched_out`.

-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 for the PBS scheduler is taken from PBS_SCHEDULER_SERVICE_PORT, in `pbs.conf`.

Default: *15004*.

--version

> The pbs_sched command returns its PBS version information and exits. This option can only be used alone.

2.29.4 Signal Handling

SIGHUP

> The scheduler will close and reopen its log file and reread the config file if one exists.

SIGALRM

> If the scheduler exceeds the time limit, the alarm will cause the scheduler to attempt to core dump and restart itself.

SIGINT and SIGTERM

> Will result in an orderly shutdown of the scheduler.

All other signals have the default action installed.

2.29.5 Exit Status

Zero upon normal termination.

2.29.6 See Also

The PBS Professional Administrator's Guide, pbs_server(8B), pbs_mom(8B)

2.30 pbs_server

Starts a PBS batch server

2.30.1 Synopsis

pbs_server [-a active] [-A acctfile] [-C] [-d config_path] [-e mask] [-F seconds] [-L logfile] [-M mom_port] [-N] [-p port] [-R momRPP_port] [-S scheduler_port] [-t type]

pbs_server --version

2.30.2   Description

The `pbs_server` command starts the operation of a batch server on the local host. Typically, this command will be in a local boot file such as `/etc/rc.local` . If the batch server is already in execution, `pbs_server` will exit with an error. To insure that the `pbs_server` command is not runnable by the general user community, the server will only execute if its real and effective UID is zero.

The server will record a diagnostic message in a log file for any error occurrence. The log files are maintained in the server_logs directory below the home directory of the server. If the log file cannot be opened, the diagnostic message is written to the system console. The server writes its PBS version and build information to the logfile whenever it starts up or the logfile is rolled to a new file.

To kill the server:

UNIX/Linux:

>   **qterm (see qterm(8B))**

>   or

>   **"kill <server_pid>", which sends a SIGTERM.**

Windows:

if you're running "`pbs_server -N`" for a standalone mode server, use

>   **<cntrl>-<break>.**

2.30.3   Options to `pbs_server`

-a <value>

>   When *True*, the server is in state "*active*" and the scheduler is called to schedule jobs. When *False*, the server is in state "*idle*" and the scheduler is not called to schedule jobs. Sets the server's scheduling attribute. If the `-a T|F` option is not specified, the server uses the prior value for the scheduling attribute.

>   Format: Boolean

-A acctfile

>   Specifies an absolute path name for 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.

-C

>   The server starts up, creates the database, and exits. Windows only.

-d config_path

>   Specifies the path of the directory which is home to the servers configura-
>   tion files, PBS_HOME. A host may have multiple servers. Each server
>   must have a different configuration directory. The default configuration
>   directory is given by the symbol $PBS_HOME which is typically /usr/
>   spool/PBS.

-e mask

>   Specifies a log event mask to be used when logging. See "log_events" in
>   the pbs_server_attributes(7B) man page.

-F seconds

>   Specifies the number of seconds that the secondary server should wait
>   before taking over when it believes the primary server is down. If the num-
>   ber of seconds is specified as *-1*, the secondary will make one attempt to
>   contact the primary and then become active.

>   Default: *30 seconds*

-L logfile

>   Specifies an absolute path name of the file to use as the log file. If not spec-
>   ified, 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 con-
>   nect the job executor, MOM. The option argument, mom_conn, is one of
>   the forms:

>   *host_name, [:]port_number*

>   or

>   *host_name:port_number*

>   If host_name not specified, the local host is assumed. If port_number is not
>   specified, the default port is assumed.

>   See the -M option for pbs_mom(8B).

>   Default: *15002*

-N

>   The server runs in standalone mode. In Windows, it does not register as a
>   Windows service. On other platforms, MOM will not detach from the cur-
>   rent session.

-p port

> Specifies the port number on which the server will listen for batch requests. If multiple servers are running on a single host, each must have its own unique port number. This option is for use in testing with multiple batch systems on a single host.

> Default: *15001*

-R mom_RPPport

> Specifies the port number on which the server should query the up/down status of Mom. See the -R option for pbs_mom(8B).

> Default: *15003*

-s replacement_string

> Specifies the string to use when replacing spaces in accounting entity names. Only available under Windows.

> See the PBS Professional Administrator's Guide.

-S scheduler_port

> Specifies the port number to which the server should connect when contacting the Scheduler. The option argument, scheduler_conn, is of the same syntax as under the -M option.

> Default: *15004*

-t type

> Specifies behavior when the server restarts. The type argument is one of the following:

> cold
>> All jobs are purged. Positive confirmation is required before this direction is accepted.

> create
>> The server will discard any existing configuration files: server, nodes, queues and jobs, and initialize configuration files to the default values. The server is idled (scheduling set *False*).

> 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 node being down, the server will attempt to restart it periodically for up to 5 minutes. After that period, the server will revert to a normal state, as if warm started, and will no longer attempt to restart any remaining jobs which were running prior to the shutdown.

updatedb

Updates format of PBS data from the previous format to the data service format.

warm

All jobs in the *Running* state are retained in that state. All other jobs are maintained in their current state. The job scheduler will typically make new selections for which jobs are placed into execution. warm is the default if -t is not specified.

--version

The pbs_server command returns its PBS version information and exits. This option can only be used alone.

2.30.4   Files

$PBS_HOME/server_priv

default directory for configuration files.

$PBS_HOME/server_logs

directory for log files recorded by the server.

2.30.5   Signal Handling

On receipt of the following signals, the server performs the defined action:

SIGHUP

The current server log and accounting log are closed and reopened. This allows for the prior log to be renamed and a new log started from the time of the signal.

SIGTERM

Causes a rapid orderly shutdown of pbs_server, identical to "qterm -t quick".

SIGSHUTDN

On systems (Unicos) where SIGSHUTDN is defined, it also causes an orderly "quick" shutdown of the server.

SIGPIPE, SIGUSR1, SIGUSR2
> These signals are ignored.

All other signals have their default behavior installed.

2.30.6   Exit Status

If the server command fails to begin batch operation, the server exits with a value greater than zero.

2.30.7   See Also

The PBS Professional Administrator's Guide and the following manual pages: `qsub (1B)`, `pbs_connect(3B)`, `pbs_mom(8B)`, `pbs_sched(8B)`, `pbsnodes(8B)`, `qdisable(8B)`, `qenable(8B)`, `qmgr(8B)`, `qrun(8B)`, `qstart(8B)`, `qstop(8B)`, and `qterm(8B)`

2.31   pbs_tclsh

TCL shell with TCL-wrapped PBS API

2.31.1   Synopsis

pbs_tclsh

pbs_tclsh -version

2.31.2   Description

The `pbs_tclsh` is a version of the TCL shell which includes wrapped versions of the PBS external API. The PBS TCL API is documented in the `pbs_tclapi (3B)` manual page.

Root privilege is required in order to query MOM for dynamic resources. Root privilege is not required in order to query MOM for built-in resources and site-defined static resources.

_____

The `pbs_tclsh` command is used to query MOM. For example:

> **pbs_tclsh**

tclsh> openrm <hostname>

<file descriptor>

tclsh> addreq <file descriptor> "loadave"

tclsh> getreq <file descriptor>

<load average>

tclsh> closereq <file descriptor>

2.31.3   Options

--version

> The `pbs_tclsh` command returns its PBS version information and exits. This option can only be used alone.

2.31.4   Standard Error

The `pbs_tclsh` command will write a diagnostic message to standard error for each error occurrence.

2.31.5   See Also

The PBS Professional Administrator's Guide, the PBS Programmer's Guide, and the following manual pages: `pbs_wish(8B)`, `pbs_server(8B)`, `pbs_mom(8B)`, `pbs_sched(8B)`

2.32   pbs_tmrsh

TM-enabled replacement for `rsh/ssh` for use by MPI implementations

2.32.1   Synopsis

pbs_tmrsh host [-l username] [-n] command [args ...]

pbs_tmrsh --version

2.32.2   Description

The `pbs_tmrsh` command attempts to emulate an "`rsh`" connection to the specified host, via underlying calls to the Task Management (TM) API. The program is intended to be used during MPI integration activities, and not by end-users. The initial version of this program is targeted for use with MPICH and HP-MPI.

Running "`pbs_tmrsh host command`" will cause a PBS task to be started on "host" running "command". The "host" may be in IP dot address form.

The environment variables used by the two MPI implementations to point to the `rsh` work-alike (MPI_REMSH in the case of HP and P4_RSHCOMMAND for MPICH) must be set in the job environment and point to the full path for `pbs_tmrsh`.

The file `$PBS_HOME/pbs_environment` will be used to set an environment variable PATH to be used to search for the program executable. This applies to both Windows and UNIX. It is expected that a full path will be specified for the command and the PATH variable will not be needed.

Output and errors are written to the PBS job's output and error files, not to standard output/error.

2.32.3   Options

-l username
> Specifies the username under which to execute the task. If used, username must match the username running the `pbs_tmrsh` command.

-n
> Currently a no-op; provided for MPI implementations that expect to call `rsh` with the "-n" option.

--version
> The `pbs_tmrsh` command returns its PBS version information and exits. This option can only be used alone.

2.32.4   Standard Error

The `pbs_tmrsh` command will write a diagnostic message to the PBS job's error file for each error occurrence.

2.32.5   Exit Status

The pbs_tmrsh program will exit with the exit status of the remote command or with *255* if an error occurred. This is because ssh works this way.

2.32.6   See Also

The PBS Professional Administrator's Guide and the following manual pages:
pbs_attach(8B), tm(3)

2.33   pbs_topologyinfo

Reports topological information about vnodes

2.33.1   Synopsis

pbs_topologyinfo [-a][-s]

pbs_topologyinfo [-s <vnode name> [<vnode name> ...]]

pbs_topologyinfo [-h]

2.33.2   Description

The pbs_topologyinfo command reports topological information for one or more vnodes. To use the command, you must specify what kind of topological information you want. The command reports only the requested information.

pbs_topologyinfo -a -s reports socket counts for all vnodes that have reported sockets.

pbs_topologyinfo -s <vnode name> reports socket count for vnode <vnode name>.

This command is not supported on Windows.

This command must be run on the server host.

2.33.3   Permissions for **pbs_topologyinfo**

This command can be run only by root.

2.33.4 Options for `pbs_topologyinfo`

-a, --all

> Reports requested topological information for all vnodes. When this option is used alone, the command does not report any information.

-s, --sockets

> This option specifies socket count information. The command reports derived socket counts.

-h, --help

> Prints usage and exits.

2.33.5 Operands

<vnode name>

> Name of vnode about which to report.

2.33.6 Standard Error

0

> Success

1

> Any error following successful command line processing.
>
> If an invalid vnode name is specified, a message is printed to standard error.

2.33.7 See Also

The PBS Professional Administrator's Guide

2.34 pbs_wish

TK window shell with TCL-wrapped PBS API

_____

2.34.1   Synopsis

pbs_wish

pbs_wish --version

2.34.2   Description

The `pbs_wish` command is a version of the TK window shell which includes wrapped versions of the PBS external API. The PBS TCL API is documented in the `pbs_tclapi(3B)` manual page.

2.34.3   Options

--version

The `pbs_wish` command returns its PBS version information and exits. This option can only be used alone.

2.34.4   Standard Error

The `pbs_wish` command will write a diagnostic message to standard error for each error occurrence.

2.34.5   See Also

The PBS Professional Administrator's Guide and the following manual pages:
`pbs_tclsh(8B), pbs_mom(8B), pbs_server(8B), pbs_sched(8B)`

2.35   pbsdsh

Distributes task(s) to vnodes under PBS

2.35.1   Synopsis

pbsdsh [-c <copies>] [-s] [-v] [-o] -- program [program_args]

pbsdsh [-n <node_index>] [-s] [-v] [-o] -- program [program_args]

pbsdsh --version

2.35.2   Description of `pbsdsh` Command

The `pbsdsh` command allows you to distribute and execute a task on each of the vnodes assigned to your job by executing (spawning) the application on each vnode. The `pbsdsh` command uses the PBS Task Manager, or TM, to distribute the program on the allocated vnodes.

When run without the -c or the -n option, `pbsdsh` will spawn the program on all vnodes allocated to the PBS job. The spawns take place concurrently; all execute at (about) the same time.

Note that the double dash must come after the options and before the program and arguments. The double dash is only required for Linux.

The following example shows the `pbsdsh` command inside of a PBS batch job. The options indicate that the user wants `pbsdsh` to run the *myapp* program with one argument (*app-arg1*) on all four vnodes allocated to the job (i.e. the default behavior).

```
#!/bin/sh
#PBS -l select=4:ncpus=1
#PBS -l walltime=1:00:00


pbsdsh ./myapp app-arg1
```

The `pbsdsh` command runs one task for each line in the PBS_NODEFILE. Each MPI rank will get a single line in the PBS_NODEFILE, so if you are running multiple MPI ranks on the same host, you will still get multiple `pbsdsh` tasks on that host.

The `pbsdsh` command is not supported on Windows.

2.35.3   Options to `pbsdsh` Command

-c <copies>

> The program is spawned copies times on the vnodes allocated, one per vnode, unless copies is greater than the number of vnodes. If copies is greater than the number of vnodes, it will wrap around, running multiple instances on some vnodes. This option is mutually exclusive with -n.

-n <node_index>

> The program is spawned only on a single vnode, which is the node_index -th vnode allocated. This option is mutually exclusive with -c.

-o

> No obit request is made for spawned tasks. The program will not wait for the tasks to finish.

-s

The program is run in turn on each vnode, one after the other.

-v

Produces verbose output about error conditions and task exit status.

--version

The `pbsdsh` command returns its PBS version information and exits. This option can only be used alone

2.35.4　　Operands

program

The first operand, `program` , is the program to execute. The double dash must precede the program under Linux.

program_args

Additional operands, `program_args` , are passed as arguments to the program.

2.35.5　　Standard Error

The `pbsdsh` command writes a diagnostic message to standard error for each error occurrence.

2.35.6　　See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide, `qsub(1B), tm(3)`

2.36　　**pbsfs**

Shows or manipulates PBS fairshare usage data

2.36.1 Synopsis

pbsfs -[t|p]

pbsfs -g entity

pbsfs -s entity usage_value

pbsfs -d

pbsfs -e

pbsfs -c entity1 entity2

pbsfs --version

2.36.2 Description

The `pbsfs` command is used to print or manipulate the PBS scheduler's fairshare usage data. Some options should only be used when the scheduler is not running. There are multiple parts to a fairshare node and you can print these data in different formats. The `pbsfs` command must be run by root; otherwise it will print the error message, "`Unable to access fairshare data`".

The data:

fairshare entity
>   The entity in the fairshare tree.

group
>   The group ID the node is in (i.e. the node's parent).

cgroup
>   The group ID of this group

shares
>   The number of shares the group has

usage
>   The amount of usage

percentage
>   The percentage the entity has of the tree. Note that only the leaf nodes sum to 100%. If all of the nodes are summed, the result will be greater then 100%. Only the leaf nodes of the tree are fairshare entities.

usage / perc
>   The value the scheduler will use to the pick which entity has priority over another. The smaller the number the higher the priority.

Path from root

> The path from the root of the tree to the node. This is useful because the scheduler will look down the path to compare two nodes to see which has the higher priority.

resource

> The resource for which the scheduler accumulates usage for its fairshare calculations. This defaults to cput (CPU seconds) but can be set in the scheduler's config file.

2.36.3   Options to `pbsfs`

Scheduler can be running or down:

-t

> Print the fairshare tree in a hierarchical format.

-p

> Print the fairshare tree in a flat format with more data.

-g entity

> Print one entry with all data and print the path from the root of the tree to the node.

-c entity1 entity2

> Compare two fairshare entities

--version

> The `pbsfs` command returns its PBS version information and exits. This option can only be used alone.

Scheduler must be down:

-s entity usage_value

> Set entity's usage value to usage_value. Please note that editing a non-leaf node is ignored. All non-leaf node usage values are calculated each time the scheduler is run/HUPed.

-d

> Decay the fairshare tree (divide all values in half)

-e

> Trim fairshare tree to just the entities in the resource_group file

2.36.4   See Also

The PBS Professional Administrator's Guide, `pbs_sched(8B)`

2.37   pbsnodes

Queries PBS host status or mark hosts free or offline

2.37.1   Synopsis

pbsnodes [-o | -r] [-s server] hostname [hostname ...]

pbsnodes [-l] [-s server]

pbsnodes -a [-v] [-s server]

pbsnodes --version

2.37.2   Description

The `pbsnodes` command is used to query the status of hosts, or to mark hosts *FREE* or *OFFLINE*. The `pbsnodes` command obtains host information by sending a request to the PBS server.

To print the status of the specified host or hosts, run `pbsnodes` with no options (except the `-s` option) and with a list of hosts.

To print the command usage, run `pbsnodes` with no options and without a list of hosts.

PBS Manager or Operator privilege is required to execute `pbsnodes` with the `-o` , or `-r` options.

To remove a node from the scheduling pool, mark it *OFFLINE*. If it is marked *DOWN*, when the server next queries the MOM, and can connect, the node will be marked *FREE*.

For hosts with multiple vnodes, `pbsnodes` operates on a host and all of its vnodes, where the hostname is resources_available.host. To offline a single vnode in a multi-vnoded system, use:

```
qmgr -c "set node <vnode name> state=offline"
```

Users without operator or manager privilege cannot view custom resources which have been created to be invisible to users.

To act on individual vnodes, use the `qmgr` command.

2.37.3   Options to `pbsnodes`

(no options) If neither options nor a host list is given, the `pbsnodes` command prints usage syntax.

-a

Lists all hosts and all their attributes (available and used.)

When listing a host with multiple vnodes:

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.

For consumable resources, the output for each resource is the sum of that resource across all vnodes on that host.

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>`".

-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.)

For hosts with multiple vnodes, pbsnodes operates on a host and all of its vnodes, where the hostname is resources_available.host, which is the name of the natural vnode. To offline a single vnode in a multi-vnoded system, use:

```
Qmgr: qmgr –c "set node <vnode name> state=offline"
```

Requires PBS Manager or Operator privilege.

-r host_list

Clears *OFFLINE* from listed hosts.

-s server

Specifies the PBS server to which to connect.

-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.

--version

> The `pbsnodes` command returns its PBS version information and exits. This option can only be used alone.

2.37.4   Operands

server

> Specifies the server to which to connect. Default: default server.

host_list

> Specifies the host(s) whose status will be returned. Format:
>
> *hostname [hostname ...]*

2.37.5   Exit Status

Zero upon success.

Greater than zero, if:

- incorrect operands are given,
- `pbsnodes` cannot connect to the server,
- there is an error querying the server for the nodes.

2.37.6   See Also

The PBS Professional Administrator's Guide, `pbs_server(8B) and qmgr(8B)`

2.38   pbsrun

General-purpose wrapper script for `mpirun`

2.38.1   Synopsis

pbsrun

pbsrun --version

2.38.2   Description

pbsrun is a wrapper script for any of several versions of mpirun. This provides a user-transparent way for PBS to control jobs which call mpirun in their job scripts. The pbsrun_wrap script instantiates pbsrun so that the wrapper script for the specific version of mpirun being used has the same name as that version of mpirun.

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] refer 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.

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.

For all of the wrapped MPIs, the maximum number of ranks that can be launched is the number of entries in $PBS_NODEFILE.

The wrapped MPIs are:

- MPICH-GM's `mpirun` (`mpirun.ch_gm`) with `rsh/ssh`

- MPICH-MX's `mpirun` (`mpirun.ch_mx`) with `rsh/ssh`

- MPICH-GM's `mpirun` (`mpirun.mpd`) with MPD

- MPICH-MX's `mpirun` (`mpirun.mpd`) with MPD

- MPICH2's `mpirun`

- Intel MPI's `mpirun`

- MVAPICH1's `mpirun`

- MVAPICH2's `mpiexec`

- IBM's `poe`

2.38.3   Options

--version

The `pbsrun` command returns its PBS version information and exits. This option can only be used alone.

2.38.4   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"
```

2.38.4.1   Initialization Script Options

options_to_retain

> Space-separated list of options and values that `pbsrun.<mpirun ver-sion/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 ver-sion/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.

option_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 "configfile" 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 option_to_configfile parameter.

configfile_firstline_action()

> Returns the item that is put in the first line of the configuration file specified in the option_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`."

2.38.4.2   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.

2.38.5   Versions/Flavors of `mpirun`

2.38.5.1   MPICH-GM's `mpirun` (`mpirun.ch_gm`) with `rsh`/ `ssh`: `pbsrun.ch_gm`

2.38.5.1.i   Syntax

pbsrun.ch_gm <options> <executable> <arg1> <arg2> ... <argn>

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.

2.38.5.1.ii   Options Handling

If executed inside a PBS job script, all `mpirun.ch_gm` options given are passed on to the actual `mpirun` call with these exceptions:

-machinefile <file>

> The file argument contents are ignored and replaced by the contents of the $PBS_NODEFILE.

-np

> If not specified, the number of entries found in the $PBS_NODEFILE is used.

-pg

> The use of the -pg option, for having multiple executables on multiple hosts, is allowed but it is up to user to make sure only PBS hosts are specified in the process group file; MPI processes spawned are not guaranteed to be under the control of PBS.

2.38.5.1.iii   Wrap/Unwrap

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
```

2.38.5.2   MPICH-MX's `mpirun` (`mpirun.ch_mx`) with `rsh/ssh`: `pbsrun.ch_mx`

2.38.5.2.i   Syntax

pbsrun.ch_mx <options> <executable> <arg1> <arg2> ... <argn>

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.

2.38.5.2.ii   Options HANDLING

If executed inside a PBS job script, all `mpirun.ch_gm` options given are passed on to the actual `mpirun` call with some exceptions:

-machinefile <file>
> The file argument contents is ignored and replaced by the contents of the $PBS_NODEFILE.

-np
> If not specified, the number of entries found in the $PBS_NODEFILE is used.

-pg
> The use of the -pg option, for having multiple executables on multiple hosts, is allowed but it is up to user to make sure only PBS hosts are specified in the process group file; MPI processes spawned are not guaranteed to be under the control of PBS.

2.38.5.2.iii   Wrap/Unwrap

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**

2.38.5.3   MPICH-GM's `mpirun` (`mpirun.mpd`) with MPD: `pbsrun.gm_mpd`

2.38.5.3.i   Syntax

pbsrun.gm_mpd <options> <executable> <arg1> <arg2> ... <argn>

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.

2.38.5.3.ii   Options Handling

If executed inside a PBS job script, all mpirun.ch_gm with MPD options given are passed on to the actual mpirun call with these exceptions:

-m <file>

The file argument contents are ignored and replaced by the contents of the $PBS_NODEFILE.

-np

If not specified, the number of entries found in the $PBS_NODEFILE is used.

-pg

The use of the -pg option, for having multiple executables on multiple hosts, is allowed but it is up to user to make sure only PBS hosts are specified in the process group file; MPI processes spawned are not guaranteed to be under the control of PBS.

2.38.5.3.iii   Startup/Shutdown

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. The default is rsh.

The script also takes care of shutting down the MPD daemons at the end of a run.

2.38.5.3.iv   Wrap/Unwrap

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

2.38.5.4 MPICH-MX's `mpirun (mpirun.mpd)` with MPD: `pbsrun.mx_mpd`

2.38.5.4.i Syntax

pbsrun.mx_mpd <options> <executable> <arg1> <arg2> ... <argn>

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.

2.38.5.4.ii Options Handling

If executed inside a PBS job script, all `mpirun.mx_mpd` with MPD options given are passed on to the actual `mpirun` call with these exceptions:

-m <file>

> The file argument contents are ignored and replaced by the contents of the $PBS_NODEFILE.

-np

> If not specified, the number of entries found in the $PBS_NODEFILE is used.

-pg

> The use of the -pg option, for having multiple executables on multiple hosts, is allowed but it is up to user to make sure only PBS hosts are specified in the process group file; MPI processes spawned are not guaranteed to be under the control of PBS.

2.38.5.4.iii Startup/Shutdown

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.

2.38.5.4.iv   **Wrap/Unwrap**

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

2.38.5.5   MPICH2's `mpirun`: `pbsrun.mpich2`

2.38.5.5.i   **Syntax**

pbsrun.mpich2 [global args] [local args] executable [args] [: [local args] executable [args]]

    - or -

pbsrun.mpich2 -configfile <configfile>

where <configfile> contains command line segments as lines:

```
[local args] executable1 [args]
[local args] executable2 [args]
[local args] executable3 [args]
```

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.

2.38.5.5.ii   **Options Handling**

If executed inside a PBS job script, all MPICH2's `mpirun` options given are passed on to the actual `mpirun` call with these exceptions:

-host and -ghost
> For specifying the execution host to run on. Not passed on to the actual `mpirun` call.

-machinefile <file>
> The file argument contents are ignored and replaced by the contents of the $PBS_NODEFILE.

MPICH2 s mpirun -localonly <x>
> For specifying the <x> number of processes to run locally. Not supported. The user is advised instead to use the equivalent arguments: -np <x> -localonly . The reason for this is that the `pbsrun` wrapper script cannot handle

a variable number of arguments to an option (e.g. "-localonly" has 1 argument and "-localonly <x>" has 2 arguments).

-np

If user did not specify a -np option, then no default value is provided by the PBS wrapper scripts. It is up to the local mpirun to decide what the reasonable default value should be, which is usually 1.

2.38.5.5.iii Startup/Shutdown

The script takes care of ensuring that the MPD daemons on each of the hosts 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.

2.38.5.5.iv Wrap/Unwrap

To wrap MPICH2's mpirun script:

    # pbsrun_wrap [MPICH2_BIN_PATH]/mpirun pbsrun.mpich2

To unwrap MPICH2's mpirun script:

    # pbsrun_unwrap pbsrun.mpich2

In the case where MPICH2 uses mpirun.py, run pbsrun_wrap on mpirun.py itself.

2.38.5.6 Intel MPI's `mpirun`: `pbsrun.intelmpi`

2.38.5.6.i Syntax

pbsrun.intelmpi [mpdboot options] [mpiexec options] executable [prog-args] [: [mpiexec options] executable [prog-args]]

 - or -

pbsrun.intelmpi [mpdboot options] -f <configfile>

where [mpdboot options] are any options to pass to the mpdboot program, which is automatically called by Intel MPI's mpirun to start MPDs, and <configfile> contains command line segments as lines.

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.

2.38.5.6.ii   Options Handling

If executed inside a PBS job script, all of the options to the PBS interface to MPI's `mpirun` are passed to the actual `mpirun` call with these exceptions:

-host and -ghost
> For specifying the execution host to run on. Not passed on to the actual `mpirun` call.

-machinefile <file>
> The file argument contents are ignored and replaced by the contents of the $PBS_NODEFILE.

mpdboot options --totalnum=* and --file=*
> Ignored and replaced by the number of unique entries in $PBS_NODEFILE and name of $PBS_NODEFILE respectively.

arguments to mpdboot options --file=* and -f <mpd_hosts_file>
> Replaced by $PBS_NODEFILE.

-s

> If `pbsrun.intelmpi` is called inside a PBS job, Intel MPIs `mpirun` **-s** argument to `mpdboot` are not supported as this closely matches the `mpirun` option -s <spec> . The user can simply run a separate `mpdboot` **-s** before calling `mpirun`. A warning message is issued by `pbsrun.intelmpi` upon encountering a -s option telling users of the supported form.

-np

> If the user does not specify a -np option, then no default value is provided by the PBS wrap scripts. It is up to the local `mpirun` to decide what the reasonable default value should be, which is usually 1.

2.38.5.6.iii   Startup/Shutdown

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.

2.38.5.6.iv   Wrap/Unwrap

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**

2.38.5.7   MVAPICH1's `mpirun`: `pbsrun.mvapich1`

2.38.5.7.i   Syntax

pbsrun.mvapich1 <mpirun options> <executable> <options>

The PBS wrapper script to MVAPICH1's mpirun is called `pbsrun.mvapich1`.

Only one executable can be specified. MVAPICH1 allows the use of InfiniBand.

If executed inside a PBS job, this allows for PBS to be aware of all MVAPICH1 ranks and track their resources, 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` was used.

2.38.5.7.ii   Options Handling

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.

2.38.5.7.iii   Wrap/Unwrap

To wrap MVAPICH1's `mpirun` script:

> `# pbsrun_wrap <path-to-actual-mpirun> pbsrun.mvapich1`

To unwrap MVAPICH1's `mpirun` script:

> `# pbsrun_unwrap pbsrun.mvapich1`

2.38.5.8   MVAPICH2's `mpiexec`: `pbsrun.mvapich2`

2.38.5.8.i   Syntax

pbsrun.mvapich2 <mpiexec args> executable <executable's args> [: <mpiexec args> executable <executable's args>]

The PBS wrapper script to MVAPICH2's mpiexec is called `pbsrun.mvapich2`.

Multiple executables can be specified using the colon notation. MVAPICH2 allows the use of InfiniBand.

If executed inside a PBS job, this allows for PBS to be aware of all MVAPICH2 ranks and track their resources, so that PBS can perform accounting and have complete job control.

If executed outside of a PBS job, it behaves exactly as if standard `mpiexec` was used.

2.38.5.8.ii   Options Handling

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.

2.38.5.8.iii   Wrap/Unwrap

To wrap MVAPICH2's `mpiexec` script:

```
# pbsrun_wrap <path-to-actual-mpiexec> pbsrun.mvapich2
```

To unwrap MVAPICH2's `mpiexec` script:

```
# pbsrun_unwrap pbsrun.mvapich2
```

2.38.5.9   IBM's `poe`: `pbsrun.poe`

2.38.5.9.i   Syntax

pbsrun.poe <options> <executable> <arg1> <arg2> ... <argn>

The PBS wrapper script to IBM's poe is called `pbsrun.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` allows LAPI or MPI programs to use InfiniBand or the HPS in US mode.

If executed inside a PBS job, this allows for PBS to track all resources and MPI ranks. PBS can perform accounting and have complete job control.

If executed outside of a PBS job, it behaves exactly as if standard `poe` was used.

The script will use the -euilib {ip | us} option and the MP_EUILIB environment variable to indicate use of US mode, to maintain compatibility with standard `poe`.

2.38.5.9.ii   Options and Environment Variables

Users submitting jobs whose programs use `poe` can set environment variables instead of using options to `poe`. The equivalent environment variable is listed with its `poe` option. If executed inside a PBS job script, all pbsrun.`poe` options and environment variables except the following are passed on to `poe`:

-devtype, MP_DEVTYPE

> If InfiniBand is not specified in either the option or the environment variable, the InfiniBand interconnect is not used for the program.

-euidevice, MP_EUIDEVICE

> Ignored by PBS.

-euilib {ip|us}, MP_EUILIB

> If the command line option -euilib is set, it will take precedence over the MP_EUILIB environment variable. If set to "*us*", the program uses User Space mode. If set to any other value, that value is passed to IBM `poe`.

-hostfile, -hfile, MP_HOSTFILE

> Ignored. If this is specified, PBS prints the following:
>
> > "pbsrun.poe: Warning, –hostfile value replaced by PBS"
>
> or
>
> > "pbsrun.poe: Warning –hfile value replaced by PBS"
>
> If this environment variable is set when a `poe` job is submitted, PBS prints the following error message:
>
> > "pbsrun.poe: Warning MP_HOSTFILE value replaced by PBS"

-instances, MP_INSTANCES

> The option and the environment variable are treated differently:
>
> -instances
> > If the option is set, PBS prints a warning:
> >
> > > "pbsrun.poe: Warning, –instances cmd line option removed by PBS"
>
> MP_INSTANCES
> > If the environment variable is set, PBS uses it to calculate the number of network windows for the job. The maximum value allowed can be requested by using the string "`max`" for the environment variable. If the environment variable is set to a value greater than the maximum allowed value, it is replaced with the maximum allowed value. The default maximum value is *4*.

-procs, MP_PROCS

> This option or environment variable should be set to the total number of mpiprocs requested by the job when using US mode. If neither this option nor the MP_PROCS environment variable is set, PBS uses the number of entries in $PBS_NODEFILE. If this option is set to N, and the job is submitted with a total of M mpiprocs:

> If N >=M:

>> The value N is passed to IBM poe.

> If N < M and US mode is not being used:

>> The value N is passed to poe.

> If N < M and US mode is being used:

>> US mode is turned off and a warning is printed:

>> "pbsrun.poe: Warning, user mode disabled due to MP_PROCS setting"

2.38.5.9.iii   Wrap/Unwrap

To wrap IBM's poe:

```
# pbsrun_wrap <path_to_actual_poe> pbsrun.poe
```

To unwrap the IBM poe:

```
# pbsrun_unwrap pbsrun.poe
```

2.38.6   Requirements

The mpirun being wrapped must be installed and working on all the nodes in the PBS cluster.

2.38.7   Errors

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.

2.38.8   See Also

The PBS Professional Administrator's Guide

pbs_attach(8B), pbsrun_wrap(8B), pbsrun_unwrap(8B)

2.39   pbsrun_unwrap

Unwraps `mpirun`, reversing `pbsrun_wrap`

2.39.1   Synopsis

pbsrun_unwrap pbsrun.<mpirun version/flavor>

pbsrun_unwrap --version

2.39.2   Description

The `pbsrun_unwrap` script is used to reverse the actions of the `pbsrun_wrap` script.

Use `pbsrun_wrap` to wrap `mpirun`.

2.39.3   Usage

2.39.3.1   Syntax:

pbsrun_unwrap pbsrun.<mpirun version/flavor>

For example, running the following:

pbsrun_unwrap pbsrun.ch_gm

causes the following actions:

Checks for a link in `$PBS_EXEC/lib/MPI/pbsrun.ch_gm.link`; If one exists, get the pathname it points to:

/opt/mpich–gm/bin/mpirun.ch_gm.actual

rm $PBS_EXEC/lib/MPI/pbsrun.mpirun.ch_gm.link

rm /opt/mpich–gm/bin/mpirun.ch_gm

rm $PBS_EXEC/bin/pbsrun.ch_gm

mv /opt/mpich–gm/bin/mpirun.ch_gm.actual /opt/mpich–gm/bin/mpirun.ch_gm

2.39.4   Options

--version

The `pbsrun_unwrap` command returns its PBS version information and exits. This option can only be used alone.

2.39.5   See Also

The PBS Professional Administrator's Guide

`pbs_attach(8B), pbsrun(8B), pbsrun_wrap(8B)`

2.40   pbsrun_wrap

General-purpose script for wrapping `mpirun` in pbsrun

2.40.1   Synopsis

pbsrun_wrap [-s] <path_to_actual_mpirun> pbsrun.<mpirun version/flavor>

pbsrun_wrap --version

2.40.2   Description

The `pbsrun_wrap` script is used to wrap any of several versions of `mpirun` in pbsrun. The `pbsrun_wrap` script creates a symbolic link with the same path and name as the `mpirun` being wrapped. This calls `pbsrun`, which uses `pbs_attach` to give MOM control of jobs. The result is transparent to the user; when `mpirun` is called from inside a PBS job, PBS can monitor and control the job, but when `mpirun` is called from outside of a PBS job, it behaves as it would normally. See the `pbs_attach(8B)` and `pbsrun(8B)` man pages.

Use `pbsrun_unwrap` to reverse the process.

2.40.3   Options

-s

> Sets the "strict_pbs" options in the various initialization scripts (e.g. `pbsrun.bgl.init`, `pbsrun.ch_gm.init`, etc...) to *1* from the default *0*. This means that the `mpirun` being wrapped by `pbsrun` will only be executed if inside a PBS environment. Otherwise, the user will get the error:
>
> > Not running under PBS exiting since strict_pbs is enabled; exe-
> >   cute only in PBS

--version

> The `pbsrun_wrap` command returns its PBS version information and exits. This option can only be used alone.

2.40.4   USAGE

2.40.4.1   Syntax:

pbsrun_wrap [-s] <path_to_actual_mpirun> pbsrun.<mpirun version/flavor>

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 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

2.40.5   Requirements

The mpirun being wrapped must be installed and working on all the nodes in the PBS cluster.

2.40.6   See Also

The PBS Professional Administrator's Guide

pbs_attach(8B), pbsrun(8B), pbsrun_unwrap(8B)

2.41 printjob

Prints job information

2.41.1 Synopsis

printjob [-a | -s] job ID

printjob [-a] <file path> [<file path>...]

printjob --version

2.41.2 Description

The `printjob` command is used to print job information. You can print information either from the server, using a job ID, or from the execution host, using a file path.

Whether or not a MOM is running on the server host, you must use the job ID at the server host.

By default all the job data including job attributes are printed. This can be suppressed with the `-a` option.

You can print out just the job script using the `-s` option at the server and execution hosts.

This command is mainly useful for troubleshooting, as during normal operation, the `qstat(8B)` command is the preferred method for displaying job-specific data and attributes.

2.41.3 Permissions

In order to execute `printjob`, the user must have root or Windows Administrator privilege.

2.41.4 Options to **printjob**

(no options>

> Prints all job data including job attributes.

-a

> Suppresses the printing of job attributes. Cannot be used with `-s` option.

-s

> Prints out the job script only. Cannot be used with `-a` option. Cannot be used with *file path* argument.

--version

The printjob command returns its PBS version information and exits. This option can only be used alone.

2.41.5   Operands for `printjob`

<file path>

The printjob command accepts one or more file path operands at the execution host. Files are found in PBS_HOME/mom_priv/jobs/ on the primary execution host. File path must include full path to file. Cannot be used with -s option.

<job ID>

The printjob command accepts a job ID at the server host. The format is described in . Data service must be running.

2.41.6   Standard Error

The printjob command writes a diagnostic message to standard error for each error occurrence.

2.41.7   Exit Status

Zero upon successful processing of all the operands presented to the printjob command.

Greater than zero if the printjob command fails to process any operand.

2.41.8   See Also

The PBS Professional Administrator's Guide, , and

2.42   qalter

Alters a PBS job

2.42.1   Synopsis

qalter [-a date_time] [-A account_string] [-c interval] [-e path] [-h hold_list] [-j join] [-k
keep] [-l resource_list] [-m mail_events] [-M user_list] [-N name] [-o path] [-p priority]
[-P project] [-r c] [-S path] [-u user_list] [-W additional_attributes] job_identifier_list

qalter --version

2.42.2   Description

The `qalter` command is used to alter one or more PBS batch jobs. The attributes listed with
the options to the `qalter` command can be modified. If any of the modifications to a job
fails, none of the job's attributes is modified.

A job that is in the process of provisioning cannot be altered.

2.42.2.1   Required privilege

•   A non-privileged user may only lower the limits for resources

•   A Manager or Operator may lower or raise requested resource limits, except for per-pro-
      cess limits such as `pcput` and `pmem`, because these are set when the process starts, and
      enforced by the kernel.

•   The `qalter` command cannot be used by a non-privileged user to alter a custom
      resource which has been created to be invisible or read-only for users.

2.42.2.2   Modifying resources and job placement:

If a job is running, the only resources that can be modified are `cput` (CPU time) and `walltime`.

If a job is queued, any resource mentioned in the options to the `qalter` command can be
modified, but requested modifications must fit within the limits set at the server and queue for
the amount of each resource allocated for queued jobs. If a requested modification does not
fit within these limits, the modification is rejected.

Note that a job's resource request must fit within the queue's and server's resource run limits.
If a modification to a resource exceeds the amount of the resource allowed by the queue or
server to be used by running jobs, the job is never run.

Resources are modified by using the -l option, either in chunks inside of selection statements,
or in job-wide modifications using r*esource_name=value* pairs. The selection statement is of
the form:

-l select=[N:]chunk[+[N:]chunk ...]

where *N* specifies how many of that chunk, and a chunk is of the form:

resource_name=value[:resource_name=value ...]

Job-wide *resource_name=value* modifications are of the form:

-l resource_name=value[,resource_name=value ...]

Placement of jobs on nodes is changed using the place statement:

-l place=modifier[:modifier]

where modifier is any combination of *group*, *excl*, and/or one of *free|pack|scatter*.

For more on resource requests, usage limits and job placement, see `pbs_resources(7B)`.

2.42.2.3   Modifying attributes:

The user alters job attributes by giving options to the `qalter` command. Each `qalter` option changes a job attribute.

The behavior of the `qalter` command may be affected by any site hooks. Site hooks can modify the job's attributes, change its routing, etc.

2.42.3   Options to `qalter`

-a date_time

> Changes the point in time after which the job is eligible for execution. Given in pairs of digits. Sets job's Execution_Time attribute to date_time.
>
> Format: *Datetime*
>
> Each portion of the date defaults to the current date, as long as the next-smaller portion is in the future. For example, if today is the 3rd of the month and the specified day DD is the 5th, the month MM will be set to the current month.
>
> If a specified portion has already passed, the next-larger portion will be set to one after the current date. For example, if the day DD is not specified, but the hour hh is specified to be 10:00 a.m. and the current time is 11:00 a.m., the day DD will be set to tomorrow.
>
> The job's Execution_Time attribute can be altered after the job has begun execution, in which case it will not take effect until the job is rerun.

-A account_string

Replaces the accounting string associated with the job. Used for labeling accounting data. Sets job's Account_Name attribute to account_string. This attribute cannot be altered once the job has begun execution.

Format: String

-c checkpoint_spec

Changes when the job will be checkpointed. Sets job's Checkpoint attribute. An $action script is required to checkpoint the job. This attribute can be altered after the job has begun execution, in which case the new value will not take effect until the job is rerun.

See the pbs_mom(8B) man page.

The argument checkpoint_spec can take on one of the following values:

c

Checkpoint at intervals, measured in CPU time, set on job's execution queue. If no interval set at queue, job is not checkpointed

c=<minutes of CPU time>

Checkpoint at intervals of specified number of minutes of job CPU time. This value must be > 0. If interval specified is less than that set on job's execution queue, queue's interval is used.

Format: Integer

w

Checkpoint at intervals, measured in walltime, set on job's execution queue. If no interval set at queue, job is not checkpointed.

w=<minutes of walltime>

Checkpoint at intervals of the specified number of minutes of job wall-time. This value must be greater than zero. If the interval specified is less that that set on the execution queue in which the job resides, the queue's interval is used.

Format: Integer

n

No checkpointing.

s

Checkpoint only when the server is shut down.

u

Unset. Defaults to behavior when interval argument is set to s.

Default: u.

Format: *String.*

-e path

> Replaces the path to be used for the job's standard error stream. Sets job's Error_Path attribute to path.
>
> Format: *[hostname:]path_name*
>
> The path will be interpreted as follows:
>
> *path_name*
>> If *path_name* is a relative path, then it is taken to be relative to the current working directory of the qalter command, where it is executing on the current host.
>>
>> If *path_name* is an absolute path, then it is taken to be an absolute path on the current host where the qalter command is executing.
>
> *hostname:path_name*
>> If *path_name* is a relative path, then it is taken to be relative to the user's home directory on the host named hostname.
>>
>> If *path_name* is an absolute path, then it is the absolute path on the host named hostname.
>>
>> If *path_name* does not include a filename, the default filename will be
>>
>> *jobid.ER*
>
> If the -e option is not specified, the default filename for the standard error stream is used.
>
> Format: *job_name.esequence_number*
>
> This attribute can be altered after the job has begun execution, in which case the new value will not take effect until the job is rerun.

-h hold_list

> Updates the job's hold list. Adds hold_list to the job's Hold_Types attribute. The hold_list is a string of one or more of the following:

Table 2-4: Hold Types

Hold Type	Meaning
u	Add a USER hold.
o	Add OTHER hold. Requires operator privilege.
n	Clear the holds for which the user has privilege.

This attribute can be altered after the job has begun execution, in which case the new value will not take effect until the job is rerun.

-j join

Changes whether and how to join the job's standard error and standard output streams. Sets job's Join_Path attribute to join.

Possible values of join:

Table 2-5: Join Path Options

Value	Meaning
oe	Standard error and standard output are merged into standard output.
eo	Standard error and standard output are merged into standard error.
n	Standard error and standard output are not merged.

This attribute can be altered after the job has begun execution, in which case the new value will not take effect until the job is rerun.

Default: not merged.

-k keep

Changes whether and which of the standard output and standard error streams will be retained on the execution host. Overrides default path names for these streams. Sets the job's Keep_Files attribute to keep.

The keep argument can take on the following values:

Table 2-6: Keep Argument Values

Option	Meaning
e	The standard error stream is retained on the execution host, in the job's staging and execution directory. The filename will be: *job_name.e<sequence number>*
o	The standard output stream is retained on the execution host, in the job's staging and execution directory. The filename will be: *job_name.o<sequence number>*
eo, oe	Both standard output and standard error streams are retained on the execution host, in the job's staging and execution directory.
n	Neither stream is retained.

This attribute cannot be altered once the job has begun execution.

In the case where output and/or error is retained on the execution host in a job-specific staging and execution directory created by PBS, these files are deleted when PBS deletes the directory.

Default: neither is retained.

-l resource_arg

Allows the user to change requested resources and job placement. Sets job's Resource_list attribute to resource_arg. Uses resource request syntax. Requesting a resource places a limit on its usage. Users without manager or operator privilege cannot alter a custom resource which was created to be invisible or read-only for users.

Requesting resources in chunks:

Format: *-l select=[N:]chunk[+[N:]chunk ...]*

where N specifies how many of that chunk, and a chunk is:

Format: *resource_name=value[:resource_name=value ...]*

Requesting job-wide resources:

Format: *-l resource_name=value[,resource_name=value ...]*

Specifying placement of jobs:

Format: *-l place=[arrangement][: sharing][: grouping]*

where

arrangement is one of *free | pack | scatter*

sharing is one of *excl | shared*

grouping can have only one instance of *group=resource*

and where

free

> Place job on any vnode(s).

pack:

> All chunks will be taken from one host.

scatter

> Only one chunk with any MPI processes will be taken from a host. A chunk with no MPI processes may be taken from the same node as another chunk.

excl

> Only this job uses the vnodes chosen.

shared

> This job can share the vnodes chosen.

group=resource

> Chunks will be grouped according to a resource. All nodes in the group must have a common value for the resource, which can be either the built-in resource host or a site-defined node-level resource.

If a requested modification to a resource would exceed the job's queue's limits, the resource request will be rejected. For a running job, resources may only be reduced. Which resources can be altered is system-dependent.

If the job was submitted with an explicit "**-l select=**", then node level resources must be qaltered using the "**-l select=**" form. In this case a node level resource RES cannot be qaltered with the "**-l RES**" form.

Examples:

1. Submit the job:

```
% qsub –l select=1:ncpus=2:mem=512mb jobscript
```

Job's ID is 230

2. qalter the job using "-l RES" form:

`% `**`qalter –l ncpus=4 230`**

Error reported by `qalter`:

`qalter`**`: Resource must only appear in "select" specification when select is used: ncpus 230`**

3. qalter the job using the "-l select=" form:

`% `**`qalter –l select=1:ncpus=4:mem=512mb 230`**

No error reported by `qalter`:

`%`

For more on resource requests, usage limits and job placement, see `pbs_resources(7B)`.

-m mail_events

Changes the set of conditions under which mail about the job is sent. Sets job's Mail_Points attribute to mail_events. The mail_events argument can be either "*n*" or any combination of "*a*", "*b*", and "*e*".

Table 2-7: Mail Events Options

Option	Meaning
n	No mail will be sent.
a	Mail is sent when the job is aborted by the batch system.
b	Mail is sent when the job begins execution.
e	Mail is sent when the job terminates.

Format: *String*.

Default value: "*a*".

-M user_list

Alters list of users to whom mail about the job is sent. Sets job's Mail_Users attribute to user_list.

Format: *user[@host][,user[@host],...]*

Default: job owner.

-N name

        Renames the job. Sets job's Job_Name attribute to name.

        Format: string, up to 15 characters in length. It must consist of an alphabetic character followed by printable, non-white-space characters.

        Default: if a script is used to submit the job, the job's name is the name of the script. If no script is used, the job's name is "*STDIN*".

-o path

        Alters path to be used for the job's standard output stream. Sets job's Output_Path attribute to path.

        Format: *[hostname:]path_name*

        The path will be interpreted as follows:

        path_name

            If *path_name* is a relative path, then it is taken to be relative to the current working directory of the command, where it is executing on the current host.

            If *path_name* is an absolute path, then it is taken to be an absolute path on the current host where the command is executing.

        hostname:path_name

            If *path_name* is a relative path, then it is taken to be relative to the user's home directory on the host named hostname.

            If *path_name* is an absolute path, then it is the absolute path on the host named hostname.

            If *path_name* does not include a filename, the default filename will be

            jobid.OU

        If the -o option is not specified, the default filename for the standard output stream is used. It has this form:

        job_name.osequence_number

        This attribute can be altered after the job has begun execution, in which case the new value will not take effect until the job is rerun.

-p priority

        Alters priority of the job.

        Sets job's Priority attribute to priority.

        This attribute can be altered after the job has begun execution, in which case the new value will not take effect until the job is rerun.

        Format: host-dependent integer.

Range: [*-1024, +1023*] inclusive.

Default: *zero*.

-P project

Specifies a project for the job. Sets job's project attribute to specified value.

Format: String.

Project name can contain any characters except for the following: Slash ("/"), left bracket ("["), right bracket ("]"), double quote ("""), semicolon (";"), colon (":"), vertical bar ("|"), left angle bracket ("<"), right angle bracket (">"), plus ("+"), comma (","), question mark ("?"), and asterisk ("*").

Default value: "*_pbs_project_default*".

-r y|n

Changes whether the job is rerunnable. Sets job's Rerunnable attribute to the argument. Does not affect how job is treated when the job was unable to begin execution.

See the qrerun(1B) command.

Format: single character, "*y*" or "*n*".

y
Job is rerunnable.

n
Job is not rerunnable.

Default: "*y*".

-S path_list

Specifies the interpreter or shell path for the job script. Sets job's Shell_Path_List attribute to path_list.

The path_list argument is the full path to the interpreter or shell including the executable name.

Only one path may be specified without a host name. Only one path may be specified per named host. The path selected is the one whose host name is that of the server on which the job resides.

This attribute can be altered after the job has begun execution, but in this case the new value will not take effect until the job is rerun.

Format:

path[@host][,path@host ...]
Default: user's login shell on execution node.

Example of using `bash` via a directive:

```
#PBS -S /bin/bash@mars,/usr/bin/bash@jupiter
```

Example of running a Python script from the command line on UNIX/ Linux:

```
qsub -S $PBS_EXEC/bin/pbs_python <script name>
```

Example of running a Python script from the command line on Windows:

```
qsub -S %PBS_EXEC%\bin\pbs_python.exe <script name>
```

-u user_list

Alters list of usernames. Job will be run under a username from this list. Sets job's User_List attribute to user_list.

Only one username may be specified without a host name. Only one username may be specified per named host. The server on which the job resides will select first the username whose host name is the same as the server name. Failing that, the next selection will be the username with no specified hostname. The usernames on the server and execution hosts must be the same. The job owner must have authorization to run as the specified user.

This attribute cannot be altered once the job has begun execution.

Format: *user[@host][,user@host ...]*

Default: job owner (username on submit host.)

-W additional_attributes

The -W option allows change in specification of additional job attributes.

Format: *-W attribute_name = value[,attribute_name=value...]*

If white space occurs within the additional_attributes argument, or the equal sign ("=") occurs within an *attribute_value* string, then that must be enclosed with single or double quotes. PBS supports the following attributes within the -W option:

depend=dependency_list

Defines dependencies between this and other jobs. Sets the job's depend attribute to dependency_list. The dependency_list has the form:

type:arg_list[,type:arg_list ...]

where except for the *on* type, the *arg_list* is one or more PBS job IDs in the form:

jobid[:jobid ...]

The type can be:

after: arg_list

> This job may be scheduled for execution at any point after all jobs in *arg_list* have started execution.

afterok: arg_list

> This job may be scheduled for execution only after all jobs in *arg_list* have terminated with no errors. See section 2.42.6.1, "Warning about exit status with csh:", on page 143.

afternotok: arg_list

> This job may be scheduled for execution only after all jobs in *arg_list* have terminated with errors. See section 2.42.6.1, "Warning about exit status with csh:", on page 143.

afterany: arg_list

> This job may be scheduled for execution after all jobs in *arg_list* have terminated, with or without errors.

before: arg_list

> Jobs in *arg_list* may begin execution once this job has begun execution.

beforeok: arg_list

> Jobs in *arg_list* may begin execution once this job terminates without errors. See section 2.42.6.1, "Warning about exit status with csh:", on page 143.

beforenotok: arg_list

> If this job terminates execution with errors, then jobs in *arg_list* may begin. See section 2.42.6.1, "Warning about exit status with csh:", on page 143.

beforeany: arg_list

> Jobs in *arg_list* may begin execution once this job terminates execution, with or without errors.

on: count

> This job may be scheduled for execution after *count* dependencies on other jobs have been satisfied. This type is used in conjunction with one of the **before** types listed. *count* is an integer greater than 0.

Restrictions:

Job IDs in the *arg_list* of *before* types must have been submitted with a type of *on*.

To use the *before* types, the user must have the authority to alter the jobs in *arg_list*. Otherwise, the dependency is rejected and the new job aborted.

Error processing of the existence, state, or condition of the job on which the newly-submitted job is a deferred service, i.e. the check is performed after the job is queued. If an error is detected, the new job will be deleted by the server. Mail will be sent to the job submitter stating the error.

Dependency examples:

```
qalter -W depend = afterok:123.host1.domain.com /tmp/script

qalter -W depend= before:234.host1.com:235.host1.com /tmp/
  script
```

group_list=g_list

Alters list of group names. Job will be run under a group name from this list. Sets job's group_List attribute to g_list.

Only one group name may be specified without a host name. Only one group name may be specified per named host. The server on which the job resides will select first the group name whose host name is the same as the server name. Failing that, the next selection will be the group name with no specified hostname. The group names on the server and execution hosts must be the same.

Format: *group[@host][,group@host ...]*

Default: login group name of job owner.

sandbox=<value>

Changes which directory PBS uses for the job's staging and execution.

Allowed values:

> *PRIVATE*
> PBS creates a job-specific directory for staging and execution.

> *HOME* or unset
> PBS uses the user's home directory for staging and execution.

Format: String

stagein=path_list

stageout=path_list

Changes files or directories to be staged-in before execution or staged-out after execution is complete. Sets the job's stagein and stageout attributes to the specified path_lists. On completion of the job, all

staged-in and staged-out files and directories are removed from the execution host(s). The path_list has the form:

filespec[,filespec]

where *filespec* is

local_path@hostname:remote_path

regardless of the direction of the copy. The name *local_path* is the name of the file or directory on the primary execution host. It can be relative to the staging and execution directory on the execution host, or it can be an absolute path.

The "@" character separates *local_path* from *remote_path*.

The name *remote_path* is the path on *hostname*. The name can be relative to the staging and execution directory on the primary execution host, or it can be an absolute path.

If path_list has more than one *filespec*, i.e. it contains commas, it must be enclosed in double-quotes.

umask=NNNN

Alters the umask with which the job will be started. Controls umask of job's standard output and standard error. Sets job's umask attribute to *NNNN*. Can be used with one to four digits; typically two.

The following example allows group and world read on the job's output:

    –W umask=33

Default value: *077*

--version

The qalter command returns its PBS version information and exits. This option can only be used alone.

2.42.4   Operands

The qalter command accepts a *job_identifier_list* as its operand. The *job_identifier_list* is one or more job IDs for normal jobs or array jobs. Individual subjobs of an array job are not alterable.

Note that some shells require that you enclose a job array ID in double quotes.

2.42.5   Standard Error

The qalter command will write a diagnostic message to standard error for each error occurrence.

2.42.6   Exit Status

Zero upon successful processing of input. Exit value will be greater than zero upon failure of `qalter`.

2.42.6.1   Warning about exit status with csh:

If a job is run in `csh` and a .logout file exists in the home directory in which the job executes, the exit status of the job is that of the `.logout` script, not the job script. This may impact any inter-job dependencies.

2.42.7   See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide, `pbs_job_attributes(7B)`, `pbs_resources(7B)`, `qdel(1B)`, `qhold(1B)`, `qmove(1B)`, `qmsg(1B)`, `qrerun(1B)`, `qrls(1B)`, `qselect(1B)`, `qstat(1B)`, `qsub(1B)`

2.43   qdel

Deletes PBS jobs

2.43.1   Synopsis

qdel [-x] [-Wforce| -Wsuppress_email=<N>] job_identifier [job_identifier ...]

qdel --version

2.43.2   Description

The `qdel` command deletes jobs in the order given, whether they are at the local server or at a remote server.

The `qdel` command is used without options to delete queued, running, held, or suspended jobs, while the `-x` option gives it the additional capacity to delete finished or moved jobs. With the `-x` option, this command can be used on finished and moved jobs, in addition to queued, running, held, or suspended jobs.

When this command is used without the -x option, if job history is enabled, the deleted job's history is retained. The -x option is used to additionally remove the history of the job being deleted.

A PBS job may be deleted by its owner, an operator, or the administrator. The server deletes a PBS job by sending a SIGTERM signal, then, if there are remaining processes, a SIGKILL signal.

If someone other than the job's owner deletes the job, mail is sent to the job's owner, or to a list of mail recipients if specified during qsub. See the qsub(1B) man page.

2.43.2.1 How Behavior of **qdel** Command Can Be Affected

The server's default_qdel_arguments attribute may affect the behavior of the qdel command. This attribute is settable by the administrator via the qmgr command. The attribute may be set to "-*Wsuppress_email=<N>*". The server attribute is overridden by command line arguments. See <u>section 6.6, "Server Attributes", on page 314</u>.

If the job is in the process of provisioning, it can be deleted only by using the **-W force** option.

2.43.2.2 Sequence of Events

1. The job's running processes are killed.

2. The epilogue runs.

3. Files that were staged in are staged out. This includes

4. standard out (.o) and standard error (.e) files.

5. Files that were staged in or out are deleted.

6. The job's temp directory is removed.

7. The job is removed from the MOM(s) and the server.

2.43.3 Options to **qdel**

(no options)
> Can delete queued, running, held, or suspended jobs. Does not delete job history for specified job(s).

-W force
> Deletes the job whether or not the job's execution host is reachable. Deletes the job whether or not the job is in the process of provisioning. Cannot be used with the -Wsuppress_email option.

-Wsuppress_email=<N>

> Sets limit on number of emails sent when deleting multiple jobs. If N >= *1* and N or more *job_identifiers* are given, N emails are sent. If N >=1 and less than N job identifiers are given, the number of emails is the same as the number of jobs. If N = *0*, this option is ignored. If N = *-1*, no mail is sent.
>
> Note that there is no space between "W" and "suppress_email".
>
> The <N> argument is an integer.
>
> Cannot be used with -Wforce option.

-x

> Can delete running, queued, suspended, held, finished, or moved jobs. Deletes job history for the specified job(s).

--version

> The qdel command returns its PBS version information and exits. This option can only be used alone.

2.43.4   Operands

The qdel command accepts one or more *job_identifier* operands. These operands can be job identifiers, job array identifiers, job array range identifiers, or subjob identifiers. See Chapter 7, "Formats", on page 403.

Job array identifiers must be enclosed in double quotes for some shells.

2.43.5   Standard Error

The qdel command writes a diagnostic message to standard error for each error occurrence.

2.43.6   Exit Status

Zero upon successful processing of input.

Greater than zero upon error.

2.43.7   See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide, pbs_queue_attributes(7B), pbs_server_attributes(1B), qsub(1B), qsig(1B), pbs_deljob(3B)

2.44 qdisable

Prevents jobs from being enqueued in a queue

2.44.1 Synopsis

qdisable destination ...

qdisable --version

2.44.2 Description

The `qdisable` command directs that a destination queue should no longer accept batch jobs. If the command is accepted, the queue will no longer accept Queue Job requests which specified the disabled queue. Jobs which already reside in the queue will continue to be processed. This allows a queue to be "drained."

In order to execute `qdisable`, the user must have PBS Operator or Manager privilege.

2.44.3 Options

--version

> The `qdisable` command returns its PBS version information and exits. This option can only be used alone.

2.44.4 Operands

The `qdisable` command accepts one or more destination operands. The operands take one of three forms:

queue

@server

queue@server

If *queue* is specified, the request is to disable that queue at the default server. If the *@server* form is given, the request is to disable all the queues at that server. If a full destination identifier, *queue@server*, is given, the request is to disable the named queue at the named server.

2.44.5   Standard Error

The `qdisable` command will write a diagnostic message to standard error for each error occurrence.

2.44.6   Exit Status

Upon successful processing of all the operands presented to the `qdisable` command, the exit status will be a value of zero.

If the `qdisable` command fails to process any operand, the command exits with a value greater than zero.

2.44.7   See Also

The PBS Professional Administrator's Guide and the following manual pages: `pbs_server(8B), qmgr(8B), and qenable(8B)`

2.45   qenable

Allow jobs to be enqueued in a queue

2.45.1   Synopsis

qenable destination ...

qenable --version

2.45.2   Description

The `qenable` command directs that a destination queue should accept batch jobs.

The `qenable` command sends a Manage request to the batch server specified by queue. If the command is accepted, the destination will accept Queue Job requests which specified the queue.

In order to execute `qenable`, the user must have PBS Operator or Manager privilege.

2.45.3   Options

--version

> The `qenable` command returns its PBS version information and exits.
> This option can only be used alone.

2.45.4   Operands

The `qenable` command accepts one or more destination operands. The operands are one of three forms:

queue

@server

queue@server

If *queue* is specified, the request is to enable that queue at the default server. If the *@server* form is given, the request is to enable all the queues at that server. If a full destination identifier, *queue@server*, is given, the request is to enable the named queue at the named server.

2.45.5   Standard Error

The `qenable` command will write a diagnostic message to standard error for each error occurrence.

2.45.6   Exit Status

Upon successful processing of all the operands presented to the `qenable` command, the exit status will be a value of zero.

If the `qenable` command fails to process any operand, the command exits with a value greater than zero.

2.45.7   See Also

The PBS Professional Administrator's Guide and the following manual pages:
`pbs_server(8B), qdisable(8B), and qmgr(8B)`

2.46   qhold

Holds PBS batch jobs

2.46.1   Synopsis

qhold [-h hold_list] job_identifier_list

qhold --version

2.46.2   Description

The `qhold` command requests that a server place one or more holds on a job. A job that has a hold is not eligible for execution. Supported holds: *USER , OTHER* (also known as *operator*), *SYSTEM*, and *bad password*.

A user may place a *USER* hold upon any job the user owns. An operator, who is a user with operator privilege, may place either a *USER* hold or an *OTHER* hold on any job. The batch administrator may place any hold on any job.

The `p` option can only be set by root or admin user via `qhold -h p`. The owning user can release with `qrls -h p` or query by `qselect -h p`.

If no `-h` option is given, the *USER* hold will be applied to the jobs described by the *job_identifier_list* operand list.

If the job identified by *job_identifier_list* is in the queued, held, or waiting states, then all that occurs is that the hold type is added to the job. The job is then placed into the held state if it resides in an execution queue.

If the job is running, then the result of the `qhold` command depends upon whether the job can be checkpointed. The job can be checkpointed if the OS supports checkpointing, or if the application being checkpointed supports checkpointing. See the PBS Professional Administrator's Guide. If the job can be checkpointed, the following happens:

• The job is checkpointed and its execution is interrupted.

• The resources assigned to the job are released.

• The job is placed in the held state in the execution queue.

• The job's Hold_Types attribute is set to *u* for *User Hold*.

If checkpoint / restart is not supported, `qhold` simply sets the job's Hold_Types attribute to *u*. The job continues to execute.

A job's dependency places a system hold on the job. When the dependency is satisfied, the system hold is removed. This system hold is the same as the one set by an administrator. If the administrator sets a system hold on a job with a dependency, then when the dependency is satisfied, the job becomes eligible for execution.

The `qhold` command can be used on job arrays, but not on subjobs or ranges of subjobs.

If the job is in the process of provisioning, it cannot be held.

2.46.3   Options to `qhold`

-h hold_list

> Defines the types of holds to be placed on the job.

> The hold_list argument is a string consisting of one or more of the letters "*u*", "*o*", or "*s*" in any combination or the character "*n*" or "*p*". The hold type associated with each letter is:

Table 2-8: Hold Types

Hold Type	Meaning
u	*USER*
o	*OTHER*
s	*SYSTEM*
n	*None*
p	*Bad password*

--version

> The `qhold` command returns its PBS version information and exits. This option can only be used alone.

2.46.4   Operands

The `qhold` command accepts a *job_identifier_list* which is one or more space-separated job IDs in the form:

sequence_number[.server_name][@server]

Note that some shells require that you enclose a job array identifier in double quotes.

2.46.5   Standard Error

The qhold command will write a diagnostic message to standard error for each error occurrence.

2.46.6   Exit Status

Zero upon successful processing of all the operands.

Greater than zero if the qhold command fails to process any operand.

2.46.7   See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide, qrls(1B), qalter(1B), qsub(1B), pbs_alterjob(3B), pbs_holdjob(3B), pbs_rlsjob(3B), pbs_job_attributes(7B), pbs_resources(7B)

2.47   qmgr

Administrator's command interface for managing PBS

2.47.1   Synopsis

qmgr [-a] [-c directive] [-e] [-n] [-z] [server [server]...]

qmgr import hook <hook_name> <content-type> <content-encoding> {<input_file>|-}

qmgr export hook <hook_name> <content-type> <content-encoding> [<output_file>]

qmgr --version

2.47.2   Description

The PBS manager command, qmgr, provides a command-line interface to the PBS Server. The qmgr command is used to create or delete queues, vnodes, and hooks, to set or change vnode, queue, hook, server, or scheduler attributes, including resources, and to view information about hooks, queues, vnodes, the server, and the scheduler.

For information about attributes, see Chapter 6, "Attributes", on page 309.

Attributes whose values are unset do not appear in the output of the qmgr command.

If `qmgr` is invoked without the `-c` option and standard output is connected to a terminal, `qmgr` writes a prompt to standard output and reads a directive from standard input. See .

For a `qmgr` prompt, type:

> `qmgr <return>`

For syntax and command information, type "`help`" at the `qmgr` prompt.

2.47.2.1   Required Privilege

The `qmgr` command requires different levels of privilege depending on the operation to be performed. All users can list or print attributes. PBS Operator or Manager privilege is required in order to set or change vnode, queue, hook, server, or scheduler attributes. PBS Manager privilege is required in order to create or delete queues, vnodes, and hooks.

Under UNIX/Linux, root privilege is required in order to operate on hooks or the job_sort_formula server attribute. Under Windows, this must be done from the installation account. For domained environments, the installation account must be a local account that is a member of the local Administrators group on the local computer. For standalone environments, the installation account must be a local account that is a member of the local Administrators group on the local computer.

Users without manager or operator privilege cannot view custom resources which were created to be invisible to users.

2.47.2.2   When To Run `qmgr` At Server Host

When operating on hooks or on the job_sort_formula server attribute, the `qmgr` command must be run at the server host.

2.47.3   Options to `qmgr`

The following table lists the options to the `qmgr` command:.

<div align="center">

Table 2-9: `qmgr` Options

</div>

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

Table 2-9: `qmgr` Options

Option	Action
-c directive	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.
server [,server ...]	Makes the specified server(s) active. See <u>section 2.47.6.1, "Making Objects Active", on page 159</u>.
-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.

2.47.4   Directives

A `qmgr` *directive* is a *command* together with the *object* to be operated on, the *attribute* belonging to the object that is to be changed, the *operator*, and the *value* the attribute will take.

A directive is terminated by a newline or a semicolon (";"). Multiple directives may be entered on a single line. A directive may extend across lines by escaping the newline with a backslash ("\").

2.47.4.1   Directive Syntax

A `qmgr` directive takes one of the following forms:

command <object type> [object names] [attribute OP value[,attribute OP value,...]]

import hook <hook_name> <content-type> <content-encoding> {<input_file>|-)

export hook <hook_name> <content-type> <content-encoding> [<output_file>]

The directive can be used from the command line or from within the `qmgr` command. To use a directive from the command line, enclose the command and its arguments in single or double quotes:

qmgr -c "command <object type> [object names] [attribute OP value[,attribute OP value,...]]"

For example:

    qmgr –c 'print queue Q1'

To use a directive from within the qmgr command, first start qmgr:

    qmgr <return>

At the qmgr prompt, enter the command and its arguments. For example:

    Qmgr: print queue Q1

Each command is explained in the following subsections.

2.47.4.2   Comments

Comments begin with the "#" character and continue to the end of the line. Comments and blank lines are ignored by qmgr.

2.47.5   Arguments to Commands

2.47.5.1   Objects

The qmgr command can operate on servers, schedulers, queues, vnodes, and hooks. Each of these can be abbreviated in a directive. The following table lists the objects and their abbreviations:

Table 2-10: qmgr Objects

Object Name	Abbr	Object	Can be Created/ Deleted By:	Can be Modified By:
server	*s*	A Server	No one	Operator, Manager
queue	*q*	A queue	Operator, Manager	Operator, Manager
node	*n*	A vnode	Operator, Manager	Operator, Manager
hook	*h*	A hook	UNIX/Linux: root; Windows: installation account	UNIX/Linux: root; Windows: installation account
sched	*sc*	A scheduler	No one	Operator, Manager

2.47.5.2   Specifying Server

The `qmgr` command operates on objects (queues, vnodes, etc.) at the active server. There is always at least one active server; the default server is the active server unless other servers have been made active.

The default server is the server managing the host where the `qmgr` command runs, meaning it is the server specified in that host's `pbs.conf` file.

You can specify the server you want:

@default

> Specifies the default server.

@<server name>

> Specifies a named server.

@active

> Specifies all active servers.

Server names have the following format:

hostname[:port]

where *hostname* is the fully-qualified domain name of the host on which the server is running and *port* is the port number to which to connect. If *port* is not specified, the default port number is used.

2.47.5.3   Object Names

In a `qmgr` directive, *object names* is a list of one or more names of specific objects. All objects in a list must be of the same type. The name list is in the form:

<object name>[@<server>][,<object name>[@<server>] ...]

where *<server>* is replaced in the directive with "*default*", "*active*", or "*<server name>*". There must be no space between the name and the @ sign.

Example 2-1: List queues workq, slowq, and fastq at the active server:

    Qmgr: list queue workq,slowq,fastq

Example 2-2: List queues Queue1 at the default server, Queue2 at Server2, and Queue3 at the active server:

    Qmgr: list queue Queue1@default,Queue2@Server2,Queue3@active

Name lists must not contain white space between entries.

The administrator specifies the name of an object when creating the object.

Node attributes cannot be used as node names.

2.47.5.4   Specifying Objects

You can specify objects in the following ways:

<object type>

Acts on the active objects of the named type at the active server.

For example, to list all active vnodes, along with their attributes, at the active server:

list node

<object type> @<server> (note space before @ sign)

Acts on the active objects of the named type at the specified server.

For example, to list all active vnodes at the default server, along with their attributes:

list node @default

For example, to print out all queues at the default server, along with their attributes:

qmgr –c "print queue @default"

<object type> <object name>

Acts on the named object.

For example, to list Node1 and its attributes:

list node Node1

<object type> <object name>@<server>

Acts on the named object at the specified server.

For example, to list Node1 at the default server, along with the attributes of Node1:

list node Node1@default

2.47.5.5   Attributes

In a qmgr directive, *attribute* is the name of the attribute belonging to the object on which qmgr is to operate. You can set or modify the value of this attribute.

If the attribute is one which describes a set of resources, then the attribute is specified in the form:

attribute.<resource name>

For example, to set the amount of memory on a vnode:

Qmgr: set node Vnode1 resources_available.mem = 2mb

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."
```

2.47.5.6   Operators

In a `qmgr` directive, *OP* is the operation to be performed with the attribute and its value. Operators are listed here:

Table 2-11: Operators

Operator	Effect
=	Sets the value of the attribute. If the attribute has an existing value, the current value is replaced with the new value.
+=	Increases the current value of the attribute by the amount in the new value. When used for a string array, adds the new value as another string after a comma.
-=	Decreases the current value of the attribute by the amount in the new value. When used for a string array, removes the first matching string.

Example 2-3: Set routing destination for queue Queue1 to be Dest1:

```
Qmgr: set queue route_destinations = Dest1
```

Example 2-4: Add new routing destination for queue Queue1:

```
Qmgr: set queue route_destinations += Dest2
```

Example 2-5: Remove new routing destination for queue Queue1:

```
Qmgr: set queue route_destinations -= Dest2
```

2.47.5.7   Attribute Values

In a `qmgr` directive, *value* is the value to assign to an attribute. An attribute's value must be in the correct format for the attribute's type. Each attribute's type is listed in Chapter 6, "Attributes", on page 309. Each format is described in Chapter 7, "Formats", on page 403.

If the value includes whitespace, commas or other special characters, such as the # character, the value string must be enclosed in single or double quotes.

Resource values can be any string made up of alphanumeric, comma (","), underscore ("_"), dash ("-"), colon (":"), slash ("/"), backslash ("\"), space (" "), and equal sign ("=") characters.

2.47.5.8 Windows Requirements

Under Windows, use double quotes when specifying arguments to qmgr. For example:

```
Qmgr: import hook hook1 application/x-python default "\Documents and
    Settings\pbsuser1\hook1.py"
```

or

```
qmgr -c 'import hook hook1 application/x-python default "\Documents and
    Settings\pbsuser1\hook1.py"'
```

2.47.6 Commands

The qmgr commands can be used in two ways. One is to start qmgr, then use directives at the qmgr prompt. For example, type:

```
qmgr <return>
```

The qmgr prompt appears:

```
Qmgr:
```

Now you can enter commands, using directives, for example:

```
Qmgr: print server
```

The other is to call qmgr with the -c option and a directive in quotes. For example, to enter the same "*print server*" directive:

```
qmgr -c "print server"
```

Commands can be abbreviated to their minimum unambiguous form. Commands apply to all objects unless explicitly limited. The following table lists the commands, briefly tells what they do, and gives a link to a full description:

Table 2-12: qmgr Commands

Command	Abbrev	Effect	Description
active	a	Specifies active objects	See section 2.47.6.1, "Making Objects Active", on page 159
create	c	Creates object	See section 2.47.6.2, "Creating Objects", on page 162

Table 2-12: `qmgr` Commands

Command	Abbrev	Effect	Description
delete	d	Deletes object	See section 2.47.6.3, "Deleting Objects", on page 163
export	e	Exports hook	See section 2.47.6.4, "Exporting Hooks", on page 163
help	h	Prints usage to std-out	See section 2.47.6.10, "Printing Usage Information", on page 168
import	i	Imports hook	See section 2.47.6.5, "Importing Hooks", on page 164
list	l	Lists object attributes and their values	See section 2.47.6.8, "Listing Object Attributes", on page 166
print	p	Prints creation and configuration commands	See section 2.47.6.9, "Printing Creation and Configuration Commands", on page 167
quit	q	Exits the qmgr command	
set	s	Sets value of attribute	See section 2.47.6.6, "Setting Attribute Values", on page 165
unset	u	Unsets value of attribute	See section 2.47.6.7, "Unsetting Attribute Values", on page 166

2.47.6.1   Making Objects Active

Making objects active is a way to set up a list of objects, all of the same type, on which you can then use a single command. Can be used on any object. For example, if you are going to set the same attribute to the same value on several vnodes, you can make all of the target vnodes active before using a single command to set the attribute value, instead of having to give the command once for each vnode.

When an object is active, it is acted upon when you specify its type but do not specify names. When you specify any object names in a directive, active objects are not operated on unless they are named in the directive.

You can specify a list of active objects for each type of object. You can have active objects of multiple types at the same time. The active objects of one type have no effect on whether objects of another type are active.

Objects are active only until the `qmgr` command is exited, so this feature can be used only at the `qmgr` prompt.

Each time you make any objects active, that list of objects replaces any active objects of the same kind. For example, if you have four queues, and you make Q1 and Q2 active, then later make Q3 and Q4 active, the result is that Q3 and Q4 are the only active queues.

You can make objects be active at different servers simultaneously. For example, you can set vnodes N1 and N2 at the default server, and vnodes N3 and N4 at server Server2 to be active at the same time.

To make all objects inactive, quit `qmgr`. When you quit `qmgr`, any object that was active is no longer active.

2.47.6.1.i   Using the `active` Command

active <object type> [<object name>[,<object name> ...]]

Makes the named object(s) of the specified type active.

Example: To make queue Queue1 active:

Qmgr: active queue Queue1

Example: To make queues Queue1 and Queue2 at the active server be active, then enable them:

Qmgr: active queue Queue1,Queue2
Qmgr: set queue enabled=True

Example: To make queue Queue1 at the default server and queue Queue2 at Server2 be active:

Qmgr: active queue Queue1@default,Queue2@Server2

Example: To make vnodes N1, N2, N3, and N4 active, and then give them all the same value for their max_running attribute:

Qmgr: active node N1,N2,N3,N4
Qmgr: set node max_running = 2

active <object type> @<server>   (note space before @ sign)

Makes all object(s) of the specified type at the specified server active.

Example: To make all queues at the default server active:

Qmgr: active queue @default

Example: To make all vnodes at server Server2 active:

Qmgr: active node @Server2

active <object type>

Queries which objects of the specified type are active. The qmgr command prints a list of names of active objects of the specified type to stdout.

2.47.6.2 Creating Objects

create <object type> <object name>[,<object name> ...] [[attribute = value] [,attribute = value] ...]

Creates one new object of the specified type for each name, and gives it the specified name. Can be used only with queues, vnodes, and hooks.

For example, to create a queue named Q1 at the active server:

Qmgr: create queue Q1

For example, to create a vnode named N1 and a vnode named N2:

Qmgr: create node N1,N2

For example, to create queue Queue1 at the default server and queue Queue2 at Server2:

Qmgr: create queue Queue1@default,Queue2@Server2

For example, to create vnodes named N1, N2, N3, and N4 at the active server, and to set their Mom attribute to *Host1* and their max_running attribute to *1*:

Qmgr: create node N1,N2,N3,N4 Mom=Host1, max_running = 1

All objects of the same type at a server must have unique names. For example, each queue at server Server1 must have a unique name. Objects at one server can have the same name as objects at another server.

You can create multiple objects of the same type with a single command. You cannot create multiple types of objects in a single command.

2.47.6.3   Deleting Objects

delete <object type> <object name>[,<object name> ...]

> Deletes the named object(s). Can be used only with queues, vnodes, and hooks. For example, to delete queue Q1 at the active server:

> **Qmgr: delete queue Q1**

> For example, to delete vnodes N1 and N2 at the active server:

> **Qmgr: delete node N1,N2**

> For example, to delete queue Queue1 at the default server and queue Queue2 at Server2:

> **Qmgr: delete queue Queue1@default,Queue2@Server2**

delete <object type>

> Deletes the active objects of the specified type. For example, to delete the active queues:

> **Qmgr: delete queue**

delete <object type> @<server>

> Deletes the active objects of the specified type at the specified server. For example, to delete the active queues at server Server2:

> **Qmgr: delete queue @Server2**

You can delete multiple objects of the same type with a single command. You cannot delete multiple types of objects in a single command.

2.47.6.4   Exporting Hooks

Format for exporting a hook:

export hook <hook name> <content-type> <content-encoding> [<output_file>]

This dumps the script contents of hook <hook_name> into <output_file>, or **stdout** if <output_file> is not specified.

- The resulting *<output_file>* or **stdout** data is of *<content-type>* and *<content-encoding>*.

- The only *<content-type>* currently supported is "*application*/*x-python*".

- The allowed values for *<content-encoding>* are "*default*" (7bit) and "*base64*".

- *<output_file>* must be a path that can be created by **qmgr**.

- Any relative path *<output_file>* is relative to the directory where **qmgr** was executed.

- If *<output_file>* already exists it is overwritten. If PBS is unable to overwrite the file due

to ownership or permission problems, then an error message is displayed in `stderr`.

- If the *<output_file>* name contains spaces like the ones used in Windows file names, then *<output file>* must be enclosed in quotes.

Example 2-6: Dumps hook1's script contents directly into a file "`hello.py.out`":

```
# qmgr –c 'export hook hook1 application/x–python default hello.py'
# cat hello.py
import pbs
pbs.event().job.comment="Hello, world"
```

Example 2-7: To dump the script contents of a hook 'hook1' into a file in "`\My Hooks\hook1.py`":

```
Qmgr: export hook hook1 application/x–python default "\My
    Hooks\hook1.py"
```

2.47.6.5   Importing Hooks

To import a hook, you import the contents of a hook script into the hook. You must specify a filename that is locally accessible to `qmgr` and the PBS Server.

Format for importing a hook:

import hook <hook name> <content-type> <content-encoding> {<input_file>|-}

This uses the contents of *<input file>* or `stdin` (-) as the contents of hook *<hook_name>*.

- The *<input_file>* or `stdin` (-) data must have a format *<content-type>* and must be encoded with *<content-encoding>*.
- The only *<content-type>* currently supported is "*application/x-python*".
- The allowed values for *<content-encoding>* are "*default*" (7bit) and "*base64*".
- If the source of input is `stdin` (-) and *<content-encoding>* is "*default*", then `qmgr` expects the input data to be terminated by EOF.
- If the source of input is `stdin` (-) and *<content-encoding>* is "base64", then `qmgr` expects input data to be terminated by a blank line.
- *<input file>* must be locally accessible to both `qmgr` and the requested batch server.
- A relative path *<input_file>* is relative to the directory where `qmgr` was executed.
- If a hook already has a content script, then that is overwritten by this import call.
- If *<input_file>* name contains spaces as are used in Windows filenames, then *<input*

file> must be quoted.

• There is no restriction on the size of the hook script.

Example 2-8: Given a Python script in ASCII text file "`hello.py`", this makes its contents be the script contents of hook1:

#cat hello.py

import pbs

pbs.event().job.comment="Hello, world"

 # qmgr –c 'import hook hook1 application/x–python default hello.py'

Example 2-9: Given a base64-encoded file "`hello.py.b64`", qmgr unencodes the file's contents, and then makes this the script contents of hook1:

 # cat hello.py.b64

cHJpbnQgImhlbGxvLCB3b3JsZCIK

 # qmgr –c 'import hook hook1 application/x–python base64 hello.py.b64'

2.47.6.6   Setting Attribute Values

You can use the qmgr command to set attributes of any object.

set <object type> <object name>[,<object name> ...] attribute = value [,attribute = value ...]

Sets the value of the specified attribute(s) for the named object(s). Each specified attribute is set for each named object, so if you specify three attributes and two objects, both objects get all three attributes set.

set <object type> attribute = value

Sets the attribute value for all active objects when there are active objects of the type specified.

set <object type> @<server> attribute = value

Sets the attribute value for all active objects at the specified server when there are active objects of the type specified.

You can have spaces between attribute-value pairs.

If a value contains a space or a comma, the value must be enclosed in single or double quotes. For example, to set the value of vnode N1's comment to "*Check later, replacing memory*":

Qmgr: set node N1 comment = "Check later, replacing memory"

You can set attribute values for only one type of object in each command.

2.47.6.7   Unsetting Attribute Values

You can use the `qmgr` command to unset attributes of any object.

unset <object type> <object name>[,<object name> ...] attribute[,attribute...]

Unsets the value of the specified attributes of the named object(s).

unset <object type> attribute[,attribute...]

Unsets the value of specified attributes of active objects.

unset <object type> <object name> attribute[,attribute...]

Unsets the value of specified attributes of the named object.

unset <object type> @<server> attribute[,attribute...]

Unsets the value of specified attributes of active objects at the specified server.

You can have spaces between attribute names.

You can unset attribute values for only one type of object in each command.

2.47.6.8   Listing Object Attributes

You can use the `qmgr` command to list attributes of any object.

list <object type> <object name>[,<object name> ...]

Lists the attributes, with associated values, of the named object(s).

list <object type> <object name> <attribute name>[, <attribute name>]...

Lists values of the specified attributes of the named object.

list <object type>

Lists attributes, with associated values, of active objects of the specified type at the active server.

list <object type> @<server>

Lists all objects of the specified type at the specified server, with their attributes and the values associated with the attributes.

list server

Lists attributes of the active server. If no server other than the default server has been made active, lists attributes of the default server (it is the active server).

list server <server>

Lists attributes of the specified server.

list hook

    Lists all hooks, along with their attributes.

list hook <hook name>

    Lists attributes of the specified hook.

2.47.6.9   Printing Creation and Configuration Commands

print <object type> <object name>[,<object name> ...]

    where *<object name>* follows the name rules in <u>section 2.47.5.3, "Object Names", on page 155</u>.

    Prints out the commands required to do the following:

- Create the named object(s)
- Set object attributes to their current values

print <object type> <object name> [<attribute name>[, <attribute name>]...]

    where *<object name>* follows the name rules in <u>section 2.47.5.3, "Object Names", on page 155</u>.

    Prints out the commands required to do the following:

- Create the named object
- Set specified object attributes to their current values

print <object type>

    Prints out the commands to create and configure the active objects of the named type.

print <object type> @<server>

    Prints out the commands to create and configure all of the objects of the specified type at the specified server.

print server

    Prints information for the active server; if there is no active server, prints information for the default server.

    Prints out the commands required to do the following for the server and queues, but not hooks:

- Create each queue
- Set the attributes of each queue to their current values

• Set the attributes of the server to their current values

print hook

Prints out the commands to create and configure all hooks.

print hook <hook name>

Prints out the commands to create and configure the specified hook.

2.47.6.10   Printing Usage Information

The `qmgr` built-in help function is invoked using the `help` command. You can request usage information for any of the `qmgr` commands.

help <command>

Prints out usage information for the specified command.

For example, to print usage information for the `set` command:

```
qmgr
Qmgr: help set
Syntax: set object [name][,name...] attribute[.resource] OP value
```

2.47.7   Saving and Re-creating Configuration

To save and recreate a configuration, print the configuration information to a file, then read it back in later.

See .

2.47.7.1   Saving Queue Information

Before re-creating queue and server configuration, use this command to save the configuration/creation commands to a file:

```
Qmgr: print server > savedsettings
```

When re-creating the queue and server configuration, read the commands into `qmgr`:

```
qmgr < savedsettings
```

2.47.7.2   Saving Hook Information

To save creation and configuration information for all hooks:

```
# qmgr -c "print hook" > hook.qmgr
```

To re-create all hooks:

```
# qmgr < hook.qmgr
```

2.47.8   Standard Input

The qmgr command reads standard input for directives until end-of-file is reached, or the exit or quit directive is read.

2.47.9   Standard Output

If standard output is connected to a terminal, a command prompt is written to standard output when qmgr is ready to read a directive.

If the -e option is specified, qmgr will echo the directives read from standard input to standard output.

2.47.10   Standard Error

If the -z option is not specified, the qmgr command writes a diagnostic message to standard error for each error occurrence.

2.47.11   Exit Status

Upon successful processing of all the operands presented to the qmgr command, the exit status is zero.

If the qmgr command fails to process any operand, the command exits with a value greater than zero.

2.47.12   Caveats

2.47.12.1   Setting Vnode Attributes

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. See section 3.5.2, "Choosing Configuration Method" on page 48 in the PBS Professional Administrator's Guide.

2.47.13   Examples

The following are examples of qmgr directives:

List serverA's scheduler's attributes

```
list sched @serverA
```

List attributes for default server's scheduler

```
l sched @default
```

List PBS version for default server's scheduler

```
l sched @default pbs_version
```

Set software resource on mynode

```
set node mynode resources_available.software = "myapp=/tmp/foo"
```

Create queue

```
create queue fast priority=10,queue_type=e,enabled = true,max_running=0
```

Increase limit on queue

```
set queue fast max_running +=2
```

Create queue, set resources

```
create queue little
set queue little resources_max.mem=8mw,resources_max.cput=10
```

Unset limit on queue

```
unset queue fast max_running
```

Set node offline

```
set node state = "offline"
```

Define active list

```
active server s1,s2,s3
```

List a queue

```
list queue @server1
```

Set limit on queue

```
set queue max_running = 10
```

To create a provisioning hook called Provision_Hook, and import the ASCII hook script called "master_provision.py" located in /root/data/:

```
Qmgr: create hook Provision_Hook
Qmgr: import hook Provision_Hook application/x-python default /root/
      data/master_provision.py
```

2.47.14　See Also

The PBS Professional Administrator's Guide, the PBS Professional Installation and Upgrade Guide, `pbs_queue_attributes(7B)`, `pbs_server_attributes(7B)`, `pbs_node_attributes(7B)`, `pbs_hook_attributes(7B)`, `pbs_sched_attributes(7B)`

2.48　qmove

Moves PBS batch job

2.48.1　Synopsis

qmove destination job_identifier ...

qmove --version

2.48.2　Description

To move a job is to remove the job from the queue in which it resides and place the job in another queue.

The `qmove` command can be used on job arrays, but not on subjobs or ranges of subjobs.

Note that job arrays can only be moved from one server to another if they are in the '*Q*', '*H*', or '*W*' states, and only if there are no running subjobs. The state of the job array is preserved, and the job array will run to completion on the new server.

A job in the *Running*, *Transiting*, or *Exiting* state cannot be moved.

A job in the process of provisioning cannot be moved.

The behavior of the `qmove` command may be affected by any site hooks. Site hooks can modify the job's attributes, change its routing, etc.

2.48.3   Effect of Privilege on Behavior

An unprivileged user can use the qmove command to move a job only when the move would not violate queue restrictions. A privileged user (root, Manager, Operator) can use the qmove command to move a job under some circumstances where an unprivileged user cannot. The restrictions that apply only to unprivileged users are listed here:

• The queue must be enabled

• Moving the job into the queue must not exceed the queue's limits for jobs or resources

• If the job is an array job, the size of the job array must not exceed the queue's max_array_size

• If the queue is accepting jobs only from routing queues, unprivileged users cannot move jobs into it using the qmove command

2.48.4   Options

--version

The qmove command returns its PBS version information and exits. This option can only be used alone.

2.48.5   Operands

The first operand is the new destination for the jobs. It will be accepted in the syntax:

queue

@server

queue@server

See [Chapter 7, "Formats", on page 403](#) for destination identifier information.

If the destination operand describes only a queue, then qmove will move jobs into the queue of the specified name at the job's current server.

If the destination operand describes only a batch server, then qmove will move jobs into the default queue at that batch server.

If the destination operand describes both a queue and a batch server, then qmove will move the jobs into the specified queue at the specified server.

All following operands are *job_identifiers* which specify the jobs to be moved to the new destination . The qmove command accepts one or more job_identifier operands of the form:

sequence_number[.server_name][@server]

Note that some shells require that you enclose a job array identifier in double quotes.

2.48.6 Standard Error

The `qmove` command will write a diagnostic messages to standard error for each error occurrence.

2.48.7 Exit Status

Upon successful processing of all the operands presented to the `qmove` command, the exit status will be a value of zero.

If the `qmove` command fails to process any operand, the command exits with a value greater than zero.

2.48.8 See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide, `qsub(1B), pbs_movejob(3B)`

2.49 qmsg

Sends message to PBS batch jobs

2.49.1 Synopsis

qmsg [-E] [-O] message_string job_identifier ...

qmsg --version

2.49.2 Description

To send a message to a job is to write a message string into one or more output files of the job. Typically this is done to leave an informative message in the output of the job.

The `qmsg` command writes messages into the files of jobs by sending a Message Job batch request to the batch server that owns the job. The `qmsg` command does not directly write the message into the files of the job.

The qmsg command cannot be used on job arrays, subjobs or ranges of subjobs.

2.49.3   Options

-E

    Specifies that the message is written to the standard error of each job.

-O

    Specifies that the message is written to the standard output of each job.

--version

    The qmsg command returns its PBS version information and exits. This option can only be used alone.

If no option is specified, the message will be written to the standard error of the job.

2.49.4   Operands

The first operand, *message_string* , is the message to be written. If the string contains blanks, the string must be quoted. If the final character of the string is not a newline, a newline character will be added when written to the job's file.

All following operands are *job_identifiers* which specify the jobs to receive the message string. The qmsg command accepts one or more job_identifier operands of the form:

sequence_number[.server_name][@server]

2.49.5   Standard Error

The qmsg command will write a diagnostic message to standard error for each error occurrence.

2.49.6   Exit Status

Upon successful processing of all the operands presented to the qmsg command, the exit status will be a value of zero.

If the qmsg command fails to process any operand, the command exits with a value greater than zero.

2.49.7   See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide, `qsub(1B)`, `pbs_msgjob(3B)`

2.50   qorder

Exchanges order of two PBS batch jobs.

2.50.1   Synopsis

qorder job_identifier job_identifier

qorder --version

2.50.2   Description

Allows the exchange of two jobs' positions in the queue or queues in which the jobs reside.

No attribute of the job, e.g. priority, is changed. The impact of interchanging the order within or between queues is dependent on local job scheduling policy; contact your systems administrator.

2.50.3   Restrictions

• A job in the running state cannot be reordered.

• The `qorder` command can be used on job arrays, but not on subjobs or ranges of subjobs.

• The two jobs must be located at the same server.

2.50.4   Effect of Privilege on Behavior

For an unprivileged user to reorder jobs, both jobs must be owned by the user. A privileged user (Manager, Operator) can reorder any jobs.

2.50.5   Options

--version

> The qorder command returns its PBS version information and exits. This option can only be used alone.

2.50.6   Operands

Both operands are *job_identifiers* which specify the jobs to be exchanged. The qorder command accepts two *job_identifier* operands of the form:

sequence_number[.server_name][@server]

The server specification for the two jobs must agree as to the current location of the two job ids.

Note that some shells require that you enclose a job array identifier in double quotes.

2.50.7   Standard Error

The qorder command will write diagnostic messages to standard error for each error occurrence.

2.50.8   Exit Status

Upon successful processing of all the operands presented to the qorder command, the exit status will be a value of zero.

If the qorder command fails to process any operand, the command exits with a value greater than zero.

2.50.9   See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide,
qsub(1B), qmove(1B), pbs_orderjob(3B), pbs_movejob(3B)

2.51   qrerun

Reruns a PBS batch job

2.51.1 Synopsis

qrerun [-W force] job_identifier [job_identifier ...]

qrerun --version

2.51.2 Description

The `qrerun` command reruns the specified jobs if possible. PBS Manager or Operator privilege is required to use this command.

To rerun a job is to kill it and requeue it in the execution queue from which it was run.

If a job is marked as not rerunnable then `qrerun` will fail. See the `-r` option on the `qsub` and `qalter` commands.

The `qrerun` command can be used on job arrays, subjobs, and ranges of subjobs. It cannot rerun a subjob which is not running.

2.51.3 Options

-W force

The job is to be requeued even if the node on which the job is executing is unreachable, or if the job's substate is *provisioning*.

--version

The `qrerun` command returns its PBS version information and exits. This option can only be used alone.

2.51.4 Operands

The `qrerun` command accepts one or more job_identifier operands of the form:

sequence_number[.server_name][@server]

Note that some shells require that you enclose a job array identifier in double quotes.

2.51.5 Standard Error

The `qrerun` command will write a diagnostic message to standard error for each error occurrence.

2.51.6   Exit Status

Zero upon successful processing of all operands.

Greater than zero upon failure to process any operand.

2.51.7   See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide,
qsub(1B), qalter(1B), pbs_alterjob(3B), pbs_rerunjob(3B)

2.52   qrls

Releases hold on PBS batch jobs

2.52.1   Synopsis

qrls [-h hold_list] job_identifier ...

qrls --version

2.52.2   Description

The qrls command removes or releases holds which exist on batch jobs.

A job may have one or more types of holds which make the job ineligible for execution. The
types of holds are *USER , OTHER , SYSTEM*, and *bad password*. The different types of
holds may require that the user issuing the qrls command have special privilege. Typically,
the owner of the job will be able to remove a *USER* hold, but not an *OTHER* or *SYSTEM*
hold. An Attempt to release a hold for which the user does not have the correct privilege is an
error and no holds will be released for that job.

If no -h option is specified, the *USER* hold will be released.

Only root or admin can set a bad password hold via qhold -h p. The owner of the job can
qrls -h p a hold set with qhold -h p.

If the job has no execution_time pending, the job will change to the *queued* state. If an
execution_time is still pending, the job will change to the *waiting* state.

2.52.3   Options

-h hold_list

Defines the types of hold to be released from the jobs. The hold_list option argument is a string consisting of one or more of the letters *u* , *o* , or *s* in any combination, or one or more of the letters *n* or *p*. The hold type associated with each letter is:

Table 2-13: Hold Types

Hold Type	Meaning
u	*USER*
o	*OTHER*
s	*SYSTEM*
n	*None*
p	*Bad password*

--version

The `qrls` command returns its PBS version information and exits. This option can only be used alone.

2.52.4   Operands

The `qrls` command accepts one or more job_identifier operands of the form:

sequence_number[.server_name][@server]

Note that some shells require that you enclose a job array identifier in double quotes.

2.52.5   Standard Error

The `qrls` command will write a diagnostic message to standard error for each error occurrence.

2.52.6　　Exit Status

Upon successful processing of all the operands presented to the `qrls` command, the exit status will be a value of zero.

If the `qrls` command fails to process any operand, the command exits with a value greater than zero.

2.52.7　　See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide, `qsub(1B)`, `qalter(1B)`, `qhold(1B)`, `pbs_alterjob(3B)`, `pbs_holdjob(3B)`, and `pbs_rlsjob(3B)`

2.53　　qrun

Runs a PBS batch job now

2.53.1　　Synopsis

qrun [-a] [-H vnode-specification] job_identifier_list

qrun --version

2.53.2　　Description

The `qrun` command is used to force a job to run, regardless of scheduling position or resource requirements.

In order to execute `qrun`, the user must have PBS Operator or Manager privilege, and the job must be in the *Queued* state and reside in an execution queue.

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.

The `qrun` command cannot be used on a job that is in the process of provisioning.

NOTE: If you use a -H vnode_specification option to run a job, but specify insufficient vnodes or resources, the job may not run correctly. Avoid using this option unless you are sure.

When preemption is enabled, the scheduler preempts other jobs in order to run this job. Running a job via qrun gives the job higher preemption priority than any other class of job.

2.53.3   Options to `qrun`

-a

> The `qrun` command exits before the job actually starts execution.

(no -H option)

> A request is made of the Scheduler to schedule this job. The job is run immediately regardless of scheduling policy as long as the following are true:
>
> • The queue in which the job resides is an execution queue.
>
> • 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.
>
> The `qrun` command alone overrides the following:
>
> • Limits on resource usage by users, groups, and projects
>
> • Limits on the number of jobs that can be run at a vnode
>
> • Boundaries between primetime and non-primetime, specified in backfill_prime
>
> • Whether the job is in a primetime queue: you can run a job in a prime-time queue even when it's not primetime, or vice versa. Primetime boundaries are not honored.
>
> • Dedicated time: you can run a job in a dedicated time queue, even if it's not in a dedicated time queue, and vice versa. However, dedicated time boundaries are still honored.
>
> The `qrun` command alone does not override the following:
>
> • Server and queue resource usage limits

(with -H option)

> With the -H option, all scheduling policies are bypassed and the job is run directly. The job will be run immediately on the named vnodes, regardless of current usage on those vnodes with the exception of vnode state. The job will not be run and the `qrun` request will be rejected if any named vnode is down, offline, already allocated exclusively or would need to be allocated exclusively and another job is already running on the vnode.
>
> If the `qrun -H` command is used on a job that requests an AOE, and that AOE is not instantiated on those vnodes, the vnodes are provisioned with the AOE.
>
> If the job requests an AOE, and that AOE is not available on the specified vnodes, the job is held.

-H vnode_specification, without resources

The vnode_specification without resources has this format:

(vchunk)[+(vchunk) ...]

where vchunk has the format

vnode[+vnode ..]

Example:

```
-H (VnodeA+VnodeB)+(VnodeC)
```

PBS will apply one requested chunk from the job's selection directive in round-robin fashion to each vchunk in the list. Each vchunk must be sufficient to run the job's corresponding chunk, otherwise the job may not execute correctly.

-H vnode_specification, with resources

The vnode_specification with resources has this format:

(vchunk)[+(vchunk) ...]

where vchunk has the format

vnode:vnode_resources[+vnode:vnode_resources ...]

and where *vnode_resources* has the format

resource=value[:resource=value ...]

Example:

```
-H (VnodeA:mem=100kb:ncpus=1)
  +(VnodeB:mem=100kb:ncpus=2+VnodeC:mem=100kb)
```

PBS creates a new selection directive from the vnode_specification, using it instead of the original specification from the user. Any single resource specification will result in the job's original selection directive being ignored. Each vchunk must be sufficient to run the job's corresponding chunk, otherwise the job may not execute correctly.

If the job being run requests -l place=exclhost, take extra care to satisfy the exclhost request. Make sure that if any vnodes are from a multi-vnoded host, all vnodes from that host are allocated. Otherwise those vnodes can be allocated to other jobs.

--version

The qrun command returns its PBS version information and exits. This option can only be used alone.

2.53.4   Operands

The `qrun` command accepts a *job_identifier_list* containing one or more job_identifiers of the form:

sequence_number[.server_name][@server]

Note that some shells require that you enclose a job array identifier in double quotes.

2.53.5   Standard Error

The `qrun` command will write a diagnostic message to standard error for each error occurrence.

2.53.6   Exit Status

Zero, on success.

Greater than zero, if the `qrun` command fails to process any operand.

2.53.7   See Also

The PBS Professional Administrator's Guide, `qsub(1B), qmgr(8B), pbs_runjob(3B)`

2.54   qselect

Selects PBS batch jobs

2.54.1   Synopsis

qselect [-a [op]date_time] [-A account_string] [-c [op]interval] [-h hold_list] [-H] [-J] [-l resource_list] [-N name] [-p [op] priority] [-P project] [-q destination] [-r rerun] [-s states] [-t suboption [comparison] specified-time] [-T] [-u user_list] [-x]

qselect --version

2.54.2   Description

The `qselect` command lists those jobs that meet the specified selection criteria. Jobs are selected from a single server.

Each option acts as a filter restricting which jobs are listed. With no options, the `qselect` command will list all jobs at the server which the user is authorized to list (query status of).

When selecting jobs according to their requested resources, this command can be used only on resources in the Resource_List job attribute, or on the entire selection directive.

Jobs that are finished or moved are listed only when the -x or -H options are used. Otherwise, job selection is limited to queued and running jobs.

2.54.3　　Relations

When an option is specified with a optional op component to the option argument, then op specifies a relation between the value of a certain job attribute and the value component of the option argument. If an op is allowable on an option, then the description of the option letter will indicate the op is allowable. The only acceptable strings for the op component, and the relation the string indicates, are shown in the following table of relations:

Table 2-14: Relations

Relation	Meaning
.eq.	the value represented by the attribute of the job is equal to the value represented by the option argument.
.ne.	the value represented by the attribute of the job is not equal to the value represented by the option argument.
.ge.	the value represented by the attribute of the job is greater than or equal to the value represented by the option argument.
.gt.	the value represented by the attribute of the job is greater than the value represented by the option argument.
.le.	the value represented by the attribute of the job is less than or equal to the value represented by the option argument.
.lt.	the value represented by the attribute of the job is less than the value represented by the option argument.

2.54.4   Options to `qselect`

-a [op]date_time

> **Deprecated**. Restricts selection to a specific execution time, or a range of execution times.
>
> The `qselect` command selects only jobs for which the value of the Execution_Time attribute is related to the date_time argument by the optional op operator.
>
> The date_time argument has the format:
>
> *[[CC]YY]MMDDhhmm[.SS]*
>
> where the MM is the two digits for the month, DD is the day of the month, hh is the hour, mm is the minute, and the optional SS is the seconds. CC is the century and YY the year.
>
> If op is not specified, jobs will be selected for which the Execution_Time and date_time values are equal. If op is specified, jobs will be selected according to the definitions given in Relations above.

-A account_string

> Restricts selection to jobs whose Account_Name attribute matches the specified account_string .

-c [op]interval

> Restricts selection to jobs whose Checkpoint interval attribute matches the specified relationship.
>
> The values of the Checkpoint attribute are defined to have the following ordered relationship:
>
> *n > s > c=minutes > c > u*
>
> If the optional op is not specified, jobs will be selected whose Checkpoint attribute is equal to the interval argument. If op is specified, jobs will be selected according to the rules in Relations above.
>
> For an interval value of "*u*", only ".*eq*." and ".*ne*." are valid.

-h hold_list

> Restricts the selection of jobs to those with a specific set of hold types. Only those jobs will be selected whose Hold_Types attribute exactly match the value of the hold_list argument.
>
> The hold_list argument is a string consisting of the single letter n , or one or more of the letters *u, o, p*, or *s* in any combination. If letters are duplicated,

they are treated as if they occurred once. The letters represent the hold types:

Table 2-15: Hold Types

Hold Type	Meaning
n	none
u	user
o	other
p	bad password
s	system

-H

Restricts selection to finished and moved jobs.

-J

Limits the selection to job arrays only.

-l resource_list

Restricts selection of jobs to those with specified resource amounts. Users without operator or manager privilege cannot specify custom resources which were created to be invisible to users.

The resource_list is in the following format:

resource_name op value[,resource_name op val,...]

The relation operator op must be present.

For job-wide resources, all operators are useful. However, resource specifications for chunks using the select statement, or placement using the place statement are stored as strings. Therefore the only useful operators for these are *.eq.* and *.ne.*

The definitions given in Relations above are used when comparing the values of resources.

-N name

Restricts selection of jobs to those with a specific name.

-p [op]priority

Restricts selection of jobs to those with a priority that matches the specified relationship. If op is not specified, jobs are selected for which the job Priority attribute is equal to the priority

If the op is specified, the relationship is defined in Relations above.

-P project

Restrict selection of jobs to those matching the specified project.

Format: String.

Project name can contain any characters except for the following: Slash ("/"), left bracket ("["), right bracket ("]"), double quote ("""), semicolon (";"), colon (":"), vertical bar ("|"), left angle bracket ("<"), right angle bracket (">"), plus ("+"), comma (","), question mark ("?"), and asterisk ("*").

-q destination

Restricts selection to those jobs residing at the specified destination.

The destination may be of one of the following three forms:

queue

@server

queue@server

If the -q option is not specified, jobs will be selected from the default server.

If the destination describes only a queue, only jobs in that queue on the default batch server will be selected.

If the destination describes only a server, then jobs in all queues on that server will be selected.

If the destination describes both a queue and a server, then only jobs in the named queue on the named server will be selected.

-r rerun

Restricts selection of jobs to those with the specified Rerunnable attribute. The option argument must be a single character. The following two characters are supported by PBS: *y* and *n* .

-s states

Restricts job selection to those in the specified states.

The states argument is a character string which consists of any combination of the characters: *B, E, F, H, M, Q, R, S, T, U, W*, and *X*. (A repeated character will be accepted, but no additional meaning is assigned to it.)

Table 2-16: Job States

State	Meaning
B	Job array has started execution.
E	The *Exiting* state.
F	The *Finished* state.
H	The *Held* state.
M	The *Moved* state.
Q	The *Queued* state.
R	The *Running* state.
S	The *Suspended* state.
T	The *Transiting* state.
U	Job suspended due to workstation user activity.
W	The *Waiting* state.
X	The *eXited* state. Subjobs only.

Jobs will be selected which are in any of the specified states. Since array jobs are never in states *R, S, T,* or *U*, if those states are specified, no array job will be selected. Subjobs of the array in those states may be selected if -*T* is specified.

-t suboption [comparison] specified-time

>Allows jobs to be selected according to their time attributes. The suboption is one of:

Table 2-17: Suboptions to the –t Option

Sub option	Time Attribute Selected	Meaning
a	Execution_Time	Time the job began execution
c	ctime	Job creation time, seconds since epoch
e	etime	Time the job became eligible to run
g	eligible_time	Amount of eligible time job accrued waiting to run
m	mtime	Modification time
q	qtime	Job queued time
s	stime	Job start time
t	estimated.start_time	Job's estimated start time

>The comparison is one of the relations listed above in Relations.

>The specified-time is in datetime format. See Chapter 7, "Formats", on page 403.

>A time period can be bracketed by using the -t option twice. For example, to select jobs using stime between noon and 3 p.m.:

>```
>qselect –ts.gt.09251200 –ts.lt.09251500
>```

-T

>Limits selection to job and subjob identifiers.

-u user_list

>Restricts selection to jobs owned by the specified user names.

>This provides a means of limiting the selection to jobs owned by one or more users.

>The syntax of the user_list is:

_user_name[@host][,user_name[@host],...]_

Host names may be wild carded on the left end, e.g. "*.nasa.gov". User_name without a "@host" is equivalent to "user_name@*", that is at any host. Jobs will be selected which are owned by the listed users at the corresponding hosts.

-x

Allows selection of finished and moved jobs in addition to queued and running jobs.

-X

Allows selection of completed or deleted subjobs (subjobs in _X_ state).

--version

The qselect command returns its PBS version information and exits. This option can only be used alone.

2.54.5   Standard Output

The list of job identifiers of selected jobs is written to standard output. Each job identifier is separated by white space. Each job identifier is of the form:

_sequence_number.server_name@server_

Where sequence_number.server is the identifier assigned at submission time; see qsub. _@server_ identifies the server which currently owns the job.

2.54.6   Standard Error

The qselect command will write a diagnostic message to standard error for each error occurrence.

2.54.7   Exit Status

Upon successful processing of all options presented to the qselect command, the exit status will be a value of zero.

If the qselect command fails to process any option, the command exits with a value greater than zero.

2.54.8   See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide,
`qalter(1B)`, `qdel(1B)`, `qhold(1B)`, `qmove(1B)`, `qrls(1B)`,
`qstat(1B)`, `qsub(1B)`, `pbs_job_attributes(7B)`, `pbs_resources(7B)`

2.55   qsig

Send signal to PBS batch job

2.55.1   Synopsis

qsig [-s signal] job_identifier ...

qsig --version

2.55.2   Description

The `qsig` command requests that a signal be sent to the specified executing batch jobs. The
signal is sent to the session leader of the job.

If the `-s` option is not specified, `SIGTERM' is sent.

* The request to signal a batch job is rejected if:
* The user is not authorized to signal the job.
* The job is not in the running state.
* The requested signal is not supported by the system upon which the job is executing.
* The job is in the process of provisioning

The `qsig` command sends a Signal Job batch request to the server which owns the job.

The `qsig` command can be used for job arrays, ranges of subjobs, and subjobs. If it is used
on a range of subjobs, the subjobs in the range which are running will be signaled.

2.55.3   Options

-s signal
        Declares which signal is sent to the job.

The signal argument is either a signal name, e.g. SIGKILL, the signal name without the SIG prefix, e.g. *KILL*, or an unsigned signal number, e.g. *9*. The signal name SIGNULL is allowed; the server will send the signal 0 to the job which will have no effect. Not all signal names will be recognized by `qsig` signal name; try issuing the signal number instead.

Two special signal names, "*suspend*" and "*resume*", [note, all lower case], are used to suspend and resume jobs. When suspended, a job continues to occupy system resources but is not executing and is not charged for wall-time. Manager or operator privilege is required to suspend or resume a job.

If `qsig -s resume` is used on a job that was suspended using `qsig -s suspend`, the job will be resumed when there are sufficient resources.

--version

The `qsig` command returns its PBS version information and exits. This option can only be used alone.

2.55.4 Operands

The `qsig` command accepts one or more *job_identifier* operands. For a job, this has the form:

sequence_number[.server_name][@server]

and for a job array, it is:

sequence_number[][.server_name][@server]

Note that some shells require that you enclose a job array identifier in double quotes.

2.55.5 Standard Error

The `qsig` command will write a diagnostic messages to standard error for each error occurrence.

2.55.6 Exit Status

Upon successful processing of all the operands presented to the `qsig` command, the exit status will be a value of zero.

If the `qsig` command fails to process any operand, the command exits with a value greater than zero.

2.55.7   See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide, `qsub(1B)`, `pbs_sigjob(3B)`, `pbs_resources(7B)`

2.56   qstart

Allows PBS jobs to be run from a queue

2.56.1   Synopsis

qstart destination ...

qstart --version

2.56.2   Description

The `qstart` command directs that a destination queue should process batch jobs. If the destination is an execution queue, the scheduler will begin to schedule jobs that reside in the queue for execution. If the destination is a routing queue, the server will begin to route jobs from that queue.

In order to execute `qstart`, the user must have PBS Operator or Manager privilege.

2.56.3   Options

--version

     The `qstart` command returns its PBS version information and exits. This option can only be used alone.

2.56.4   Operands

The `qstart` command accepts one or more destination operands. The operands are one of three forms:

queue

@server

queue@server

If *queue* is specified, the request is to start that queue at the default server. If the *@server* form is given, the request is to start all queues at that server. If a full destination identifier, *queue@server*, is given, the request is to start the named queue at the named server.

2.56.5   Standard Error

The `qstart` command will write a diagnostic message to standard error for each error occurrence.

2.56.6   Exit Status

Upon successful processing of all the operands presented to the `qstart` command, the exit status will be a value of zero.

If the `qstart` command fails to process any operand, the command exits with a value greater than zero.

2.56.7   See Also

The PBS Professional Administrator's Guide and the following manual pages:
`pbs_server(8B), qstop(8B), and qmgr(8B)`

2.57   qstat

Displays status of PBS batch jobs, queues, or servers

2.57.1   Synopsis

2.57.1.1   Displaying Job Status

Default format:

qstat [-p] [-J] [-t] [-x] [[job_identifier | destination] ...]

Long format:

qstat -f [-p] [-J] [-t] [-x] [[job_identifier | destination] ...]

Alternate format:

qstat [-a [-w]| -H | -i | -r] [-G | -M] [-J] [-n [-1][-w]] [-s [-1][-w]] [-t] [-u user_list] [[job_identifier | destination] ...]

2.57.1.2 Displaying Queue Status

Default format:

qstat -Q [destination ...]

Long format:

qstat -Q -f [destination ...]

Alternate format:

qstat -q [-G | -M] [destination ...]

2.57.1.3 Displaying Server Status

Default format:

qstat -B [server_name ...]

Long format:

qstat -B -f [server_name ...]

2.57.1.4 Displaying Version Information

qstat --version

2.57.2 Description

The `qstat` command is used to display the status of jobs, queues, and batch servers. The status information is written to standard output.

Status information can be displayed in a default format, an alternate format, or a long format, depending upon the options given. Default and alternate formats display all status information for a job, queue or server on one line, in columns. Long formats display status information one attribute to a line.

Status information for finished and moved jobs can be displayed using the -x and -H options.

When displaying job status information, the `qstat` command will display status information about all *job_identifiers* and destinations specified.

If your job has been moved to another server through peer scheduling, give the job ID as an argument to `qstat`. If you only give the `qstat` command, your job will not appear to exist. For example, your 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

If your default server is ServerB, and your job started at ServerA but was moved to ServerB, to see the job, you must use:

qstat 123@ServerA

Users without manager or operator privilege cannot view resources or attributes that are invisible to unprivileged users.

2.57.3 Displaying Job Status

2.57.3.1 Job Status in Default Format

The `qstat` command will display job status in default format when the options given are among -p, -J, -t or -x, regardless of operands. Jobs are displayed one to a line, with these column headers:

```
Job id   Name       User      Time Use S Queue
-------- ---------- --------- -------- - -----
```

Description of columns:

Table 2-18: Description of Default Job Status Columns

Column	Description
Job id	The job_identifier assigned by PB
Name	Job name assigned by submitter.
User	Username of job owner.

Table 2-18: Description of Default Job Status Columns

Column	Description	
Time Use	The CPU time or walltime used by the job, depending on which is specified or inherited. Before the application has actually started running, for example during stage-in, this field is "0". At the point where the application starts accumulating cput or walltime, this field changes to "00:00:00". After that, every time the MOM polls for resource usage, the field is updated.	
	The MOM on the each execution host polls for the usage of all processes belonging to the job on her host. Usage is summed. The polling interval is short when a job first starts running and lengthens to a maximum 2 minutes. See section 3.6.1, "Configuring MOM's Polling Cycle" on page 53 in the PBS Professional Administrator's Guide.	
S	The job's state:	
	B	Array job has at least one subjob running.
	E	Job is exiting after having run.
	F	Job is finished.
	H	Job is held.
	M	Job was moved to another server.
	Q	Job is queued.
	R	Job is running.
	S	Job is suspended.
	T	Job is being moved to new location.
	U	Cycle-harvesting job is suspended due to keyboard activity.
	W	Job is waiting for its submitter-assigned start time to be reached.
	X	Subjob has completed execution or has been deleted.
Queue	The queue in which the job resides.	

2.57.3.2   Job Status in Long Format

If the -f (full) option is given, full job status information for each job is displayed in this order:

- The job ID
- Each job attribute, one to a line
- The job's submission arguments
- The job's executable, in JSDL format
- The executable's argument list, in JSDL format

The job attributes are listed as *name = value* pairs. This includes the exec_host string and the exec_vnode string. The full output can be very large.

The exec_host string has the format:

hosta/J1+hostb/J2*P+...

where *J1* and *J2* are an index of the job on the named host and *P* is the number of processors allocated from that host to this job. *P* does not appear if it is *1*.

The exec_vnode string has the format:

(vnodeA:ncpus=N1:mem=M1)+(vnodeB:ncpus=N2:mem=M2)+...

where *N1* and *N2* are the number of CPUs allocated to that job on that vnode, and *M1* and *M2* are the amount of memory allocated to that job on that vnode.

2.57.3.3   Job Status in Alternate Format

The qstat command will display job status in the alternate format if any of the -a, -i, -G, -H, -M, -n, -r, -s, or -u user_list options is given. Jobs are displayed one to a line. If jobs are running and the -n option is specified, or if jobs are finished or moved and the -H and -n options are specified, there is a second line for the exec_host string.

2.57.3.3.i   Output:

The output contains the following columns:

```
                                          Req'd  Req'd   Elap
   Job ID Username Queue Jobname SessID NDS TSK Memory Time  S Time
   ------ -------- ----- ------- ------ --- --- ------ ----- - ----
```

If -n is specified, the column output is followed by the exec_host string.

Description of columns:

Table 2-19: Description of Alternate Format Job Columns

Column	Description
Job ID	The *job_identifier* assigned by PBS.
Username	Username of job owner.
Queue	Queue in which the job resides.
Jobname	Job name assigned by submitter.
SessID	Session ID. Only appears if the job is running.
NDS	Number of chunks or nodes requested by the job.
TSK	Number of CPUs requested by the job.
Req'd Memory	Amount of memory requested by the job.
Req'd Time	If CPU time is requested, this shows CPU time. Otherwise, shows walltime.
S	The job's state. (See listing above.)
Elap Time	If CPU time is requested, this shows CPU time. Otherwise, shows walltime.

2.57.4   Displaying Queue Status

2.57.4.1   Queue Status in Default Format

The `qstat` command will display queue status in the default format if the only option is `-Q`, regardless of operands. Queue status is displayed one queue to a line, with these column headers:

```
Queue        Max  Tot  Ena  Str Que  Run  Hld  Wat  Trn  Ext  Type

----------- ---- ---- ---- --- ---- ---- ---- ---- ---- ---- ----
```

Description of columns:

Table 2-20: Description of Default Queue Status Columns

Column	Description
Queue	Queue name.
Max	Maximum number of jobs allowed to run concurrently in the queue.
Tot	Total number of jobs in the queue.
Ena	Whether the queue is enabled or disabled.
Str	Whether the queue is started or stopped.
Que	Number of queued jobs.
Run	Number of running jobs.
Hld	Number of held jobs.
Wat	Number of waiting jobs.
Trn	Number of jobs being moved (transiting.)
Ext	Number of exiting jobs.
Type	Type of queue: execution or routing.

2.57.4.2　Queue Status in Long Format

If the -f (full) option is given, full queue status information for each queue is displayed starting with the queue name, followed by each attribute, one to a line, as *name = value* pairs.

2.57.4.2.i　Queue Status: Alternate Format

The qstat command will display queue status in the alternate format if any of the -q, -G or -M options is given. Queue status is displayed one queue to a line, with these column headers:

```
Queue   Memory CPU Time Walltime Node Run Que Lm State

------- ------ -------- -------- ---- --- --- -- -----
```

Description of columns:

Table 2-21: Description of Queue Alternate Status Columns

Column	Description
Queue	Queue name.
Memory	Maximum amount of memory that can be requested by a job in the queue.
CPU Time	Maximum amount of CPU time that can be requested by a job in the queue.
Walltime	Maximum amount of wall time that can be requested by a job in the queue.
Node	Maximum number of nodes that can be requested by a job in the queue.
Run	Number of running jobs. Lowest row is total number of running jobs in all the queues shown.
Que	Number of queued jobs. Lowest row is total number of queued jobs in all the queues shown.
Lm	Maximum number of jobs allowed to run concurrently in the queue.
State	State of the queue: *E* (enabled) or *D* (disabled), and *R* (running) or *S* (stopped).

2.57.5 Displaying Server Status

2.57.5.1 Server Status in Default Format:

The `qstat` command will display server status if the only option given is -B, regardless of operands.

Column headers for default server status:

```
Server   Max   Tot   Que   Run   Hld   Wat   Trn   Ext   Status

-------- ----- ----- ----- ----- ----- ----- ----- ----- ------
```

Description of columns:

Table 2-22: Description of Server Status Default Display Columns

Column	Description
Server	Name of the server.
Max	Maximum number of jobs allowed concurrently running on the server.
Tot	Total number of jobs currently managed by the server.
Que	Number of queued jobs.
Run	Number of running jobs.
Hld	Number of held jobs.
Wat	Number of waiting jobs.
Trn	Number of transiting jobs.
Ext	Number of exiting jobs.
Status	Status of the server.
Server	Status in Long Format:

2.57.5.2   Server Status in Long Format

If the -f (full) option is given, full server status information is displayed starting with the server name, followed by each attribute, one to a line, as *name = value* pairs. PBS version information is listed.

2.57.6   Options to `qstat`

2.57.6.1   Default Job Status Options

-J

Limits status information to job arrays.

-t

Displays status information for jobs, job arrays, and subjobs. When used with -J option, limits status information to subjobs.

-p

The Time Use column is replaced with the percentage completed for the job. For an array job this is the percentage of subjobs completed. For a normal job, it is the larger of percentage used walltime or percentage used CPU time. Default format used.

-x

Displays status information for finished and moved jobs in addition to running and queued jobs.

2.57.6.2 Alternate Job Status Options

The following options will cause the alternate job status format to be used:

-a

All queued and running jobs are displayed. If a destination is given, information for all jobs at that destination is displayed. If a *job_identifier* is given, information about that job is displayed. Always specify this option before the -n or -s options, otherwise they will not take effect.

-H

Without a job identifier, displays information for all finished or moved jobs. If a job identifier is given, displays information for that job regardless of its state.

-i

If a destination is given, information for queued, held or waiting jobs at that destination is displayed. If a *job_identifier* is given, information about that job is displayed regardless of its state.

-r

If a destination is given, information for running or suspended jobs at that destination is displayed. If a *job_identifier* is given, information about that job is displayed regardless of its state.

-T

Displays estimated start time for queued jobs, replacing the *Elap Time* field with the *Est Start* field. Jobs with earlier estimated start times are displayed before those with later estimated start times.

Running jobs are displayed before other jobs. Running jobs are sorted by their stime attribute (start time).

Queued jobs whose estimated start times are unset (estimated.start_time = *unset*) are displayed after those with estimated start times, with estimated start time shown as a double dash ("--"). Queued jobs with estimated start times in the past are treated as if their estimated start times are unset.

Time displayed is local to the `qstat` command. Current week begins on Sunday.

The following table shows the format used without the `-w` option:

Table 2-23: Format for Estimated Start Time Field without -w Option

Format	Job's Estimated Start Time	Example
HH:MM	Today	`15:34`
<2-letter weekday> HH	Within 7 days, but after today	`We 15`
<3-letter month name>	This calendar year, but after this week	`Feb`
YYYY	Less than or equal to 5 years from today, after this year	`2012`
">5yrs"	More than 5 years from today	`>5yrs`

The following table shows the format used with the `-w` option:

Table 2-24: Format for Estimated Start Time Field with -w Option

Format	Job's Estimated Start Time	Example
Today HH:MM	Today	`Today 13:34`

Table 2-24: Format for Estimated Start Time Field with -w Option

Format	Job's Estimated Start Time	Example
Day HH:MM	This week, but after today	`Wed 15:34`
Day Mon Daynum HH:MM	This year, but after this week	`Wed Feb 10 15:34`
Day Mon Daynum Year HH:MM	After this year	`Wed Feb 10 2011 15:34`

When used with the -f option, prints the full timezone-qualified start time.

If a job's estimated start time cannot be calculated, the start time is shown as a question mark ("?").

Estimated start time information can be made unavailable to unprivileged users; in this case, the estimated start time appears to be unset.

-u user_list

If a destination is given, status for jobs at that destination owned by users in user_list is displayed. If a *job_identifier* is given, status information for that job is displayed regardless of the job's ownership.

Hostnames may be wildcarded, but not domain names. When no hostname is specified, username is for any host.

Format: *username[@host]* in comma-separated list.

-n

The exec_host string is listed on the line below the basic information. If the -1 option is given, the exec_host string is listed on the end of the same line. If using the -a option, always specify the -n option after -a otherwise the -n option will not take effect.

-s

Any comment added by the administrator or scheduler is shown on the line below the basic information. If the -1 option is given, the comment string is listed on the end of the same line. If using the -a option, always specify the -s option after -a otherwise the -s option will not take effect.

-w

> Allows display of wider fields. User name, Queue and Job name can be up
> to 15 characters wide. Session ID can be up to 8 characters wide and NDS
> can be up to 4 characters wide. Can only be used with -a, -n or -s.

-1

> Reformats qstat output to a single line. Can only be used in conjunction
> with the -n and/or -s options.

2.57.6.3   Queue Status Options

-Q

> Display queue status in default format. Operands must be destinations.

-q

> Display queue status in alternate format. Operands must be destinations.

2.57.6.4   Server Status Options

-B

> Display server status. Operands must be names of servers.

2.57.6.5   Job, Queue, and Server Status Options

-f

> Full display. Job, queue or server attributes displayed one to a line.

-G

> Show size in gigabytes. Alternate format is used.

-M

> Show size in megawords. A word is considered to be 8 bytes. Alternate
> format is used.

2.57.6.6   Version Information

--version

> The qstat command returns its PBS version information and exits. This
> option can only be used alone.

2.57.7 Operands

2.57.7.1 Job Identifier Operands

job_identifier

Job identifier assigned by PBS at submission. Only used with job status requests. Status information for this job is displayed.

2.57.7.2 Destination Operands

Name of queue, name of queue at a specific server, or specification of server.

Name of queue:

<queue name>

Name of queue at server:

<queue name>@<server>

All queues at a server:

@<server>

When displaying job status:

- If *<queue name>* is given, status is displayed for all jobs in the named queue at the default server.
- If *<queue name>@<server>* is given, status is displayed for all jobs in queue_name at server.
- If *@<server>* is given, status is displayed for all jobs at all queues at that server.

When displaying queue status:

- If *<queue name>* is given, status is displayed for that queue at the default server.
- If *<queue name>@<server>* is given, status is displayed for the named queue at the named server.
- If *@<server>* is given, status is displayed for all queues at that server.

2.57.7.3 Server Name Operands

server_name

Name of server. Used with the -B option to display status for that server.

2.57.8   Standard Error

The `qstat` command writes a diagnostic message to standard error for each error occurrence.

2.57.9   Exit Status

- Zero upon successful processing of all the operands.
- Greater than zero if any operands could not be processed.
- Non-zero if `x` option is not provided when querying finished jobs.

2.57.10   See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide,
`qalter(1B)`, `qsub(1B)`, `pbs_alterjob(3B)`, `pbs_statjob(3B)`,
`pbs_statque(3B)`, `pbs_statserver(3B)`, `pbs_submit(3B)`,
`pbs_job_attributes(7B)`, `pbs_queue_attributes(7B)`,
`pbs_server_attributes(7B)`, `pbs_resources(7B)`

2.58   qstop

Prevents PBS jobs from running from the specified queue

2.58.1   Synopsis

qstop destination ...

qstop --version

2.58.2   Description

The `qstop` command directs that a destination queue should stop processing batch jobs. If the destination is an execution queue, the server will cease scheduling jobs that reside in the queue for execution. If the destination is a routing queue, the server will cease routing jobs from that queue.

In order to execute `qstop`, the user must have PBS Operator or Manager privilege.

2.58.3   Options

--version

The `qstop` command returns its PBS version information and exits. This option can only be used alone

2.58.4   Operands

The `qstop` command accepts one or more destination operands. The operands are one of three forms:

<queue>

@<server>

<queue>@<server>

If *<queue>* is specified, the request is to stop that queue at the default server. If the *@<server>* form is given, the request is to stop all the queues at that server. If a full destination identifier, *<queue>@<server>*, is given, the request is to stop the named queue at the named server.

2.58.5   Standard Error

The `qstop` command will write a diagnostic message to standard error for each error occurrence.

2.58.6   Exit Status

Upon successful processing of all the operands presented to the `qstop` command, the exit status will be a value of zero.

If the `qstop` command fails to process any operand, the command exits with a value greater than zero.

2.58.7   See Also

The PBS Professional Administrator's Guide and the following manual pages:
pbs_server(8B), qstart(8B), and qmgr(8B)

2.59 qsub

Submits PBS job

2.59.1 Synopsis

qsub [-a date_time] [-A account_string] [-c interval] [-C directive_prefix] [-e path] [-h] [-I] [-j join] [-J range] [-k keep] [-l resource_list] [-m mail_events] [-M user_list] [-N name] [-o path] [-p priority] [-P project] [-q destination] [-r c] [-S path_list] [-u user_list] [-v variable_list] [-V] [-W additional_attributes] [-X] [-z] [script | -- executable [arglist for executable]]

qsub --version

2.59.2 Description

The `qsub` command is used to submit a batch job to PBS. Submitting a PBS job specifies a task, requests resources and sets job attributes.

The `qsub` command can read from a job script, from standard input, or from the command line. When the user has submitted the job, PBS returns the job identifier for that job. For a job, this is of the form:

sequence_number.servername

For an array job, this is of the form:

sequence_number[].servername

During execution, jobs can be interactive or non-interactive.

2.59.2.1 Where PBS Puts Job Files

By default, PBS copies the `stdout` and `stderr` files from the job back to the current working directory where the `qsub` command is executed. See the `-o` and `-e` options.

2.59.2.2 Submitting Jobs By Using Scripts

To submit a PBS job script, the user types

    qsub [options] scriptname

Scripts can be written in Python, UNIX shells such as `csh` and `sh`, the Windows command batch language, Perl, etc. The same Python script can be run under UNIX/Linux or under Windows. A PBS job script consists of the following:

- Optional shell specification

- Any PBS directives

- The user's tasks: programs, commands or applications

Example 2-10: A Python job script named "myjob.py" for a job named "HelloJob" that prints "Hello" under UNIX/Linux or Windows:

```
#PBS -l select=1:ncpus=3:mem=1gb
#PBS -N HelloJob
print "Hello"
```

To run a Python job script under UNIX/Linux:

```
qsub -S $PBS_EXEC/bin/pbs_python <script name>
```

To run a Python job script under Windows:

```
qsub -S %PBS_EXEC%\bin\pbs_python.exe <script name>
```

Example 2-11: A script named "weatherscript" for a job named "Weather1" which runs the executable "weathersim" on UNIX/Linux:

```
#!/bin/sh
#PBS -N Weather1
#PBS -l walltime=1:00:00
/usr/local/weathersim
```

To submit the job, the user types:

```
qsub weatherscript <return>
```

Example 2-12: A script named "weather.exe" for a job named "Weather1" which runs under Windows:

```
#PBS -N Weather1
#PBS -l walltime=1:00:00
weathersim.exe
```

To submit the job, the user types:

```
qsub weather.exe <return>
```

Scripts can contain comments. Under Windows, comments can contain only ASCII characters. See the PBS Professional User's Guide.

2.59.2.3 Submitting Jobs From Standard Input

To submit a PBS job by typing job specifications at the command line, the user types

> **qsub [options] <return>**

then types any directives, then any tasks, followed by

- UNIX: **CTRL-D** on a line by itself
- Windows: **CTRL-Z <return>**

to terminate the input.

2.59.2.4 Submitting a Job From the **qsub** Command Line

To submit a job from the command line, the user types

> **qsub [options] -- executable [arguments to executable] <return>**

Example 2-13: To run **myprog** with the arguments **a** and **b**:

> **qsub -- myprog a b <return>**

Example 2-14: To run **myprog** with the arguments **a** and **b**, naming the job JobA:

> **qsub -N JobA -- myprog a b <return>**

2.59.2.5 Requesting Resources and Placing Jobs

Requesting resources includes setting limits on resource usage and controlling how the job is placed on nodes.

Resources are requested by using the -l option, either in chunks inside of selection statements, or in job-wide requests using *resource_name=value* pairs. See the **pbs_resources**(7B) man page. The selection statement is of the form:

-l select=[N:]chunk[+[N:]chunk ...]

where *N* specifies how many of that chunk, and a chunk is of the form:

resource_name=value[:resource_name=value ...]

Job-wide resource_name=value requests are of the form:

-l resource_name=value[,resource_name=value ...]

The place statement has this form:

-l place=[arrangement][: sharing][: grouping]

where

    arrangement

        one of *free* | *pack* | *scatter* | *vscatter*

    sharing

        one of *excl* | *shared* | *exclhost*

    grouping

        can have only one instance of *group=resource*

and where

    free

        Place job on any vnode(s).

    pack

        All chunks will be taken from one host.

    scatter

        Only one chunk with any MPI processes will be taken from a host. A chunk with no MPI processes may be taken from the same node as another chunk.

    vscatter

        Only one chunk is taken from any vnode. Each chunk must fit on a vnode.

    excl

        Only this job uses the vnodes chosen.

    shared

        This job can share the vnodes chosen.

    exclhost

        The entire host is allocated to the job.

    group=resource

        Chunks will be grouped according to a resource. All nodes in the group must have a common value for the resource, which can be either the built-in resource host or a site-defined node-level resource.

        Resource must be a string or a string array.

Note that nodes can have sharing attributes that override job placement requests. See the `pbs_node_attributes(7B)` man page.

For more on resource requests, usage limits and job placement, see `pbs_resources(7B)`.

2.59.2.6 Caveats

Do not mix old style resource or node specifications with the new *select* and *place* statements. Do not use one in a job script and the other on the command line. Mixing the two will result in an error.

You cannot submit a job requesting a custom resource which has been created to be invisible or read-only for users, regardless of your privilege. A manager or operator can use the qalter command to change a job's request for this kind of custom resource.

2.59.2.7 Setting Attributes

The user sets job attributes by giving options to the qsub command or by using PBS directives. Each qsub option except -C, -q, and -z sets a job attribute, and has a corresponding PBS directive with the same syntax as the option. Attributes set via command-line options take precedence over those set using PBS directives. See the PBS Professional User's Guide, or section 6.11, "Job Attributes", on page 375.

2.59.2.8 Changing `qsub` Behavior

The behavior of the qsub command may be affected by the server's default_qsub_arguments attribute. This attribute can set the default for any job attribute. The default_qsub_arguments server attribute is settable by the administrator, and is over-ridden by command-line arguments and script directives. See section 6.6, "Server Attributes", on page 314.

The behavior of the qsub command may also be affected by any site hooks. Site hooks can modify the job's attributes, change its routing, etc.

2.59.3 Options to `qsub`

-a date_time

Point in time after which the job is eligible for execution. Given in pairs of digits. Sets job's Execution_Time attribute to date_time.

Format: datetime:

[[[[CC]YY]MM]DD]hhmm[.SS]

where *CC* is the century, *YY* is the year, *MM* is the month, *DD* is the day of the month, *hh* is the hour, *mm* is the minute, and *SS* is the seconds.

Each portion of the date defaults to the current date, as long as the next-smaller portion is in the future. For example, if today is the 3rd of the

month and the specified day *DD* is the 5th, the month *MM* is set to the current month.

If a specified portion has already passed, the next-larger portion is set to one after the current date. For example, if the day *DD* is not specified, but the hour hh is specified to be 10:00 a.m. and the current time is 11:00 a.m., the day *DD* is set to tomorrow.

-A account_string

Accounting string associated with the job. Used for labeling accounting data. Sets job's Account_Name attribute to account_string.

Format: string.

-c checkpoint_spec

Determines when the job will be checkpointed. Sets job's Checkpoint attribute. An $action script is required to checkpoint the job.

See the pbs_mom(8B) man page.

The argument checkpoint_spec can take on one of the following values:

c
Checkpoint at intervals, measured in CPU time, set on job's execution queue. If no interval set at queue, job is not checkpointed

c=<minutes of CPU time>
Checkpoint at intervals of specified number of minutes of job CPU time. This value must be > 0. If interval specified is less than that set on job's execution queue, queue's interval is used.

Format: Integer

w
Checkpoint at intervals, measured in walltime, set on job's execution queue. If no interval set at queue, job is not checkpointed.

w=<minutes of walltime>
Checkpoint at intervals of the specified number of minutes of job walltime. This value must be greater than zero. If the interval specified is less that that set on the execution queue in which the job resides, the queue's interval is used.

Format: Integer

n
No checkpointing.

s
Checkpoint only when the server is shut down.

u

    Unset. Defaults to behavior when interval argument is set to s.

Default: u.

Format: String.

-C directive_prefix

Defines the prefix identifying a PBS directive. Default prefix is "*#PBS*".

If the directive_prefix argument is a null string, qsub does not scan the script file for directives. Overrides the PBS_DPREFIX environment variable and the default. Cannot be used as a PBS directive.

-e path

Path to be used for the job's standard error stream. Sets job's Error_Path attribute to path. The path argument is of the form:

[hostname:]path_name

The path is interpreted as follows:

path_name

    If *path_name* is a relative path, then it is taken to be relative to the current working directory of the qsub command, where it is executing on the current host.

    If *path_name* is an absolute path, then it is taken to be an absolute path on the current host where the qsub command is executing.

hostname:path_name

    If *path_name* is a relative path, then it is taken to be relative to the user's home directory on the host named hostname.

    If *path_name* is an absolute path, then it is the absolute path on the host named hostname.

    If *path_name* does not include a filename, the default filename is

    jobid.ER

If the -e option is not specified, PBS copies the standard error to the current working directory where the qsub command was executed. The default filename for the standard error stream is used. It has this form:

job_name.e<sequence number>

-h

Applies a user hold to the job. Sets the job's Hold_Types attribute to "*u*".

-I

Job is to be run interactively. Sets job's interactive attribute to *True*. The job is queued and scheduled as any PBS batch job, but when executed, the

standard input, output, and error streams of the job are connected to the terminal session in which `qsub` is running. If a job script is given, only its directives are processed. When the job begins execution, all input to the job is taken from the terminal session. See the PBS Professional User's Guide for additional information on interactive jobs.

Interactive jobs are not rerunnable.

Job arrays cannot be interactive.

When used with `–Wblock=true`, no exit status is returned.

Not supported on Windows.

-j join

Whether and how to join the job's standard error and standard output streams. Sets job's Join_Path attribute to join.

Possible values of join:

Table 2-25: Suboptions to Join Option

Suboption	Meaning
oe	Standard error and standard output are merged into standard output.
eo	Standard error and standard output are merged into standard error.
n	Standard error and standard output are not merged.

Default: not merged.

-J range

Declares that this job is an array job. Sets job's array attribute to *True*. The argument range identifies the integers greater than or equal to zero that are associated with the subjobs of the array. *range* is specified in the form *X-Y[:Z]* where *X* is the first index, *Y* is the upper bound on the indices and *Z* is the stepping factor. For example, 2-7:2 will produce indices of 2, 4, and 6. If *Z* is not specified, it is taken to be 1.

-k keep

Specifies whether and which of the standard output and standard error streams is retained on the execution host. Overrides default path names for these streams. Sets the job's Keep_Files attribute to keep.

Default: neither is retained.

In the case where output and/or error is retained on the execution host in a job-specific staging and execution directory created by PBS, these files are deleted when PBS deletes the directory.

The keep argument can take on the following values:

Table 2-26: Suboptions to keep Option

Suboption	Meaning
e	The standard error stream is retained on the execution host, in the job's staging and execution directory. The filename is *job_name.e<sequence number>*
o	The standard output stream is retained on the execution host, in the job's staging and execution directory. The filename is *job_name.o<sequence number>*
eo, oe	Both standard output and standard error streams are retained on the execution host, in the job's staging and execution directory.
n	Neither stream is retained.

-l resource_list

Allows the user to request resources and specify job placement. Sets job's Resource_list attribute to resource_list. Requesting a resource places a limit on its usage.

Requesting resources in chunks:

-l select=[N:]chunk[+[N:]chunk ...]

where N specifies how many of that chunk, and a chunk is:

resource_name=value[:resource_name=value ...]

Requesting job-wide resources:

-l resource_name=value[,resource_name=value ...]

Specifying placement of jobs:

-l place=modifier[:modifier]

where modifier is any combination of *group*, *excl*, and/or one of *free|pack|scatter*.

For more on resource requests, usage limits and job placement, see pbs_resources(7B).

-m mail_events

The set of conditions under which mail about the job is sent. Sets job's Mail_Points attribute to mail_events. The mail_events argument can be either "*n*" or any combination of *a*, *b*, and *e*.

Table 2-27: Suboptions to m Option

Suboption	Meaning
n	No mail will be sent.
a	Mail is sent when the job is aborted by the batch system.
b	Mail is sent when the job begins execution.
e	Mail is sent when the job terminates.

Format: string.

Default value: "a".

-M user_list

List of users to whom mail about the job is sent. Sets job's Mail_Users attribute to user_list.

The user_list argument is of the form:

user[@host][,user[@host],...]
Default: job owner.

-N name

Sets job's Job_Name attribute and name to name.

Format: string, up to 15 characters in length. It must consist of an alphabetic or numeric character followed by printable, nonwhite-space characters.

Default: if a script is used to submit the job, the job's name is the name of the script. If no script is used, the job's name is "*STDIN*".

-o path

Path to be used for the job's standard output stream. Sets job's Output_Path attribute to path. The path argument is of the form:

[hostname:]path_name
The path is interpreted as follows:

path_name

If *path_name* is a relative path, then it is taken to be relative to the current working directory of the command, where it is executing on the current host.

If *path_name* is an absolute path, then it is taken to be an absolute path on the current host where the command is executing.

hostname:path_name

If *path_name* is a relative path, then it is taken to be relative to the user's home directory on the host named hostname.

If *path_name* is an absolute path, then it is the absolute path on the host named hostname.

If *path_name* does not include a filename, the default filename is

jobid.OU

If the -o option is not specified, PBS copies the standard output to the current working directory where the qsub command was executed. The default filename for the standard output stream is used. It has this form:

job_name.o<sequence number>

-p priority

Priority of the job. Sets job's Priority attribute to priority.

Format: host-dependent integer.

Range: [-1024, +1023] inclusive.

Default: *zero*.

-P project

Specifies a project for the job. Sets job's project attribute to specified value.

Format: String.

Project name can contain any characters except for the following: Slash ("/"), left bracket ("["), right bracket ("]"), double quote ("""), semicolon (";"), colon (":"), vertical bar ("|"), left angle bracket ("<"), right angle bracket (">"), plus ("+"), comma (","), question mark ("?"), and asterisk ("*").

Default value: "*_pbs_project_default*".

-q destination

Where the job is sent upon submission.

Specifies a queue, a server, or a queue at a server. The destination argument can have one of these formats:

<queue>

Job is submitted to the named queue at the default server.

@*<server>*

   Job is submitted to the default queue at the named server.

<queue>@<server>

   Job is submitted to the named queue at the named server.

Default: default queue at default server.

-r y|n

Declares whether the job is rerunnable. See the qrerun(1B) command.

Sets job's Rerunnable attribute to the argument.

Format: single character, "*y*" or "*n*".

Table 2-28: Suboptions to r Option

Suboption	Meaning
y	Job is rerunnable.
n	Job is not rerunnable.

Default: "*n*".

Interactive jobs are not rerunnable.

-S path_list

Specifies the interpreter or shell path for the job script. Sets job's
Shell_Path_List attribute to path_list.

 The path_list argument is the full path to the interpreter or shell including
the executable name.

Only one path may be specified without a host name. Only one path may be
specified per named host. The path selected is the one whose host name is
that of the server on which the job resides.

Format:

path[@host][,path@host ...]
Default: user's login shell on execution node.

Example of using bash via a directive:

   #PBS –S /bin/bash@mars,/usr/bin/bash@jupiter

Example of running a Python script from the command line on UNIX/
Linux:

   qsub –S $PBS_EXEC/bin/pbs_python <script name>

Example of running a Python script from the command line on Windows:

```
qsub –S %PBS_EXEC%\bin\pbs_python.exe <script name>
```

-u user_list

List of usernames. Job is run under a username from this list. Sets job's User_List attribute to user_list.

Only one username may be specified without a host name. Only one user-name may be specified per named host. The server on which the job resides will select first the username whose host name is the same as the server name. Failing that, the next selection will be the username with no speci-fied hostname. The usernames on the server and execution hosts must be the same. The job owner must have authorization to run as the specified user.

Format of user_list:

user[@host][,user@host ...]
Default: job owner (username on submit host.)

-v variable_list

Lists environment variables to be exported to the job. This is the list of environment variables which is added to those already automatically exported. These variables exist in the user's login environment from which qsub is run. The job's Variable_List attribute is appended with the vari-ables in variable_list and their values. See section 2.59.7, "Environment Variables", on page 227.

Format: comma-separated list of strings in the form:

variable

or

variable=value

If a *variable=value* pair contains any commas, the value must be enclosed in single or double quotes, and the *variable=value* pair must be enclosed in the kind of quotes not used to enclose the value. For example:

```
qsub –v "var1='A,B,C,D'" job.sh

qsub –v a=10, "var2='A,B'", c=20, HOME=/home/zzz job.sh
```

Default: no environment variables are added to job's variable list.

-V

Declares that all environment variables in the user's login environment where qsub is run are to be exported to the job. The job's Variable_List attribute is appended with all of these environment variables and their val-ues.

-W additional_attributes

The -W option allows specification of any job attribute. Some job attributes must be specified using this option. Those attributes are listed below. Format:

-W attribute_name=value[,attribute_name=value...]

If white space occurs within the additional_attributes argument, or the equal sign "=" occurs within an attribute_value string, then that must be enclosed with single- or doublequotes.

The following attributes must be set using the -W option:

depend=dependency_list

Defines dependencies between this and other jobs. Sets the job's depend attribute to dependency_list. The dependency_list has the form:

type:arg_list[,type:arg_list ...]

where except for the *on* type, the *arg_list* is one or more PBS job IDs in the form:

jobid[:jobid ...]

The type can be:

after: arg_list

> This job may be scheduled for execution at any point after all jobs in *arg_list* have started execution.

afterok: arg_list

> This job may be scheduled for execution only after all jobs in *arg_list* have terminated with no errors. See "Warning about exit status with csh" in Exit Status.

afternotok: arg_list

> This job may be scheduled for execution only after all jobs in *arg_list* have terminated with errors. See section 2.59.8.1, "Warning About Exit Status with csh", on page 229.

afterany: arg_list

> This job may be scheduled for execution after all jobs in *arg_list* have finished execution, with any exit status (with or without errors.) This job will not run if a job in the *arg_list* was killed.

before: arg_list

> Jobs in *arg_list* may begin execution once this job has begun execution.

beforeok: arg_list

> Jobs in *arg_list* may begin execution once this job terminates without errors. See "Warning about exit status with csh" in Exit Status.

beforenotok: arg_list

> If this job terminates execution with errors, then jobs in *arg_list* may begin. See .

beforeany: arg_list

> Jobs in *arg_list* may begin execution once this job terminates execution, with or without errors.

on: count

> This job may be scheduled for execution after *count* dependencies on other jobs have been satisfied. This type is used in conjunction with one of the *before* types listed. *count* is an integer greater than 0.

Job IDs in the *arg_list* of *before* types must have been submitted with a type of *on*.

To use the *before* types, the user must have the authority to alter the jobs in *arg_list*. Otherwise, the dependency is rejected and the new job aborted.

Error processing of the existence, state, or condition of the job on which the newly submitted job is a deferred service, i.e. the check is performed after the job is queued. If an error is detected, the new job will be deleted by the server. Mail will be sent to the job submitter stating the error.

Dependency examples:

```
qsub -W depend=afterok:123.host1.domain.com /tmp/script

qsub -W depend=before:234.host1.com:235.host1.com /tmp/script
```

group_list=g_list

> List of group names. Job is run under a group name from this list. Sets job's group_List attribute to g_list.

> Only one group name may be specified without a host name. Only one group name may be specified per named host. The server on which the job resides will select first the group name whose host name is the same as the server name. Failing that, the next selection will be the group

name with no specified hostname. The group names on the server and execution hosts must be the same.

Under Windows, the primary group is the first group found for the user by PBS when it queries the accounts database.

Format of g_list:

group[@host][,group@host ...]
Default: login group name of job owner.

pwd
pwd=
pwd=
These forms prompt the user for a password. A space between W and pwd is optional. Spaces between the quotes are optional. Examples:

```
qsub ... -Wpwd <return>

qsub ... -W pwd='' <return>

qsub ... -W pwd=" " <return>
```

Available on Windows and supported Linux x86 and x86_64 platforms only.

block=true
Specifies that qsub waits for the job to terminate, then returns the job's exit value. Sets job's block attribute to *TRUE*. When used with X11 forwarding or interactive jobs, no exit value is returned. See section 2.59.8, "Exit Status", on page 229.

sandbox=<value>
Determines which directory PBS uses for the job's staging and execution. If value is *PRIVATE* , PBS creates a job-specific directory for staging and execution. If value is *HOME* or is unset, PBS uses the user's home directory for staging and execution.

stagein=path_list
stageout=path_list
Specifies files or directories to be staged-in before execution or staged-out after execution is complete. Sets the job's stagein and stageout attributes to the specified path_lists. On completion of the job, all staged-in and staged-out files and directories are removed from the execution host(s). The path_list has the form:

filespec[,filespec]
where *filespec* is

local_path@hostname:remotepath

regardless of the direction of the copy. The name *local_path* is the name of the file or directory on the primary execution host. It can be relative to the staging and execution directory on the execution host, or it can be an absolute path.

The "@" character separates *local_path* from *remote_path*.

The name *remote_path* is the path on *hostname*. The name can be relative to the staging and execution directory on the primary execution host, or it can be an absolute path.

If path_list has more than one *filespec*, i.e. it contains commas, it must be enclosed in double-quotes.

umask=NNNN

The umask with which the job is started. Sets job's umask attribute to NNNN. Controls umask of job's standard output and standard error.

The following example allows group and world read on the job's output:

```
-W umask=33
```

Can be used with one to four digits; typically two.

Default value: *077*

-X

Allows user to receive X output from interactive job.

DISPLAY variable in submission environment must be set to desired display.

Can be used with interactive jobs only: must be used with -I or -W interactive=true.

Cannot be used with -v DISPLAY.

When used with -Wblock=true, no exit status is returned.

Can be used with -V option.

Not available under Windows.

-z

Job identifier is not written to standard output.

--version

The qsub command returns its PBS version information and exits. This option can only be used alone.

2.59.4   Operands

The qsub command accepts as operands one of the following:

(script)

> Path to script. Can be absolute or relative to current directory where qsub is run.

-

> (a dash)

> Any PBS directives and user tasks are read from the command line. Same as for no operands.

-- executable [arguments to executable]

> a single executable (preceded by two dashes) and its arguments

> The executable, and any arguments to the executable, are given on the qsub command line. The executable is preceded by two dashes, "--".

If a script or executable is specified, it must be the last argument to qsub. The arguments to an executable must follow the name of the executable.

2.59.5   Standard Output

Unless the -z option is set, the job identifier assigned to the job is written to standard output if the job is successfully created.

2.59.6   Standard Error

The qsub command writes a diagnostic message to standard error for each error occurrence.

2.59.7   Environment Variables

The qsub command uses the following environment variables:

PBS_DEFAULT

> Name of default server.

PBS_DPREFIX

> Prefix string which identifies PBS directives.

Environment variables beginning with "*PBS_O_*" are created by qsub. PBS automatically exports the following environment variables to the job, and the job's Variable_List attribute is set to this list:

PBS_ENVIRONMENT
> Set to *PBS_BATCH* for a batch job. Set to *PBS_INTERACTIVE* for an interactive job. Created upon execution.

PBS_JOBDIR
> Pathname of job's staging and execution directory on the primary execution host.

PBS_JOBID
> Job identifier given by PBS when the job is submitted. Created upon execution.

PBS_JOBNAME
> Job name given by user. Created upon execution.

PBS_NODEFILE
> Name of file containing the list of nodes assigned to the job. Created upon execution.

PBS_O_HOME
> User's home directory. Value of HOME taken from user's submission environment.

PBS_O_HOST
> Name of submit host. Value taken from user's submission environment.

PBS_O_LANG
> Value of LANG taken from user's submission environment.

PBS_O_LOGNAME
> User's login name. Value of LOGNAME taken from user's submission environment.

PBS_O_MAIL
> Value of MAIL taken from user's submission environment.

PBS_O_PATH
> User's PATH. Value of PATH taken from user's submission environment.

PBS_O_QUEUE
> Name of the queue to which the job was submitted. Value taken from user's submission environment.

PBS_O_SHELL
> Value taken from user's submission environment.

PBS_O_SYSTEM
> Operating system, from `uname -s`, on submit host. Value taken from user's submission environment.

PBS_O_TZ
> Value taken from user's submission environment.

PBS_O_WORKDIR
> Absolute path to directory where `qsub` is run. Value taken from user's submission environment.

PBS_QUEUE
> Name of the queue from which the job is executed. Created upon execution.

TMPDIR
> Pathname of job's scratch directory.

2.59.8   Exit Status

Zero upon successful processing of input. Exit value is greater than zero upon failure of `qsub`.

For blocking jobs, `qsub` exits and returns the exit value of the job. If the job is deleted without being run, `qsub` returns an exit value of *3*.

2.59.8.1   Warning About Exit Status with `csh`

If a job is run in csh and a .logout file exists in the home directory in which the job executes, the exit status of the job is that of the `.logout` script, not the job script. This may impact any inter-job dependencies.

2.59.9   See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide,
`pbs_job_attributes(7B), pbs_server_attributes(7B), pbs_resources(7B), qalter(1B), qhold(1B), qmove(1B), qmsg(1B), qrerun(1B), qrls(1B),  qselect(1B), qstat(1B)`

2.60   qterm

Terminates a PBS server

2.60.1   Synopsis

qterm [-f | -F | -i] [-m] [-s] [-t type] [server[server ...]]

qterm --version

2.60.2   Description

The qterm command terminates a PBS batch server.

Once the server is terminating, no new jobs are accepted by the server, and no jobs are allowed to begin execution. The impact on running jobs depends on the way the server is shut down.

The qterm command does not exit until the server has completed its shutdown procedure.

If the complex is configured for failover, and the primary server is shut down, the normal behavior for the secondary server is to become active. The qterm command provides options to manage the behavior of the secondary server; it can be shut down, forced to remain idle, or shut down in place of the primary server.

In order to run the qterm command, the user must have PBS Operator or Manager privilege.

2.60.3   Options

The following table lists the options to the qterm command.

Table 2-29: qterm Options

(no option)	The qterm command defaults to –t quick.
-f	If the complex is configured for failover, both the primary and secondary servers are shut down. Without the -f option, the primary server is shut down and the secondary server becomes active. The -f option cannot be used with the -i or -F options.
-F	If the complex is configured for failover, only the secondary server is shut down, and the primary server remains active. The -F option cannot be used with the -f or -i options.

Table 2-29: `qterm` Options

-i	If the complex is configured for failover, the secondary server remains idle when the primary server is shut down. The -i option cannot be used with the -f or -F options.	
-m	All MOMs (`pbs_mom`) are shut down. This option does not cause jobs or subjobs to be killed. Jobs are left running subject to other options to the `qterm` command.	
-s	The scheduler (`pbs_sched`) is shut down.	
-t <type>	immediate	All running jobs immediately stop execution. Any running jobs that can be checkpointed are checkpointed, terminated, and requeued. Jobs that cannot be checkpointed are terminated and requeued if they are rerunnable, otherwise they are killed. If any job cannot be terminated, for example the server cannot contact the MOM of a running job, the server continues to execute and the job is listed as running. The server can be terminated by a second `qterm -t immediate` command. While terminating, the server is in the *Terminating* state.
	delay	The server waits to terminate until all non-checkpointable, non-rerunnable jobs are finished executing. Any running jobs that can be checkpointed are checkpointed, terminated, and requeued. Jobs that cannot be checkpointed are terminated and requeued if they are rerunnable, otherwise they are allowed to continue to run. While terminating, the server is in the *Terminating-Delayed* state.
	quick	Running jobs are left running. Running subjobs are requeued. This is the default behavior when no options are given to the `qterm` command. While terminating, the server is in the *Terminating* state.
--version	The `qterm` command returns its PBS version information and exits. This option can only be used alone.	

2.60.4 Operands

The server list operand specifies which servers are to shut down. It is a space-separated list of server names. If no servers are specified, then the default server is shut down.

2.60.4.1 Standard Error

The `qterm` command writes a diagnostic message to standard error for each error occurrence.

2.60.4.2 Exit Status

Zero upon successful processing of all the operands presented to the `qterm` command.

Greater than zero if the `qterm` command fails to process any operand.

2.60.4.3 See Also

The PBS Professional Administrator's Guide, `pbs_server(8B)`, `pbs_mom(8B)`, `pbs_sched(8B)`

2.61 tracejob

Prints log messages for a PBS job

2.61.1 Synopsis

tracejob [-a] [-c count] [-f filter] [-l] [-m] [-n days] [-p path] [-s] [-v] [-w cols] [-z] jobid

tracejob --version

2.61.2 Description

The `tracejob` command extracts log messages for a given *jobid* and prints them in chronological order.

Log messages contain server, scheduler, accounting and MOM information. Server logs contain information such as when a job was queued or modified. Scheduler logs contain clues as to why a job is not running. Accounting logs contain accounting records for when a job was queued, started, ended or deleted. MOM logs contain information about what happened to a job while it was running.

To get MOM log messages for a job, `tracejob` must be run on the machine on which the job ran.

All users have access to server, scheduler and MOM information. Only Administrator or root can access accounting information.

Some log messages appear many times. In order to make the output of `tracejob` more readable, messages that appear over a certain number of times (see option -c below) are restricted to only the most recent message.

If `tracejob` is run on a job array, the information returned will be about the job array itself, and not its subjobs. Job arrays do not have associated MOM log messages. If `tracejob` is run on a subjob, the same types of log messages will be available as for a job. Certain log messages that occur for a regular job will not occur for a subjob.

Note that some shells require that you enclose a job array identifier in double quotes.

2.61.3   Options to `tracejob`

-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.

    The default for count is 15.

-f <filter>
    Do not include log events of type filter. The -f option can be used more than once on the command line.

    Filter values:

    Filter values can be any of: *error, system, admin, job, job_usage, security, sched, debug*, *debug2*, *resv*, *debug3*, or *debug4*, or their equiva-

lent in hexadecimal. The following table shows the hex value and category for each filter.

Table 2-30: Filters

Filter	Hex Value	Message Category
error	*0x0001*	Internal errors
system	*0x0002*	System errors
admin	*0x0004*	Administrative events
job	*0x0008*	Job-related events
job_usage	*0x0010*	Job accounting info
security	*0x0020*	Security violations
sched	*0x0040*	Scheduler events
debug	*0x0080*	Common debug messages
debug2	*0x0100*	Uncommon debug messages
resv	*0x0200*	Reservation debug messages
debug3	*0x0400*	Less common than debug2
debug4	*0x0800*	Less common than debug3

-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

        Suppresses printing of duplicate messages.

--version

        The `tracejob` command returns its PBS version information and exits.
        This option can only be used alone.

2.61.4   Exit Status

Zero upon successful processing of all options.

Exit value is greater than zero if `tracejob` is unable to process any options.

2.61.5   See Also

The PBS Professional Administrator's Guide

`pbs_server(8B), pbs_sched(8B), pbs_mom(8B)`

2.62   xpbs

GUI front end to PBS commands

2.62.1   Synopsis

xpbs [-admin]
xpbs --version

2.62.2   Description

The `xpbs` command provides a user-friendly point-and-click interface to PBS commands.
Please see the sections below for a tour and tutorials. Also, within every dialog box, a Help
button can be found for assistance.

2.62.3   Options

-admin

A mode where additional buttons are made available for terminating PBS
servers, starting/stopping/disabling/enabling queues, and running/rerunning
jobs.

--version

The xpbs command returns its PBS version information and exits. This
option can only be used alone.

2.62.4   Getting Started

Running xpbs will initialize the X resource database from various sources in the following
order:

1.   The RESOURCE_MANAGER property on the root window (updated via xrdb) with
settings usually defined in the .Xdefaults file

2.   Preference settings defined by the system administrator in the global xpbsrc file

3.   User's ~/.xpbsrc file - this file defines various X resources like fonts, colors, list
of PBS hosts to query, criteria for listing queues and jobs, and various view states. See
section 2.62.13, "Setting Preferences", on page 245 below for a list of resources that can
be set.

2.62.5   Running **xpbs**

To run xpbs as a regular, non-privileged user, type:

```
setenv DISPLAY <display_host>:0
xpbs
```

To run xpbs with the additional purpose of terminating PBS servers, stopping and starting
queues, or running/rerunning jobs, then run:

```
xpbs –admin
```

NOTE: Be sure to appropriately set ~/.rhosts file if you're planning to submit jobs to
some remote server, and expecting output files to be returned to the local host (where xpbs
was run). Usually, adding the PBS hostname running the server to your .rhosts file locally,
and adding the name of the local machine to the .rhosts file at remote host, should be suffi-
cient.

Also, be sure that the PBS client commands are in the default PATH because xpbs will call
these commands.

2.62.6   The **xpbs** Display

This section describes the main parts of the **xpbs** display. The main window is composed of 5 distinct areas (subwindows) arranged vertically (one on top of another) in the following order:

1. Menu

2. Hosts

3. Queues

4. Jobs

5. Info

2.62.6.1   Menu

The Menu area is composed of a row of command buttons that signal some action with a click of the left mouse button. The buttons are:

Manual Update
> to update the information on hosts, queues, and jobs.

Auto Update
> same as Manual Update except updating is done automatically every <some specified> number of minutes.

Track Job
> for periodically checking for returned output files of jobs.

Preferences
> for setting certain parameters such as the list of server host(s) to query.

Help
> contains some help information.

About
> tells of the author and who to send comments, bugs, suggestions to.

Close
> for exiting **xpbs** plus saving the current setup information (if anything had changed) in the user's `$HOME/.xpbsrc` file. Information saved include the selected host(s), queue(s), job(s), the different jobs listing criteria, the view states (i.e. minimized/maximized) of the Hosts, Queues, Jobs, and INFO regions, and anything in the Preferences section.

2.62.6.2 Hosts

The Hosts area is composed of a leading horizontal HOSTS bar, a listbox, and a set of command buttons. The HOSTS bar contains a minimize/maximize button, identified by a dot or a rectangular image, for displaying or iconifying the Hosts region. The listbox displays information about favorite server host(s), and each entry is meant to be selected via a single left mouse button click, shift key + mouse button 1 click for contiguous selection, or cntrl key + mouse button 1 click for non-contiguous selection. The command buttons represent actions on selected host(s), and commonly found buttons are:

detail

> for obtaining detailed information about selected server host(s). This functionality can also be achieved by double clicking on an entry in the Hosts listbox.

Submit

> for submitting a job to any of the queues managed by the selected host(s).

terminate

> for terminating PBS servers on selected host(s). (-admin only)

The server hosts can be chosen by specifying in the ~/.xpbsrc file (or .Xdefaults) the resource:

```
*serverHosts: hostname1 hostname2 ...
```

Another way of specifying the host is to click on the Preferences button in the Menu region, and manipulate the server Hosts entry widget from the preferences dialog box.

2.62.6.3 Queues

The Queues area is composed of a leading horizontal QUEUES bar, a listbox, and a set of command buttons. The QUEUES bar lists the hosts that are consulted when listing queues; the bar also contains a minimize/maximize button for displaying or iconifying the Queues region. The listbox displays information about queues managed by the server host(s) selected from the Hosts listbox; each listbox entry is meant to be selected (highlighted) via a single left mouse button click, shift key + mouse button 1 click for contiguous selection, or cntrl key + mouse button 1 click for non-contiguous selection. The command buttons represent actions for operating on selected queue(s), and commonly found buttons are:

detail

> for obtaining detailed information about selected queue(s). This functionality can also be achieved by double clicking on a Queues listbox entry.

stop

> for stopping the selected queue(s). (-admin only)

start

> for starting the selected queue(s). (-admin only)

disable

> for disabling the selected queue(s). (-admin only)

enable

> for enabling the selected queue(s). (-admin only)

2.62.6.4 Jobs

The Jobs area is composed of a leading horizontal JOBS bar, a listbox, and a set of command buttons. The JOBS bar lists the queues that are consulted when listing jobs; the bar also contains a minimize/maximize button for displaying or iconifying the Jobs region. The listbox displays information about jobs that are found in the queue(s) selected from the Queues listbox; each listbox entry is meant to be selected (highlighted) via a single left mouse button click, shift key + mouse button 1 click for contiguous selection, or cntrl key + mouse button 1 click for non-contiguous selection. The region just above the Jobs listbox shows a collection of command buttons whose labels describe criteria used for filtering the Jobs listbox contents. The list of jobs can be selected according to the owner of jobs (Job_Owner), job state (Job_State), name of the job (Job_Name), type of hold placed on the job (Hold_Types), the account name associated with the job (Account_Name), checkpoint attribute (Checkpoint), time the job is eligible for queueing/execution (Queue_Time), resources requested by the job (Resource_List), priority attached to the job (Priority), and whether or not the job is rerunnable (Rerunnable). The selection criteria can be modified by clicking on any of the appropriate command buttons to bring up a selection box. The criteria command buttons are accompanied by a Select Jobs button, which when clicked, will update the contents of the Jobs listbox based on the new selection criteria. Please see `qselect(1B)` for more details on how the jobs are filtered.

Finally, to the right of the listbox, the Jobs region is accompanied by the following command buttons, for operating on selected job(s):

detail

> for obtaining detailed information about selected job(s). This functionality can also be achieved by double clicking on a Jobs listbox entry.

modify

> for modifying attributes of the selected job(s).

delete

> for deleting the selected job(s).

hold

> for placing some type of hold on selected job(s).

release

for releasing held job(s).

signal

for sending signals to selected job(s) that are running.

msg

for writing a message string into the output streams of the selected job(s).

move

for moving selected job(s) into some specified destination queue.

order

for exchanging order of two selected jobs in a queue.

run

for running selected job(s). (-admin only)

rerun

for requeueing selected job(s) that are running. (-admin only)

2.62.6.5   Info

The Info Area shows the progress of the commands' executed by `xpbs`. Look into this box for errors. The INFO bar also contains a minimize/maximize button for displaying or iconifying the Info region.

2.62.7   **Widgets Used in `xpbs`**

Some of the widgets used in `xpbs` and how they are manipulated are described in the following:

2.62.7.1   listbox

can be multi-selectable (a number of entries can be selected/highlighted using a mouse click) or single-selectable (one entry can be highlighted at a time). For a multi-selectable listbox, the following operations are allowed:

  1.   single click with mouse button 1 to select/highlight an entry.

  2.   shift key + mouse button 1 to contiguously select more than one entry.

  3.   cntrl key + mouse button 1 to non-contiguously select more than one entry. NOTE: For systems running Tk < 4.0, the newly selected item is reshuffled to appear next to already selected items.

  4.   click the Select All/Deselect All button to select all entries or deselect all entries at once.

  5.   double clicking an entry usually activates some action that uses the selected entry as a parameter.

2.62.7.2   scrollbar

usually appears either vertically or horizontally and contains 5 distinct areas that are mouse clicked to achieve different effects:

top arrow

Causes the view in the associated widget to shift up by one unit (i.e. the object appears to move down one unit in its window). If the button is held down the action will auto-repeat.

top gap

Causes the view in the associated window to shift up by one less than the number of units in the window (i.e. the portion of the object that used to appear at the very top of the window will now appear at the very bottom). If the button is held down the action will auto-repeat.

slider

Pressing button 1 in this area has no immediate effect except to cause the slider to appear sunken rather than raised. However, if the mouse is moved with the button down then the slider will be dragged, adjusting the view as the mouse is moved.

bottom gap

Causes the view in the associated window to shift down by one less than the number of units in the window (i.e. the portion of the object that used to appear at the very bottom of the window will now appear at the very top). If the button is held down the action will auto-repeat.

bottom arrow

> Causes the view in the associated window to shift down by one unit (i.e. the object appears to move up one unit in its window). If the button is held down the action will auto-repeat.

2.62.7.3   entry

brought into focus with a click of the left mouse button. To manipulate this widget, simply type in the text value. Use of arrow keys, mouse selection of text for deletion or overwrite, copying and pasting with sole use of mouse buttons are permitted. This widget is usually accompanied by a scrollbar for horizontally scanning a long text entry string.

2.62.7.4   matrix of entry boxes

usually shown as several rows of entry widgets where a number of entries (called fields) can be found per row. The matrix is accompanied by up/down arrow buttons for paging through the rows of data, and each group of fields gets one scrollbar for horizontally scanning long entry strings. Moving from field to field can be done using the <Tab>, <Cntrl-f>, or <Cntrl-b> (move backwards) keys.

2.62.7.5   spinbox

a combination of an entry widget and a horizontal scrollbar. The entry widget will only accept values that fall within a defined list of valid values, and incrementing through the valid values is done by clicking on the up/down arrows.

2.62.7.6   button

a rectangular region appearing either raised or pressed that invokes an action when clicked with the left mouse button. When the button appears pressed, then hitting the <RETURN> key will automatically select the button.

2.62.7.7   text

an editor like widget. This widget is brought into focus with a click of the left mouse button. To manipulate this widget, simply type in the text. Use of arrow keys, backspace/delete key, mouse selection of text for deletion or overwrite, copying and pasting with sole use of mouse buttons are permitted. This widget is usually accompanied by a scrollbar for vertically scanning a long entry.

2.62.8   Submitting Jobs

Submitting a PBS job requires only to manipulate the widgets found in the Submit window. The submit dialog box is composed of 4 distinct regions:

1.   Job Script

2.   Options

3.   OTHER Options

4.   Command Buttons

The Job Script file region is at the upper left, the Options region containing various widgets for setting job attributes is scattered all over the dialog box, the OTHER Options is located just below the Job Script file region, and Command Buttons region is at the bottom.

The job script region is composed of a header box, the text box, FILE entry box, and a couple of buttons labeled load and save. If you have a script file containing PBS options and executable lines, then type the name of the file on the FILE entry box, and then click on the load button. The various widgets in the Submit window will get loaded with values found in the script file. The script file text box will only be loaded with executable lines (non-PBS) found in the script. The job script header box has a Prefix entry box that can be modified to specify the PBS directive to look for when parsing a script file for PBS options. If you don't have a script file, you can start typing the executable lines of the job in the file text box.

To submit a job, perform the following steps:

1.   Select a host from the HOSTS listbox in the main `xpbs` display.

2.   Click on the Submit button located in the Menu bar.

3.   Specify the script file containing the job execution lines and job resource and attribute values, or simply type in the execution lines in the FILE textbox.

4.   Start manipulating the various widgets in the Submit window. Particularly, pay close attention to the Destination listbox. This box lists all the queues found in the host that you selected. A special entry called "@host" refers to the default queue at host. Select appropriately the destination queue of the job. More options can be found by clicking the OTHER Options buttons.

5.   At the bottom of the Submit window, click confirm submit . You can also click on interactive to run the job interactively. Running a job interactively will open an xterm window to your display host containing the session.

   NOTE: The script FILE entry box is accompanied by a save button that you click to save the current widget values to the specified file in a form that can later be read by `xpbs` or by the `qsub` command.

2.62.9   Modifying Attributes of Jobs

Modifying a PBS job requires only to manipulate the widgets found in the Modify window. To modify a job or jobs, do the following steps:

1.   Select one or more jobs from the JOBS listbox in the main `xpbs` display.

2.   Click on the modify button located to the right of the listbox.

3.   The Modify window is structured similarly to the Submit window. Simply manipulate the widgets to specify replacement or additional values of job attributes.

4.   Click on the confirm modify button located at the bottom of the dialog box.

2.62.10   Deleting Jobs

Deleting a PBS job requires only to manipulate the widgets found in the Delete window. To delete a job or jobs, do the following steps:

1.   Select one or more jobs from the JOBS listbox in the main `xpbs` display.

2.   Click on the delete button located to the right of the listbox.

3.   Manipulate the spinbox widget to set the kill delay signal interval.

4.   Click on the delete button located at the bottom of the dialog box.

2.62.11   Tracking Returned Output Files

If you want to be informed of returned output files of current jobs, and be able to quickly see the contents of those files, then enable the "track job" feature as follows:

1.   Submit all the jobs that you want monitored.

2.   Click on the Track Job button located in the Menu bar to bring up the Track Job dialog box.

3.   Specify the list of user names, whose jobs are to be monitored for returned output files, in the matrix located at the upper left of the dialog box.

4.   Manipulate the minutes spinbox, located just below the user names matrix, to specify the interval value when output files will be periodically checked.

5.   Specify the location of job output files (whether locally or remotely) by clicking on one of the radio buttons located at the upper right of the dialog box. Returned locally means the output files will be returned back to the host where `xpbs` was run. If the output files are returned to some remote host, then `xpbs` will execute an RSH

<remote_host> test -f <output_files> to test the existence of the files. RSH is whatever you set the remote shell command to in the corresponding entry box.

NOTE: Be sure the files are accessible from the host where `xpbs` was run (i.e. `.rhosts` appropriately set).

6.   Click start/reset tracking button located at the bottom of the dialog box to:

   -    cancel any previous tracking

   -    build a new list of jobs to be monitored for returned output files based on currently queued jobs.

   -    start periodic tracking.

7.   Click on close window button.

When an output file for a job being monitored is found, then the Track Job button (the one that originally invoked the Track Job dialog box) will turn into a different color, and the Jobs Found Completed listbox, located in the Track Job dialog box, is then loaded with the corresponding job id(s). Then double click on a job id to see the contents of the output file and the error file. Click stop tracking if you want to cancel tracking.

2.62.12   Leaving **xpbs**

Click on the Close button located in the Menu bar to leave `xpbs`. If anything had changed, it will bring up a dialog box asking for a confirmation in regards to saving state information like the view states (minimize/maximize) of the HOSTS, QUEUES, JOBS, and INFO subwindows, and various criteria for listing queues and jobs. The information is saved in `~/.xpbsrc` file.

2.62.13   Setting Preferences

The resources that can be set in the X resources file, `~/.xpbsrc`, are:

   *serverHosts
                list of server hosts (space separated) to query by `xpbs` keyword
                PBS_DEFAULT_SERVER can be used which will be used as a place
                holder for the value obtained from *defServerFile.

   *defServerFile
                the file containing the name of the default server host. The content of this
                will be substituted for the *PBS_DEFAULT_SERVER* keyword in *server-
                Hosts value.

*timeoutSecs

> specify the number of seconds before timing out waiting for a connection to a PBS host.

*xtermCmd

> the xterm command to run driving an interactive PBS session.

*labelFont

> font applied to text appearing in labels.

*fixlabelFont

> font applied to text that label fixed-width widgets such as listbox labels. This must be a fixed-width font.

*textFont

> font applied to a text widget. Keep this as fixed-width font.

*backgroundColor

> the color applied to background of frames, buttons, entries, scrollbar handles.

*foregroundColor

> the color applied to text in any context (under selection, insertion, etc...).

*activeColor

> the color applied to the background of a selection, a selected command button, or a selected scroll bar handle.

*disabledColor

> color applied to a disabled widget.

*signalColor

> color applied to buttons that signal something to the user about a change of state. For example, the color of the Track Job but-
>
> ton when returned output files are detected.

*shadingColor

> a color shading applied to some of the frames to emphasize focus
>
> as well as decoration.

*selectorColor

> the color applied to the selector box of a radio button or check-
>
> button.

*selectHosts

> list of hosts (space separated) to automatically select/high-
>
> light in the HOSTS listbox.

*selectQueues

        list of queues (space separated) to automatically select/high-

        light in the QUEUES listbox.

*selectJobs

        list of jobs (space separated) to automatically select/highlight

        in the JOBS listbox.

*selectOwners

        list of owners checked when limiting the jobs appearing on the Jobs listbox
        in the main xpbs window. Specify value as "*Owners: <list_of_owners>*".
        See -u option in qselect(1B) for format of *<list_of_owners>*.

*selectStates

        list of job states to look for (do not space separate) when limiting the jobs
        appearing on the Jobs listbox in the main xpbs window. Specify value as
        "*Job_States: <states_string>*". See -s option in qselect(1B) for for-
        mat of *<states_string>*.

*selectRes

        list of resource amounts (space separated) to consult when limiting the jobs
        appearing on the Jobs listbox in the main xpbs window. Specify value as
        "*Resources: <res_string>*". See -l option in qselect(1B) for format of
        <res_string>.

*selectExecTime

        the Execution Time attribute to consult when limiting the list of jobs
        appearing on the Jobs listbox in the main xpbs window. Specify value as
        "*Queue_Time: <exec_time>*". See -a option in qselect(1B) for format
        of *<exec_time>*.

*selectAcctName

        the name of the account that will be checked when limiting the jobs appear-
        ing on the Jobs listbox in the main xpbs window. Specify value as
        "*Account_Name: <account_name>*". See -A option in qselect(1B) for
        format of *<account_name>*.

*selectCheckpoint

        the checkpoint attribute relationship (including the logical operator) to con-
        sult when limiting the list of jobs appearing on the Jobs listbox in the main
        xpbs window. Specify value as "*Checkpoint: <checkpoint_arg>*". See -c
        option in qselect(1B) for format of *<checkpoint_arg>*.

*selectHold

        the hold types string to look for in a job when limiting the jobs appearing on
        the Jobs listbox in the main xpbs window. Specify value as "*Hold_Types:*

<hold_string>". See -h option in qselect(1B) for format of *<hold_string>*.

*selectPriority

the priority relationship (including the logical operator) to consult when limiting the list of jobs appearing on the Jobs listbox in the main xpbs window. Specify value as "*Priority: <priority_value>*". See -p option in qselect(1B) for format of *<priority_value>*.

*selectRerun

the rerunnable attribute to consult when limiting the list of jobs appearing on the Jobs listbox in the main xpbs window. Specify value as "*Rerunnable: <rerun_val>*". See -r option in qselect(1B) for format of *<rerun_val>*.

*selectJobName

name of the job that will be checked when limiting the jobs appearing on the Jobs listbox in the main xpbs window. Specify value as "*Job_Name: <jobname>*". See -N option in qselect(1B) for format of *<jobname>*.

*iconizeHostsView

a boolean value (*True* or *False*) indicating whether or not to iconize the HOSTS region.

*iconizeQueuesView

a boolean value (*True* or *False*) indicating whether or not to iconize the QUEUES region.

*iconizeJobsView

a boolean value (*True* or *False*) indicating whether or not to iconize the JOBS region.

*iconizeInfoView

a boolean value (*True* or *False*) indicating whether or not to iconize the INFO region.

*jobResourceList

a curly-braced list of resource names as according to architecture known to xpbs. The format is as follows:

{ <arch-type1> resname1 resname2 ... resnameN }

{ <arch-type2> resname1 resname2 ... resnameN }

. . .

{ <arch-typeN> resname1 resname2 ... resnameN }

2.62.14   **xpbs** and PBS Commands

xpbs calls PBS commands as follows:

Table 2-31: xpbs and PBS Commands

Command Button	PBS Command
detail (Hosts)	qstat –B –f <selected server_host(s)>
terminate	qterm <selected server_host(s)>
detail (Queues)	qstat –Q –f <selected queue(s)>
stop	qstop <selected queue(s)>
start	qstart <selected queue(s)>
enable	qenable <selected queue(s)>
disable	qdisable <selected queue(s)>
detail (Jobs)	qstat –f <selected job(s)>
modify	qalter <selected job(s)>
delete	qdel <selected job(s)>
hold	qhold <selected job(s)>
release	qrls <selected job(s)>
run	qrun <selected job(s)>
rerun	qrerun <selected job(s)>
signal	qsig <selected job(s)>
msg	qmsg <selected job(s)>
move	qmove <selected job(s)>
order	qorder <selected job(s)>

2.62.15   Exit Status

Upon successful processing, the xpbs exit status will be a value of zero.

If the xpbs command fails, the command exits with a value greater than zero.

2.62.16   See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide, qalter(1B), qdel(1B), qhold(1B), qmove(1B), qmsg(1B), qrerun(1B), qrls(1B), qselect(1B), qsig(1B), qstat(1B), qorder(1B), qsub(1B), qdisable(8B), qenable(8B), qrun(8B), qstart(8B), qstop(8B), qterm(8B)

2.63   xpbsmon

GUI for displaying, monitoring execution hosts under PBS

2.63.1   Synopsis

xpbsmon

xpbsmon --version

2.63.2   Description

The xpbsmon command provides a way to graphically display the various nodes that run jobs. A node or execution host can be running a pbs_mom daemon, or not running the daemon. For the latter case, it could just be a nodename that appears in a nodes file that is managed by a main pbs_server running on another host. This utility also provides the ability to monitor values of certain system resources by posting queries to the pbs_mom of a node. With this utility, you can see what job is running on what node, who owns the job, how many nodes assigned to a job, status of each node (color-coded and the colors are user-modifiable), how many nodes are available, free, down, reserved, offline, of unknown status, in use running multiple jobs or executing only 1 job. Please see the sections below for a tour and tutorials of xpbsmon. Also, within every dialog box, a Help button can be found for assistance.

2.63.3 Getting Started

Running `xpbsmon` will initialize the X resource database from various sources in the following order:

1.   The RESOURCE_MANAGER property on the root window (updated via xrdb) with settings usually defined in the .Xdefaults file

2.   Preference settings defined by the system administrator in the global `xpbsmonrc` file

3.   User's `~/.xpbsmonrc` file - this file defines various X resources like fonts, colors, list of colors to use to represent the various status of the nodes, list of PBS sites to query, list of server hosts on each site, list of nodes/execution hosts on each server host, list of system resource queries to send to the nodes' pbs_mom, and various view states. See below for a list of resources that can be set.

2.63.4 Running `xpbsmon`

`xpbsmon` can be run either as a regular user or superuser. If you run it with less privilege, you may not be able to see all the information for a node. If it is executed as a regular user, you should still be able to see what jobs are running on what nodes, possibly state, as this information are obtained by `xpbsmon` talking directly to the specified server. If you want other system resource values, it may require special privilege since `xpbsmon` will have to talk directly to the `pbs_mom` of a node. In addition, the host where `xpbsmon` was running must also have been given explicit access permission by the MOM (unless the GUI is running on the same host where MOM is running). This is done done by updating the $clienthost and/or the $restricted parameter on the MOM's configuration file.

To run `xpbsmon`, type:

```
setenv DISPLAY <display_host>:0
xpbsmon
```

If you are running the GUI and only interested in jobs data, then be sure to set all the nodes' type to NOMOM in the Pref dialog box.

2.63.5 Options

--version
                The `xpbsmon` command returns its PBS version information and exits.
                This option can only be used alone.

2.63.6   The `xpbsmon` Display

This section describes the main parts of the `xpbsmon` display. The main window is composed of 3 distinct areas (subwindows) arranged vertically (one on top of another) in the following order:

1.   Menu

2.   Site Information

3.   Info

2.63.6.1   Menu

The Menu area is composed of a row of command buttons that signal some action with a click of the left mouse button. The buttons are:

Site..
> displays a popup menu containing the list of PBS sites that have been added using the Sites Preferences window. Simply drag your mouse and release to the site name whose servers/nodes information you would like to see.

Pref..
> brings up various dialog boxes for specifying the list of sites, servers on each site, nodes that are known to a server, and the system resource queries to be sent to a node's `pbs_mom` daemon.

Auto Update..
> brings up another window for specifying whether or not to do auto updates of nodes information, and also for specifying the interval number of minutes between updates.

Help
> contains some help information.

About
> tells who the author is and who to send comments, bugs, suggestions to.

Close
> for exiting `xpbsmon` plus saving the current setup information (if anything had changed) in the user's `$HOME/.xpbsmonrc` file. Information saved include the specified list of sites, servers on each site, nodes known to each server, and system resource queries to send to node's `pbs_mom`.

Minimize Button
> shows the iconized view of Site Information where nodes are represented as tiny boxes, where each box is colored according to status. In order to get

more information about a node, you need to double click on the colored box.

Maximize button

shows the full view of Site Information where nodes are represented in bigger boxes, still colored depending on the status, and some information on it is displayed.

2.63.6.2   Site Information

Only one site at a time can be displayed. This area (shown as one huge box referred to as the site box) can be further subdivided into 3 areas: the site name label at the top, server boxes in the middle, and the color status bar at the bottom. The site name label shows the name of the site as specified in the Pref.. window. At the middle of the site box shows a row of big boxes housing smaller boxes.

The big box is an abstraction of a server host (called a server box), showing its server display label at the top of the box, a grid of smaller boxes representing the nodes that the server knows about (where jobs are run), and summary status for the nodes under the server. Status information will show counters for the number of nodes used, available, reserved, offline, or of unknown status and even # of cpus assigned. For a cleaner display, some counters with a value of zero are not displayed. The server boxes are placed in a grid, with a new row being started when either *siteBoxMaxNumServerBoxesPerRow or *siteBoxMaxWidth limit has been reached.

The smaller boxes represent the nodes/execution hosts where jobs are run (referred to as node boxes). Each node box shows the name at the top, and a sub-box (a smaller square) that is colored according to the status of the node that it represents, and if the view type is FULL, it will display some node information according to the system resource queries specified on the Pref.. window. Clicking on the sub-box will show a much bigger box (called the MIRROR view) with bigger fonts containing nodes information. Another view is called ICON and this shows a tiny box with a colored area. The node boxes are arranged in a grid, where a new row is created if either the *serverBoxMaxNumNodeBoxesPerRow or *serverBoxMaxWidth limit has been reached. ICON view of the node boxes will be constrained by the *nodeBoxIcon-MaxHeight and *nodeBoxIconMaxWidth pixel values; FULL view of the node boxes will be bounded by *nodeBoxFullMaxWidth and *nodeBoxFullMaxHeight; the mirror view of the node boxes has its size be *nodeBoxMirrorMaxWidth, and *nodeBoxMirrorMaxHeight.

Horizontal and vertical scrollbars for the site box, server box, and node box will be displayed as needed.

Finally, the color bar information shows a color chart displaying what the various colors mean in terms of node status. The color-to-status mapping can be modified by setting the X resources: *nodeColorNOINFO, *nodeColorFREE, *nodeColorINUSEshared, *nodeColorI-NUSEexclusive, *nodeColorDOWN, *nodeColorRSVD, *nodeColorOFFL, *nodeColor-BUSY.

2.63.6.3   Info

The Info Area shows the progress of some of the background actions performed by `xpb-smon`. Look into this box for errors.

2.63.7   Widgets Used in `xpbsmon`

Some of the widgets used in `xpbsmon` and how they are manipulated are described in the following:

2.63.7.1   listbox

the ones found in this GUI are only single-selectable (one entry can be highlighted/selected at a time via a mouse click).

2.63.7.2   scrollbar

usually appears either vertically or horizontally and contains 5 distinct areas that are mouse clicked to achieve different effects:

top arrow

Causes the view in the associated widget to shift up by one unit (i.e. the object appears to move down one unit in its window). If the button is held down the action will auto-repeat.

top gap

Causes the view in the associated window to shift up by one less than the number of units in the window (i.e. the portion of the object that used to appear at the very top of the window will now appear at the very bottom). If the button is held down the action will auto-repeat.

slider

Pressing button 1 in this area has no immediate effect except to cause the slider to appear sunken rather than raised. However, if the mouse is moved with the button down then the slider will be dragged, adjusting the view as the mouse is moved.

bottom gap

> Causes the view in the associated window to shift down by one less than the number of units in the window (i.e. the portion of the object that used to appear at the very bottom of the window will now appear at the very top). If the button is held down the action will auto-repeat.

bottom arrow

> Causes the view in the associated window to shift down by one unit (i.e. the object appears to move up one unit in its window). If the button is held down the action will auto-repeat.

2.63.7.3   entry

brought into focus with a click of the left mouse button. To manipulate this widget, simply type in the text value. Use of arrow keys, mouse selection of text for deletion or overwrite, copying and pasting with sole use of mouse buttons are permitted. This widget is usually accompanied by a scrollbar for horizontally scanning a long text entry string.

2.63.7.4   box

made up of 1 or more listboxes displayed adjacent to each other giving the effect of a "matrix". Each row from the listboxes makes up an element of the box. In order to add items to the box, you need to manipulate the accompanying entry widgets, one for each listbox, and then clicking the add button. Removing items from the box is done by selecting an element, and then clicking delete.

2.63.7.5   spinbox

a combination of an entry widget and a horizontal scrollbar. The entry widget will only accept values that fall within a defined list of valid values, and incrementing through the valid values is done by clicking on the up/down arrows.

2.63.7.6   button

a rectangular region appearing either raised or pressed that invokes an action when clicked with the left mouse button. When the button appears pressed, then hitting the <RETURN> key will automatically select the button.

2.63.8   Updating Preferences

2.63.8.1   Time Sharing

Suppose you have a time-sharing environment where the front-end is called bower and you have 4 nodes: bower1, bower2, bower3, bower4. bower is the host that runs the server; jobs are submitted to host bower where it enqueues it for future execution. Also, a `pbs_mom` daemon is running on each of the execution hosts. If the server bower also maintains a nodes list containing information like state for the 4 nodes, then this will also be reported. Then to setup `xpbsmon`, do the following:

1.   Click the Pref.. button on the Menu section.

2.   On the Sites Preference dialog, enter any arbitrary site name, for example "Local". Then click the add button.

3.   On the Server_Host entry box, enter "bower", and on the DisplayLabel entry box, put an arbitrary label (as it would appear on the header of the server box) like "Bower", and then click add.

4.   Click the nodes.. button that is accompanying the Servers box. This would bring up the Server Preference dialog.

5.   Now add the entries "bower1", "bower2", "bower3", "bower4" specifying type MOM for each on the Nodes box.

6.   If you need to monitor certain system resource parameters for each of the nodes, you need to specify query expressions containing resource queries to be sent to the individual PBS moms. For example, if you want to obtain memory usage, then select a node from the Nodes list, click on the query.. button that accompanies the Nodes list, and this would bring up the Query Table dialog. Specify the following input:

```
Query_Expr: (availmem/totmem) * 100
Display_Info: Memory Usage:
Display_Type: SCALE
```

The above says to display the result of the "Query_Expr" in a scale widget calibrated over 100. The queries "availmem" and "totmem" will be sent to the PBS mom, and the

expression is evaluated upon receiving all results from the mom. If you want to display the result of another query, say "loadave", directly, then specify the following:

`Query_Expr: loadave`

`Display_Info: Load Average:`

`Display_Type: TEXT`

NOTE: For a list of queries that can be sent to a `pbs_mom`, please click on the Help button on the Query table window.

2.63.8.2 Jobs Exclusive Environment

Supposing you have a "space non-sharing" environment where the server maintains a list of nodes that it runs jobs on exclusively (one job at a time outstanding per node). Let's call this server b1. Simply update Preferences information as follows:

1.   Click the Pref.. button on the Menu section.

2.   On the Sites Preference dialog, enter a site name, for example "B System". Then click the add button.

3.   On the Server_Host entry box, enter "b1", DisplayLabel entry box type "B1" (or whatever label that you would like to appear on the header of the server box), and then click add.

2.63.8.3 Hybrid Time Sharing/Space Sharing Environment

A cluster of heterogeneous machines, time-sharing or jobs exclusive, could easily be represented in `xpbsmon` by combining steps in CASE 1 and CASE 2.

2.63.9 Leaving `xpbsmon`

Click on the Close button located in the Menu bar to leave `xpbsmon`. If anything had changed, it will bring up a dialog box asking for a confirmation in regards to saving preferences information about list of sites, their view types, list of servers on each site, the list of nodes known to each server, and the list of queries to be sent to the pbs_mom of each node. The information is saved in `~/.xpbsmonrc` file.

2.63.10 Setting Preferences

The resources that can be set in the X resources file, `~/.xpbsmonrc`, are described in the following:

2.63.10.1   Node Box Properties

Resource names beginning with "*small" or "*node" apply to the properties of the node boxes. A node box is made of an outer frame where the node label sits on top, the canvas (smaller box) is on the middle, and possibly some horizontal/ vertical scrollbars.

nodeColorNOINFO
> color of node box when information for the node it represents could not be obtained.

*nodeColorFREE
> color of canvas when node it represents is up.

*nodeColorINUSEshared
> color when node it represents has more than 1 job running on it, or when node has been marked by the server that manages it as "job-sharing".

*nodeColorINUSEexclusive
> list of colors to assign to a node box when host it represents is running only 1 job, or when node has been marked by the server that manages it as "time-sharing". xpbsmon will use this list to assign 1 distinct color per job unless all the colors have been exhausted, in which case, colors will start getting assigned more than once in a round-robin fashion.

*nodeColorDOWN
> color when node it represents is down.

*nodeColorRSVD
> color when node it represents is reserved.

*nodeColorOFFL
> color when node it represents is offline.

*nodeColorBUSY
> color when node it represents is busy (high load average).

*smallForeground
> applies to the color of text inside the canvas.

*smallBackground
> applies to the color of the frame.

*smallBorderWidth
> distance (in pixels) from other node boxes.

*smallRelief
> how node box will visually appear (style).

*smallScrollBorderWidth
>
> significant only in FULL mode, this is the distance of the horizontal/vertical scrollbars from the canvas and lower edge of the frame.

*smallScrollBackground
>
> background color of the scrollbars

*smallScrollRelief
>
> how scrollbars would visually appear (style).

*smallCanvasBackground
>
> color of the canvas (later overridden depending on status of the node it represents)

*smallCanvasBorderWidth
>
> distance of the canvas from the frame and possibly the scrollbars.

*smallCanvasRelief
>
> how the canvas is visually represented (style).

*smallLabelBorderWidth
>
> the distance of the node label from the canvas and the topmost edge of the frame.

*smallLabelBackground
>
> the background of the area of the node label that is not filled.

*smallLabelRelief
>
> how the label would appear visually (style).

*smallLabelForeground
>
> the color of node label text.

*smallLabelFont
>
> the font to use for the node label text.

*smallLabelFontWidth
>
> font width (in pixels) of *smallLabelFont

*smallLabelFontHeight
>
> font height (in pixels) of *smallLabelFont

*smallTextFont
>
> font to use for the text that appear inside a canvas.

*smallTextFontWidth
>
> font width (in pixels) of *smallTextFont.

*smallTextFontHeight
>
> font height (in pixels) of *smallTextFont.

*nodeColorTrough
> color of trough part (the /100 portion) of a canvas scale item.

*nodeColorSlider
> color of slider part (value portion) of a canvas scale item.

*nodeColorExtendedTrough
> color of extended trough (over 100 portion when value exceeds max) of a canvas scale item.

*nodeScaleFactor
> tells how much bigger you want the scale item on the canvas to appear. (1 means to keep size as is)

*nodeBoxFullMaxWidth

*nodeBoxFullMaxHeight
> maximum width and height (in pixels) of a node box in FULL mode.

*nodeBoxIconMaxWidth

*nodeBoxIconMaxHeight
> maximum width and height (in pixels) of a node box in ICON mode.

*nodeBoxMirrorMaxWidth

*nodeBoxMirrorMaxHeight
> maximum width and height (in pixels) of a node box displayed on a separate window (after it has been clicked with the mouse to obtain a bigger view)

*nodeBoxMirrorScaleFactor
> tells how much bigger you want the scale item on the canvas to appear while the node box is displayed on a separate window (1 means to keep size as is)

2.63.10.2   Server Box Properties

Resource names beginning with "*medium" apply to the properties of the server boxes. A server box is made of an outer frame where the server display label sits on top, a canvas filled with node boxes is on the middle, possibly some horizontal/vertical scrollbars, and a status label at the bottom.

*mediumLabelForeground
> color of text applied to the server display label and status label.

*mediumLabelBackground
> background color of the unfilled portions of the server display label and status label.

*mediumLabelBorderWidth
>   distance of the server display label and status label from other parts of the server box.

*mediumLabelRelief
>   how the server display label and status label appear visually (style).

*mediumLabelFont
>   font used for the text of the server display label and status label.

*mediumLabelFontWidth
>   font width (in pixels) of *mediumLabelFont.

*mediumLabelFontHeight
>   font height (in pixels) of *mediumLabelFont.

*mediumCanvasBorderWidth
>   the distance of the server box's canvas from the label widgets.

*mediumCanvasBackground
>   the background color of the canvas.

*mediumCanvasRelief
>   how the canvas appear visually (style).

*mediumScrollBorderWidth
>   distance of the scrollbars from the other parts of the server box.

*mediumScrollBackground
>   the background color of the scrollbars

*mediumScrollRelief
>   how the scrollbars appear visually.

*mediumBackground
>   the color of the server box frame.

*mediumBorderWidth
>   the distance of the server box from other boxes.

*mediumRelief
>   how the server box appears visually (style).

*serverBoxMaxWidth

*serverBoxMaxHeight
>   maximum width and height (in pixels) of a server box.

*serverBoxMaxNumNodeBoxesPerRow
>   maximum # of node boxes to appear in a row within a canvas.

2.63.10.3   Miscellaneous Properties

Resource names beginning with "*big" apply to the properties of a site box, as well as to widgets found outside of the server box and node box. This includes the dialog boxes that appear when the menu buttons of the main window are manipulated. The site box is the one that appears on the main region of xpbsmon.

*bigBackground
>   background color of the outer layer of the main window.

*bigForeground
>   color applied to regular text that appear outside of the node box and server box.

*bigBorderWidth
>   distance of the site box from the menu area and the color information area.

*bigRelief
>   how the site box is visually represented (style)

*bigActiveColor
>   the color applied to the background of a selection, a selected command button, or a selected scroll bar handle.

*bigShadingColor
>   a color shading applied to some of the frames to emphasize focus as well as decoration.

*bigSelectorColor
>   the color applied to the selector box of a radiobutton or checkbutton.

*bigDisabledColor
>   color applied to a disabled widget.

*bigLabelBackground
>   color applied to the unfilled portions of label widgets.

*bigLabelBorderWidth
>   distance from other widgets of a label widget.

*bigLabelRelief
>   how label widgets appear visually (style)

*bigLabelFont
>   font to use for labels.

*bigLabelFontWidth
>   font width (in pixels) of *bigLabelFont.

*bigLabelFontHeight
>   font height (in pixels) of *bigLabelFont.

*bigLabelForeground
> color applied to text that function as labels.

*bigCanvasBackground
> the color of the main region.

*bigCanvasRelief
> how the main region looks like visually (style)

*bigCanvasBorderWidth:
> distance of the main region from the menu and info regions.

*bigScrollBorderWidth
> if the main region has a scrollbar, this is its distance from other widgets appearing on the region.

*bigScrollBackground
> background color of the scrollbar appearing outside a server box and node box.

*bigScrollRelief
> how the scrollbar that appears outside a server box and node box looks like visually (style)

*bigTextFontWidth
> the font width (in pixels) of *bigTextFont

*bigTextFontHeight
> the font height (in pixels) of *bigTextFont

*siteBoxMaxWidth
> maximum width (in pixels) of the site box.

*siteBoxMaxHeight
> maximum height (in pixels) of the site box.

*siteBoxMaxNumServerBoxesPerRow
> maximum number of server boxes to appear in a row inside the site box.

*autoUpdate
> if set to *True*, then information about nodes is periodically gathered.

*autoUpdateMins
> the # of minutes between polling for data regarding nodes when *autoUpdate is set.

*siteInView
> the name of the site that should be in view

*rcSiteInfoDelimeterChar
> the separator character for each input within a curly-bracketed line of input of *siteInfo.

*sitesInfo

> *{<site1name><sep><site1-display-type><sep> <server-name><sep>*
> *<server-display-label><sep><nodename><sep> <nodetype><sep>*
> *<node-query-expr>}*
>
> *. . .*
>
> *{<site2name><sep><site2-display-type> <sep><server-name><sep>*
> *<server-display-label><sep><nodename> <sep><nodetype><sep>*
> *<node-query-expr>}*
>
> Information about a site where *<site1-display-type>* can be either
> {*FULL,ICON*}, *<nodetype>* can be {*MOM, NOMOM*}, and *<nodequery-*
> *expr>* has the format:
>
> *{ {<expr>} {expr-label} <output-format>}*
>
> where *<output-format>* could be {*TEXT, SCALE*}. It's probably better to
> use the Pref dialog boxes in order to specify a value for this.
>
> Example:

```
*rcSiteInfoDelimeterChar ;

*sitesInfo: {NAS;ICON;newton;Newton; newton3; NOMOM;} {Langley;
    FULL; db;DB; db.nas.nasa.gov; MOM; {{ ( availmem / totmem ) *
    100} {Memory Usage:} SCALE} {{ ( loadave / ncpus ) * 100}
    {Cpu Usage:} SCALE} {ncpus {Number of Cpus:} TEXT} {physmem
    {Physical Memory:} TEXT} {idletime {Idle Time (s):} TEXT}
    {loadave {Load Avg:} TEXT}} {NAS;ICON;newton;Newton;newton4;
    NOMOM;} {NAS;ICON;newton; Newton; newton1;NOMOM;} {NAS;
    ICON;newton;Newton; newton2;NOMOM;} {NAS;ICON;b0101;DB;aspa-
    sia.nas.nasa.gov; MOM;{{ ( availmem / totmem ) * 100} {Memory
    Usage:} SCALE} {{ ( loadave / ncpus ) * 100} {Cpu Usage:}
    SCALE} {ncpus {Number of Cpus:} TEXT} {physmem {Physical Mem-
    ory:} TEXT} {idletime {Idle Time (s):} TEXT} {loadave {Load
    Avg:} TEXT}} {NAS;ICON;newton;Newton;newton7;NOMOM;}
```

2.63.11   Exit Status

Upon successful processing, the xpbsmon exit status will be a value of zero.

If the xpbsmon command fails, the command exits with a value greater than zero.

If xpbsmon is querying a host running a server with an incompatible version, you may see
the following messages:

```
Internal error: pbsstatnode: End of File (15031)
```

The above message can be safely ignored.

2.63.12   See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide,
`pbs_sched(8B), pbs_mom(8B), pbs_tclapi(3B)`

Chapter 3

MOM Parameters

This chapter describes the configuration files used by MOM and lists the MOM configuration parameters that are found in the Version 1 MOM configuration file, `PBS_HOME/mom_priv/config`.

3.1 Syntax of MOM Configuration File

The Version 1 MOM configuration file contains parameter settings for the MOM on the local host.

Version 1 configuration files 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 to the `pbs_mom` command. Do not change the syntax of the Version 1 configuration file.

Each configuration item is listed on a single line, with its parts separated by white space. Comments begin with a hashmark ("#").

3.1.1 Externally-provided Resources

Externally-provided resources, for example dynamic resources such as scratch space, 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.

3.1.2   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
```

When you edit any PBS configuration file, make sure that you put a newline at the end of the file. The Notepad application does not automatically add a newline at the end of a file; you must explicitly add the newline.

3.2   Contents of MOM Configuration File

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. *new_action* is the site-

supplied script that replaces *default_action*. This is the complete list of values for *default_action*:

Table 3-1: 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 must be terminated by the script.
multinodebusy <timeout> requeue	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. Timeout is ignored. The only 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.

$aix_largepagemode <value>

Controls whether large page mode is available for PBS jobs on AIX. Settable by root only.

Format: Boolean.

Default: *False*.

$alps_client <path>

Cray only. Path to the Cray `apbasil` command. Must be full path to command.

Format: path to command

Default: None

$alps_release_timeout <timeout>

Cray only. Specifies the amount of time that PBS tries to release an ALPS reservation before giving up. After this amount of time has passed, PBS stops trying to release the ALPS reservation, the job exits, and the job's resources are released. PBS sends a HUP to the MoM so that she rereads the ALPS inventory to get the current available ALPS resources.

We recommend that the value for this parameter be twice the value for sus-
pectbegin.

Format: Seconds, specified as positive integer.

Default: *600* (10 minutes)

$checkpoint_path <path>

MOM passes this parameter to the checkpoint and restart scripts. This path
can be absolute or relative to PBS_HOME/mom_priv. Overrides default.
Overridden by path specified in the pbs_mom −C option and by
PBS_CHECKPOINT_PATH environment variable. Not supported in the
HPCBP MOM. See section 10.3.6.5, "Specifying Checkpoint Path" on
page 720 in the PBS Professional Administrator's Guide.

$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, the localhost, and if configured, the secondary server. 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.

Two hostnames are always allowed to connect to pbs_mom, "local-
host" and the name returned to MOM by the system call gethost-
name(). These hostnames do not need to be listed in the configuration
file.

The hosts listed as "clienthosts" 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.

$cpuset_error_action

When using a cpuset-enabled MOM, specifies the action taken when a
cpuset creation error occurs. Can take one of the following values:

continue
    The error is logged and the job is killed and requeued.

offline
    The vnodes on this host for this job are marked *offline*, and the job is
    requeued.

Format: *String*

Allowable values: *continue*, *offline*

Default: *offline*

$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 factor to a decimal value greater than 1.0. For example:

```
$cputmult 1.5
```

If MOM's system is slower, set 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
```

Not supported in the HPCBP MOM.

$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 The PBS Professional Administrator's Guide. Not supported in the HPCBP MOM.

$enforce mem

MOM will enforce each job's memory limit.

$enforce cpuaverage

MOM will enforce ncpus when the average CPU usage over a job's lifetime usage is greater than the job's limit.

$enforce average_trialperiod <seconds>

Modifies **cpuaverage**. Minimum number of seconds of job walltime before enforcement begins.

Format: Integer

Default: *120*

$enforce average_percent_over <percentage>

Modifies **cpuaverage**. Gives percentage by which a job may exceed its ncpus limit.

Format: Integer

Default: *50*

$enforce average_cpufactor <factor>
Modifies cpuaverage. The ncpus limit is multiplied by factor to produce actual limit.

Format: Float

Default: *1.025*

$enforce cpuburst
MOM will enforce the ncpus limit when CPU burst usage exceeds the job's limit.

$enforce delta_percent_over <percentage>
Modifies cpuburst. Gives percentage over limit to be allowed.

Format: Integer

Default: *50*

$enforce delta_cpufactor <factor>
Modifies cpuburst. The ncpus limit is multiplied by factor to produce actual limit.

Format: Float

Default: *1.5*

$enforce delta_weightup <factor>
Modifies cpuburst. Weighting factor for smoothing burst usage when average is increasing.

Format: Float

Default: *0.4*

$enforce delta_weightdown <factor>
Modifies cpuburst. Weighting factor for smoothing burst usage when average is decreasing.

Format: Float

Default: *0.4*

$ideal_load <load>
Defines the load below which the vnode is not considered to be busy. Used with the $max_load directive. Not supported in the HPCBP MOM.

Example:

```
$ideal_load 1.8
```

Format: Float

No default

$jobdir_root <stage_directory_root>

> Directory under which PBS creates job-specific staging and execution directories. PBS creates a job's staging and execution directory when the job's sandbox attribute is set to *PRIVATE*. If $jobdir_root is unset, it defaults to the job owner's home directory. In this case the user's home directory must exist. If stage_directory_root does not exist when MOM starts up, MOM will abort. If stage_directory_root does not exist when MOM tries to run a job, MOM will kill the job. Path must be owned by root, and permissions must be *1777*. On Windows, this directory should have *Full Control Permission* for the local Administrators group.

> Example:

> > `$jobdir_root /scratch/foo`

$kbd_idle <idle_wait> <min_use> <poll_interval>

> Declares that the vnode will be used for batch jobs during periods when the keyboard and mouse are not in use. Not supported in the HPCBP MOM.

> *idle_wait*

> > Time, in seconds, that the workstation keyboard and mouse must be idle before being considered available for batch jobs.

> > Must be set to non-zero value for cycle harvesting to be enabled.

> > Format: Integer

> > No default

> *min_use*

> > Time, in seconds, during which the workstation keyboard or mouse must continue to be in use before the workstation is determined to be unavailable for batch jobs.

> > Format: Integer

> > Default: *10*

> *poll_interval*

> > Interval, in seconds, at which MOM checks for keyboard and mouse activity.

> > Format: Integer

> > Default: *1*

> 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*.

> Log events:

Table 3-2: Event Classes

Name	Hex Value	Message Category
ERROR	*0001*	Internal errors
SYSTEM	*0002*	System errors
ADMIN	*0004*	Administrative events
JOB	*0008*	Job-related events
JOB_USAGE	*0010*	Job accounting info
SECURITY	*0020*	Security violations
SCHED	*0040*	Scheduler events
DEBUG	*0080*	Common debug messages
DEBUG2	*0100*	Uncommon debug messages
RESV	*0200*	Reservation-related info
DEBUG3	*0400*	Rare debug messages
DEBUG4	*0800*	Limit-related messages

$max_check_poll <seconds>

> Maximum time between polling cycles, in seconds. See section 3.6.1, "Configuring MOM's Polling Cycle" on page 53 in the PBS Professional Administrator's Guide. Minimum recommended value: 30 seconds.

> Minimum value: *1 second*

> Default: *120 seconds*

> Format: Integer

$max_load <load> [suspend]

> Defines the load above which the vnode is considered to be *busy*. Used with the $ideal_load directive. No new jobs are started on a *busy* vnode.

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.

We recommend setting this to a slightly higher value than the number of CPUs, for example *.25 + ncpus*.

Not supported in the HPCBP MOM.

Example:

> $max_load 3.5

Format: Float

Default: number of CPUs on machine

$min_check_poll <seconds>

Minimum time between polling cycles, in seconds. Must be greater than zero and less than $max_check_poll. See section 3.6.1, "Configuring MOM's Polling Cycle" on page 53 in the PBS Professional Administrator's Guide. Minimum recommended value: 10 seconds.

Minimum value: *1 second*

Default: *10 seconds*

Format: Integer

$prologalarm <timeout>

Defines the maximum number of seconds the prologue and epilogue may run before timing out.

Example:

> $prologalarm 30

Format: Integer

Default: *30*

$reject_root_scripts <True|False>

When set to *True*, MoM won't acquire any new hook scripts, and MoM won't run job scripts that would execute as root or Admin.

Format: *Boolean*

Default: *False*

$restart_background <value>

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. Not supported in the HPCBP MOM.

Format: Boolean

Default: *False*

$restart_transmogrify <value>

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.

Not supported in the HPCBP MOM.

Format: Boolean

Default: *False*

$restrict_user <value>

Controls whether users not submitting jobs have access to this machine. If value is *True*, restrictions are applied.

See $restrict_user_exceptions and $restrict_user_maxsysid.

Not supported on Windows.

Format: Boolean

Default: *False*

$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 of 10 names.

$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, if it exists, for the value. Otherwise the default is used.

Format: Integer

Default: *999*

$restricted <hostname>

The hostname is added to the list of hosts which will be allowed to connect to MOM without being required to use a privileged port. Queries from the hosts in the restricted list are only allowed access to information internal to this host, such as load average, memory available, etc. They may not run shell commands.

Hostnames can be wildcarded. For example, to allow queries from any host from the domain "xyz.com":

    $restricted *.xyz.com

$suspendsig <suspend_signal> [resume_signal]

Alternate signal suspend_signal is used to suspend jobs instead of SIG-STOP. Optional resume_signal is used to resume jobs instead of SIG-CONT. Not supported in the HPCBP MOM.

$tmpdir <directory>

Location where each job's scratch directory will be created.

PBS creates a temporary directory for use by the job, not by PBS. PBS creates the directory before the job is run and removes the directory and its contents when the job is finished. It is scratch space for use by the job. Permission must be 1777 on UNIX/Linux, writable by Everyone on Windows.

Not supported in the HPCBP MOM.

Example:

    $tmpdir /memfs

Default on UNIX: /tmp.

Default on Windows: value of the TMP environment variable.

$usecp <hostname:source_prefix> <destination_prefix>

MOM will use /bin/cp to deliver output files when the destination is a network mounted file system, or when the source and destination are both on the local host, or when the source_prefix can be replaced with the destination_prefix on hostname. Both source_prefix and destination_prefix are absolute pathnames of directories, not files.

Overrides PBS_RCP and PBS_SCP.

Not supported in the HPCBP MOM.

Use trailing slashes on both the source and destination. For example:

    $usecp HostA:/users/work/myproj/ /sharedwork/proj_results/

$vnodedef_additive

>Specifies whether MOM considers a vnode that appeared previously either in the inventory or in a vnode definition file, but that does not appear now, to be in her list of vnodes.
>
>When $vnodedef_additive is *True*, MOM treats missing vnodes as if they are still present, and continues to report them as if they are present. This means that the server does not mark missing vnodes as *stale*.
>
>When $vnodedef_additive is *False*, MOM does not list missing vnodes, the server's information is brought up to date with the inventory and vnode definition files, and the server marks missing vnodes as *stale*.
>
>Visible in configuration file on Cray only.
>
>Format: *Boolean*
>
>Default for MOM on Cray login node: *False*

$wallmult <factor>

>Each job's **walltime** usage is multiplied by this factor. For example:
>
>```
>$wallmult 1.5
>```

3.2.1   Cray-only Initialization Values

pbs_accounting_workload_mgmt <value>

>Controls whether CSA accounting is enabled. Name does not start with dollar sign. If set to "*1*", "*on*", or "*true*", CSA accounting is enabled. If set to "*0*", "*off*", or "*false*", accounting is disabled.
>
>Default: "*true*"; enabled.

3.2.2   SGI-only Initialization Values

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:
>
>Altix, supported versions of ProPack, Performance Suite
>
>>CPUSET_CPU_EXCLUSIVE|0
>>
>>Default: *CPUSET_CPU_EXCLUSIVE*
>
>ICE, supported versions of ProPack, Performance Suite
>
>>CPUSET_CPU_EXCLUSIVE|0
>>
>>Default: *0*

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.

> Example:

>> `cpuset_destroy_delay 10`

> Format: Integer

> Default for Altix: *0*

memreserved <megabytes>

> **Deprecated**. 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.

> For example,

>> `memreserved 16`

> Default: *0MB*

3.2.3   Static MOM Resources

Static resources local to the vnode 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
```

Chapter 4

Scheduler Parameters

This chapter lists the scheduler's configuration parameters. These parameters are found in the scheduler's configuration file, `PBS_HOME/sched_priv/sched_config`.

4.1 Format of Scheduler's Configuration File

4.1.1 Parameters with Separate Primetime and Non-primetime Specification

If a scheduler parameter can be specified separately for primetime and non-primetime, the format for the parameter is the following:

name: value [prime | non_prime | all | none]

- The *name* field cannot contain any whitespace.

- The *value* field may contain whitespace if the string is double-quoted. *value* can be: *True* | *False* | <number> | <string>. "*True*" and "*False*" are not case-sensitive.

- The third field allows you to specify that the setting is to apply during primetime, non-primetime, all the time, or none of the time. A blank third field is equivalent to "*all*" which means that it applies to both prime- and non-primetime.

  Acceptable values: "*all*", "*ALL*", "*none*", "*NONE*", "*prime*", "*PRIME*", "*non_prime*", "*NON_PRIME*"

4.1.2   Parameters without Separate Primetime and Non-primetime Specification

If a scheduler parameter cannot be specified separately for primetime and non-primetime, the format for the parameter is the same as the above, except that there is no third field.

4.1.3   Format Details

- Each entry must be a single, unbroken line.
- Entries must be quoted if they contain whitespace.
- Any line starting with a "#" is a comment, and is ignored.

4.1.4   Editing Configuration Files Under Windows

When you edit any PBS configuration file, make sure that you put a newline at the end of the file. The Notepad application does not automatically add a newline at the end of a file; you must explicitly add the newline.

4.2   Configuration Parameters

backfill

Toggle that controls whether PBS uses backfilling. If this is set to *True*, the scheduler attempts to schedule smaller jobs around higher-priority jobs 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 high-priority jobs will be considered first in the set to fit around the highest-priority job.

When this parameter is *True*, the scheduler backfills around starving jobs when help_starving_jobs is *True*.

Can be used with strict_ordering and help_starving_jobs

Format: Boolean

Default: *True all*

backfill_prime

The Scheduler will not run 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.

See also prime_spill, prime_exempt_anytime_queues.

Format: Boolean

Default: *False all*

by_queue

If set to *True*, all jobs that can be run from the highest-priority queue are run, then any jobs that can be run from the next queue are run, 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 parameter is overridden by the round_robin parameter when round_robin is set to *True*.

See section 4.8.4, "Examining Jobs Queue by Queue" on page 124 in the PBS Professional Administrator's Guide.

Format: Boolean

Default: *True all*

cpus_per_ssinode

Deprecated. Such configuration now occurs automatically.

dedicated_prefix

Queue names with this prefix are 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 section 4.8.10, "Dedicated Time" on page 148 in the PBS Professional Administrator's Guide.

Format: String

Default: *ded*

fair_share

Enables the fairshare algorithm, and turns on usage collecting. Jobs will be selected based on a function of their recent usage and priority (shares).

See section 4.8.18, "Using Fairshare" on page 165 in the PBS Professional Administrator's Guide.

Format: Boolean

Default: *False all*

fairshare_entity

Specifies the entity for which fairshare usage data will be collected. Can be one of "*euser*", "*egroup*", "*Account_Name*", "*queue*", or "*egroup:euser*".

Format: String

Default: *euser*

fairshare_enforce_no_shares

If this option is enabled, jobs whose entity has zero shares will never run. Requires fair_share to be enabled.

Format: Boolean

Default: *False*

fairshare_usage_res

Specifies the resource to collect and use in fairshare calculations and can be any valid PBS resource, including user-defined resources.

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.

See [section 4.8.18.5, "Tracking Resource Usage" on page 170 in the PBS Professional Administrator's Guide](#).

Format: String

Default: *cput*

half_life

The half-life for fairshare usage; after the amount of time specified, the fairshare usage is halved. Requires that fair_share be enabled.

See [section 4.8.18, "Using Fairshare" on page 165 in the PBS Professional Administrator's Guide](#).

Format: Duration

Default: *24:00:00*

help_starving_jobs

Setting this option enables starving job 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, and the server's eligible_time_enable attribute.

At each scheduler iteration, PBS calculates estimated.start_time and estimated.exec_vnode for starving jobs being backfilled around.

Format: Boolean

Default: *True all*

job_sort_key

> Selects how jobs should be sorted.   job_sort_key can be used to sort using either (a) resources or (b) special case sorting routines. Multiple job_sort_key entries can be used, one to a line, 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. 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"*

> *job_sort_key: "fair_share_perc HIGH|LOW"*

> *job_sort_key: "job_priority HIGH|LOW"*

> *job_sort_key: "preempt_priority HIGH|LOW"*

> Options: One of the following is required.

>> HIGH
>>> Specifies descending sort.

>> LOW
>>> Specifies ascending sort.

There are three special case sorting routines, which can be used instead of a specific PBS resource:

Table 4-1: Special Sorting in job_sort_key

Special Sort	Description
fair_share_perc *HIGH*	Sort based on the values in the resource_group file. If user A has more priority than user B, all of user A's jobs will always be run first. Past history is not used.
	This should only be used if entity share (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 <u>section 4.8.14, "Sorting Jobs by Entity Shares (Was Strict Priority)" on page 155 in the PBS Professional Administrator's Guide</u>
job_priority *HIGH|LOW*	Sort jobs by the job priority attribute regardless of job owner.
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 shows how to sort jobs so that those with high CPU count come first:

    job_sort_key: "ncpus HIGH" all

The following example shows how to sort jobs so that those with lower memory come first:

    job_sort_key: "mem LOW" prime

Format: Quoted string

Default: Not in force

key

> **Deprecated**. Use job_sort_key.

load_balancing

> When set to *True*, the scheduler takes into account the load average on vnodes as well as the resources listed in the `resources` line in `sched_config`. Load balancing can result in overloaded CPUs.
>
> See section 4.8.27, "Using Load Balancing" on page 187 in the PBS Professional Administrator's Guide.
>
> Format: Boolean
>
> Default: *False all*

load_balancing_rr

> **Deprecated**. To duplicate this setting, enable load_balancing and set smp_cluster_dist to round_robin.
>
> See section 4.8.27, "Using Load Balancing" on page 187 in the PBS Professional Administrator's Guide.

log_filter

> 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 section 13.4.4.1.iii, "Specifying Scheduler Log Events" on page 860 in the PBS Professional Administrator's Guide.
>
> Format: Integer
>
> Default: *3328*

max_starve

> The amount of time before a job is considered starving. This variable is used only if help_starving_jobs is set.
>
> Upper limit: None
>
> Format: Duration
>
> Default: *24:00:00*

mem_per_ssinode

> **Deprecated**. Such configuration now occurs automatically.

mom_resources

> 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.

Format: String

Default: Unset

node_sort_key

Defines sorting on resource or priority values on vnodes. Resource must be numerical, for example, *long* or *float*. 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>|sort_priority HIGH|LOW
node_sort_key: <resource> HIGH|LOW
         total|assigned|unused

total

 Use the resources_available value. This is the default setting when sorting on a resource.

assigned

 Use the resources_assigned value.

unused

 Use the value given by resources_available - resources_assigned.

sort_priority

 Sort vnodes by the value of the vnode priority attribute.

See section 4.8.48, "Sorting Vnodes on a Key" on page 277 in the PBS Professional Administrator's Guide.

Format: String

Default: *node_sort_key: sort_priority HIGH all*

nonprimetime_prefix

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 section 4.8.34, "Using Primetime and Holidays" on page 235 in the PBS Professional Administrator's Guide.

Format: String

Default: *np_*

peer_queue

Defines the mapping of a remote queue to a local queue for Peer Scheduling. Maximum number is 50 peer queues per scheduler.

See section 4.8.31, "Peer Scheduling" on page 199 in the PBS Professional Administrator's Guide.

Format: String

Default: unset

preemptive_sched

Enables job preemption.

See preempt_order and section 4.8.33, "Using Preemption" on page 222 in the PBS Professional Administrator's Guide for details.

Format: String

Default: *True all*

preempt_checkpoint

Deprecated. Add "C" to `preempt_order` parameter.

preempt_fairshare

Deprecated. Add "fairshare" to `preempt_prio` parameter.

preempt_order

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:*

Table 4-2: Preemption Order Symbols

Symbol	Action
S	suspend
C	checkpoint
R	Requeue

Usage: an ordering (*SCR*) optionally followed by a percentage of time remaining and another ordering.

Must be a quoted list("").

Example:

```
preempt_order: "SR"
```

This example specifies that PBS should first attempt to use suspension to preempt a job, and if that is unsuccessful, then requeue the job.

Example:

```
preempt_order: "SCR 80 SC 50 S"
```

This 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.

Format: Quoted list

Default: *SCR*

preempt_prio

Specifies the ordering of priority for different preemption levels. Two or more job types may be combined at the same priority level with a plus sign ("+") between them, using no whitespace. Comma-separated preemption levels are evaluated left to right, with higher priority to the left. The table below lists the six preemption levels. Note that any level not specified in the preempt_prio list is ignored.

Table 4-3: Preemption Levels

Level	Description
express_queue	Jobs in the express queues preempt other jobs. See preempt_queue_prio. Does not require by_queue or sort_queues to be *True*.
starving_jobs	When a job becomes starving it can preempt other jobs.
fairshare	When the entity owning a job exceeds its fair-share 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.

Example:

> `preempt_prio: "starving_jobs, normal_jobs, fairshare"`

In this example, the first line states that starving jobs have the highest priority, then normal jobs, and jobs whose entities are over their fairshare limit are third highest.

Example:

> `preempt_prio: "normal_jobs, starving_jobs+fairshare"`

This example shows that starving jobs whose entities are also over their fairshare limit are lower priority than normal jobs.

Format: Quoted list

Default: *express_queue, normal_jobs*

preempt_queue_prio

Specifies the minimum queue priority required for a queue to be classified as an express queue. Express queues do not require by_queue or sort_queues to be *True*.

Format: Integer

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.

If set to "*min_time_since_start*", first job preempted will be that with most recent start time.

If not set, preempted job will be that with longest running time.

Must be commented out in order to be unset; default scheduler configuration file has this parameter set to *min_time_since_start*.

Allowable values: "*min_time_since_start*", or no preempt_sort setting.

See section 4.8.33.7, "Sorting Within Preemption Level" on page 229 in the PBS Professional Administrator's Guide.

Format: String

Default: *min_time_since_start*

preempt_starving

Deprecated. Add "starving_jobs" to preempt_prio parameter.

preempt_suspend

Deprecated. Add an "*S*" to preempt_order parameter.

primetime_prefix

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 .

Format: String

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.

Format: 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 can be separately specified for prime- and non-primetime.   This option is only meaningful if backfill_prime is *True*.

See also backfill_prime, prime_exempt_anytime_queues.

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
prime_spill: 1:00:00 all
```

Format: Duration

Default: *00:00:00*

provision_policy

Specifies how vnodes are selected for provisioning. Can be set by Manager only; readable by all. Can be set to one of the following:

avoid_provision

> PBS first tries to satisfy the job's request from free vnodes that already have the requested AOE instantiated. PBS uses node_sort_key to sort these vnodes.

> If it cannot satisfy the job's request using vnodes that already have the requested AOE instantiated, it does the following:

>> PBS uses the server's node_sort_key to select the free vnodes that must be provisioned in order to run the job, choosing from any free vnodes, regardless of which AOE is instantiated on them.

>> Of the selected vnodes, PBS provisions any that do not have the requested AOE instantiated on them.

aggressive_provision

> PBS selects vnodes to be provisioned without considering which AOE is currently instantiated.

> PBS uses the server's node_sort_key to select the vnodes on which to run the job, choosing from any free vnodes, regardless of which AOE is instantiated on them. Of the selected vnodes, PBS provisions any that do not have the requested AOE instantiated on them.

> Format: String

> Default: *aggressive_provision*

resources

> Specifies those resources which are not to be over-allocated when scheduling jobs. Vnode-level boolean resources are automatically honored and do not need to be listed here. Limits are set by setting resources_available.resourceName on vnodes, queues, and the server. 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. If "host" is not added to the resources line, then 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.

> Format: String.

> Default: *ncpus, mem, arch, host, vnode*, *aoe*, *netwins*

resource_unset_infinite

> Resources in this list are treated as infinite if they are unset. Cannot be set differently for primetime and non-primetime.

Example:

```
resource_unset_infinite: "vmem, foo_licenses"
```

Format: Comma-delimited list of resources

Default: Empty list

round_robin

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 parameter.

When *True*, overrides the by_queue parameter.

Format: Boolean

Default: *False all*

server_dyn_res

Directs the Scheduler to replace the Server's resources_available values with new values returned by a site-specific external program.

See section 5.14.4.1, "Dynamic Server-level Resources" on page 327 in the PBS Professional Administrator's Guide for details of usage.

Format: String

Default: Unset.

smp_cluster_dist

Specifies how single-host jobs should be distributed to all hosts of the complex.

Options:

pack

Keep putting jobs onto one host until it is full and then move on to the next.

round_robin

Put one job on each vnode in turn before cycling back to the first one.

lowest_load

Put the job on the lowest-loaded host.

See section 4.8.42, "SMP Cluster Distribution" on page 267 in the PBS Professional Administrator's Guide and section 4.8.27, "Using Load Balancing" on page 187 in the PBS Professional Administrator's Guide.

Format: String

Default: *pack all*

sort_by

Deprecated. Use job_sort_key.

sort_queues

If 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.

When set to *False*, queues are not sorted.

This is a prime option, which means it can be selectively applied to primetime or non-primetime.

Note that the sorted order of queues is not taken into consideration unless by_queue is set to *True*.

See section 4.8.45, "Sorting Queues into Priority Order" on page 272 in the PBS Professional Administrator's Guide.

Format: Boolean

Default: *True ALL*

strict_fifo

Deprecated. Use strict_ordering.

strict_ordering

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. If a job due to run next cannot run, no job will run, unless backfilling is used, in which case jobs can be backfilled around the job that is due to run next.

See section 4.8.19, "FIFO Scheduling" on page 175 in the PBS Professional Administrator's Guide.

Example line in PBS_HOME/sched_priv/sched_config:

```
strict_ordering: True ALL
```

Format: Boolean.

Default: *False all*

sync_time

> **Deprecated**. The amount of time between writing the fairshare usage data to disk. Requires fair_share to be enabled.
>
> Format: Duration
>
> Default: *1:00:00*

unknown_shares

> 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 4.8.18, "Using Fairshare" on page 165 in the PBS Professional Administrator's Guide.
>
> Format: Integer
>
> Default: The unknown group gets 0 shares unless set.

Chapter 5

Resources

This chapter describes the resources provided by PBS Professional.

5.1   Resource Data Types

Data types for built-in and custom resource are described in section 7.1, "List of Formats", on page 403.

5.2   Advice on Using Resources

Resource names are case-insensitive.

The following advice will help you use resources.

5.2.1   Using **boolean** Resources

See "Boolean" on page 404.

Non-consumable.

5.2.2   Using **duration** Resources

See "Duration" on page 405.

Specifies a maximum time period the resource can be used.

Non-consumable.

5.2.3 Using **float** Resources

See "Float" on page 405.

Consumable.

5.2.4 Using **long** Resources

See "long" on page 407.

Consumable.

5.2.5 Using **size** Resources

See "Size" on page 408.

Consumable.

5.2.6 Using **string** Resources

See "String (resource value)" on page 408.

Non-consumable.

We do not recommend using non-printing characters.

When using `qsub -l <string resource>=<string value>`, you must escape string values for both `qsub` and the shell. Example:

```
qsub -lteststring='\"abc def\"'
```

The final quote should be single, not double.

Values are case-sensitive.

5.2.7 Using **string_array** Resources

See "string_array" on page 408.

Non-consumable. Resource request will succeed if request matches one of the values. Resource request can contain only one string.

A string array resource with one value works exactly like a string resource.

The value of resources_default.<string array resource> can only be one string.

5.3   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.

5.4   Built-in Resources

Different resources are available on different systems, often depending on he architecture of the computer itself. The table below lists the available resources that can be requested by PBS jobs on any system.

Table 5-1: Built-in Resources

Resource	Description
accelerator	Indicates whether this vnode is associated with an accelerator. Host-level. Can be requested only inside of a select statement. On Cray, this resource exists only when there is at least one associated accelerator. On Cray, this is set to *True* when there is at least one associated accelerator whose state is *UP*. On Cray, set to *False* when all associated accelerators are in state *DOWN*. Used for requesting accelerators. Format: Boolean. Python type: bool
accelerator_ memory	Indicates amount of memory for accelerator(s) associated with this vnode. Host-level. Can be requested only inside of a select statement. On Cray, PBS sets this resource only on vnodes with at least one accelerator with state = *UP*. For Cray, PBS sets this resource on the 0th NUMA node (the vnode with PBScrayseg=*0*), and the resource is shared by other vnodes on the compute node. For example, on vnodeA_2_0: `resources_available.accelerator_memory=4196mb` On vnodeA_2_1: `resources_available.accelerator_memory=@vnodeA_2_0` Consumable. Format: size. Python type: pbs.size

Resources

Table 5-1: Built-in Resources

Resource	Description
accelerator_ model	Indicates model of the accelerator(s) associated with this vnode. Host-level. On Cray, PBS sets this resource only on vnodes with at least one accelerator with state = *UP*. Can be requested only inside of a select statement. Non-consumable. Format: String. Python type: str
aoe	List of AOEs (Application Operating Environments) that can be instanti-ated on a vnode. Case-sensitive. An AOE is the environment that results from provisioning a vnode. Each job can request at most one AOE. Can-not be set on Server's host. Allowable values are site-dependent. Settable by Manager and Operator; visible to all. Non-consumable. Type: string array. Python type: str
arch	System architecture. 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. Can be requested only inside of a select statement. Non-consumable. Type: string. Python type: str
cput	Amount of CPU time used by the job for all processes on all vnodes. Establishes a job resource limit. Can be requested only outside of a select statement. Non-consumable. Type: duration. Python type: pbs.duration
exec_vnode	Read-only. The vnodes that PBS estimates this job will use. Cannot be requested for a job; used for reporting only. Type: string. Python type: str
file	Size of any single file that may be created by the job. Can be requested only outside of a select statement. Type: size. Python type: pbs.size
host	Name of execution host. Can be requested only inside of a select state-ment. Automatically set to the short form of the hostname in the Mom attribute. On Cray compute node, set to *<mpp_host>_<nid>*. On CLE 2.2, value is set to "*default*". Cannot be changed. Site-dependent. Type: string. Python type: str

Table 5-1: Built-in Resources

Resource	Description
max_walltime	Maximum walltime allowed for a shrink-to-fit job. Job's actual walltime is between max_walltime and min_walltime. PBS sets walltime for a shrink-to-fit job. If this resource is specifed, min_walltime must also be specified. Must be greater than or equal to min_walltime. Cannot be used for resources_min or resources_max. Cannot be set on job arrays or reservations. If not specified, PBS uses 5 years as the maximum time slot. Can be requested only outside of a select statement. Non-consumable. Default: None. Type: duration. Python type: pbs.duration
mem	Amount of physical memory i.e. workingset allocated to the job, either job-wide or vnode-level. Can be requested only inside of a select statement. Consumable. Type: size. Python type: pbs.size
min_walltime	Minimum walltime allowed for a shrink-to-fit job. When this resource is specified, job is a shrink-to-fit job. If this attribute is set, PBS sets the job's walltime. Job's actual walltime is between max_walltime and min_walltime. Must be less than or equal to max_walltime. Cannot be used for resources_min or resources_max. Cannot be set on job arrays or reservations. Can be requested only outside of a select statement. Non-consumable. Default: None. Type: duration. Python type: pbs.duration
mpiprocs	Number of MPI processes for this chunk. Defaults to 1 if ncpus > 0, 0 otherwise. Can be requested only inside of a select statement. Cannot use sum from chunks as job-wide limit. Type: integer. Python type: int 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.
mpparch	**Deprecated**. MPP compute node system type. Can be requested only outside of a select statement. Allowable values: XT or X2. Cray-only resource. Ignored at other systems. Type: string. Python type: str
mppdepth	**Deprecated**. Depth (number of threads) of each processor. Specifies the number of processors that each processing element will use. Can be requested only outside of a select statement. Cray-only resource. Ignored at other systems. Default: 1. Type: integer. Python type: int

Resources

Table 5-1: Built-in Resources

Resource	Description
mpphost	**Deprecated**. MPP host. Can be requested only outside of a select statement. Cray-only resource. Ignored at other systems. Type: string. Python type: str

mpplabels	**Deprecated**. List of node labels. Runs the application only on those nodes with the specified labels. Format: comma-separated list of labels and/or a range of labels. Any lists containing commas should be enclosed in quotes escaped by backslashes. For example:
	
	    `#PBS -l mpplabels=\"red,blue\"`
	
	or
	
	    `qsub -l mpplabels=\"red,blue\"`
	
	Can be requested only outside of a select statement. Cray-only resource. Ignored at other systems. Type: string. Python type: str, one of "soft" or "hard"

mppmem	**Deprecated**. The maximum memory for all applications. The per-processing-element maximum resident set size memory limit. Can be requested only outside of a select statement. Cray-only resource. Ignored at other systems. Type: size. Python type: pbs.size

mppnodes	**Deprecated**. Manual placement list consisting of a comma-separated list of nodes (node1,node2), a range of nodes (node1-node2), or a combination of both formats. Node values are expressed as decimal numbers. The first number in a range must be less than the second number (i.e., 8-6 is invalid). A complete node list is required. Any lists containing commas should be enclosed in quotes escaped by backslashes. For example:
	
	    `#PBS -l mppnodes=\"40-48,52-60,84,86,88,90\"`
	
	or
	
	    `qsub -l mppnodes=\"40-48,52-60,84,86,88,90\"`
	
	Can be requested only outside of a select statement. Cray-only resource. Ignored at other systems. Type: string. Python type: str

mppnppn	**Deprecated**. Number of processing elements (PEs) per node. Can be requested only outside of a select statement. Cray-only resource. Ignored at other systems. Type: integer. Python type: int

Table 5-1: Built-in Resources

Resource	Description
mppwidth	**Deprecated**. Number of processing elements (PEs) for the job. Can be requested only outside of a select statement. Cray-only resource. Ignored at other systems. Type: integer. Python type: int

naccelerators	Indicates number of accelerators on the host. Host-level. On Cray, should not be requested for jobs; PBS does not pass the request to ALPS. On Cray, PBS sets this resource only on vnodes whose hosts have at least one accelerator with state = *UP*. PBS sets this resource to the number of accelerators with state = *UP*. For Cray, PBS sets this resource on the 0th NUMA node (the vnode with PBScrayseg=*0*), and the resource is shared by other vnodes on the compute node.
	
	 For example, on vnodeA_2_0:
	
	     `resources_available.naccelerators=1`
	
	 On vnodeA_2_1:
	
	     `resources_available.naccelerators=@vnodeA_2_0`
	
	 Can be requested only inside of a select statement. Consumable. Format: long. Python type: int

nchunk	This is the number of chunks requested between plus symbols in a select statement. For example, if the select statement is `–lselect 4:ncpus=2+12:ncpus=8`, the value of nchunk for the first part is *4*, and for the second part it is *12*. The nchunk resource cannot be named in a select statement; it can only be specified by placing a number before the colon, as in the above example. When the number is omitted, nchunk is 1. Non-consumable. This resource can be used to specify the default number of chunks at the server or queue (replacing mppwidth.) Example: `set queue myqueue default_chunk.nchunk=2` Settable by Manager and Operator; readable by all. This resource cannot be used in server and queue resources_min and resources_max. Format: Integer. Python type: int. Default value: *1*
ncpus	Number of processors requested. Cannot be shared across vnodes. Can be requested only inside of a select statement. Consumable. Type: integer. Python type: int

Resources

Table 5-1: Built-in Resources

Resource	Description
netwins	Number of network windows on a high performance switch. Can be requested only inside of a select statement. Read-only. Type: integer. Python type: int
nice	Nice value under which the job is to be run. Host-dependent. Can be requested only outside of a select statement. Type: integer. Python type: int
nodect	**Deprecated**. Number of chunks in resource request from selection directive, or number of hosts requested from node specification. Otherwise defaults to value of 1. Can be requested only outside of a select statement. Read-only. Type: integer. Python type: int
nodes	**Deprecated**. Number of hosts requested. Integer.
ompthreads	Number of OpenMP threads for this chunk. Defaults to ncpus if not specified. Can be requested only inside of a select statement. Non-consumable. Cannot use sum from chunks as job-wide limit. Type: integer. Python type: int 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. Can be requested only outside of a select statement. Type: duration. Python type: pbs.duration
pmem	Amount of physical memory (workingset) for use by any single process of the job. Establishes a job resource limit. Can be requested only outside of a select statement. Non-consumable. Type: size. Python type: pbs.size
pvmem	Amount of virtual memory for use by the job. Establishes a job resource limit. Can be requested only outside of a select statement. Not consumable. Type: size. Python type: pbs.size
software	Site-specific software specification. Can be requested only outside of a select statement. Allowable values and effect on job placement are site-dependent. Type: string. Python type: pbs.software

Table 5-1: Built-in Resources

Resource	Description
start_time	Read-only. The estimated start time for this job. Cannot be requested for a job; used for reporting only. Type: long. Python type: int
vmem	Amount of virtual memory for use by all concurrent processes in the job. Establishes a per-chunk limit. Can be requested only inside of a select statement. Consumable. Type: size. Python type: pbs.size
vnode	Name of virtual node (vnode) on which to execute. For use inside chunks only. Site-dependent. Can be requested only inside of a select statement. Type: string. Python type: str See the `pbs_node_attributes(7B)` man page.
vntype	This resource represents the type of the vnode. Automatically set by PBS to one of two specific values for cray vnodes. Has no meaning for non-Cray vnodes. Can be requested only inside of a select statement. Non-consumable. Format: String array. Automatically assigned values for Cray vnodes: *cray_compute:* This vnode represents part of a compute node. *cray_login:* This vnode represents a login node. Default value: None. Python type: str
walltime	Actual elapsed (wall-clock, except during Daylight Savings transitions) time during which the job can run. Establishes a job resource limit. Can be requested only outside of a select statement. Non-consumable. Default: 5 years. Type: duration. Python type: pbs.duration

5.5   **Custom Cray Resources**

PBS provides custom resources specifically created for the Cray. They are listed in the following table:

Table 5-2: Custom Cray Resources

Resource Name	Description
PBScrayhost	On CLE 2.2, this is set to "*default*".
	On CLE 3.0 and higher, used to delineate a Cray system, containing ALPS, login nodes running PBS MOMs, and compute nodes, from a separate Cray system with a separate ALPS. Non-consumable. The value of PBScrayhost is set to the value of mpp_host for this system.
	Format: String.
	Default: CLE 2.2: "*default*"; CLE 3.0 and higher: None
PBScraylabel_ <label name>	Tracks labels applied to compute nodes. For each label on a compute node, PBS creates a custom resource whose name is a concatenation of *PBScraylabel_* and the name of the label. PBS sets the value of the resource to *True* on all vnodes representing the compute node.
	Name format: *PBScraylabel_<label name>*
	For example, if the label name is *Blue*, the name of this resource is PBScraylabel_Blue.
	Format: Boolean. Default: None
PBScraynid	Used to track the node ID of the associated compute node. All vnodes representing a particular compute node share a value for PBScraynid. Non-consumable.
	The value of PBScraynid is set to the value of node_id for this compute node.
	Non-consumable. Format: String. Default: None

Table 5-2: Custom Cray Resources

Resource Name	Description
PBScrayorder	Used to track the order in which compute nodes are listed in the Cray inventory. All vnodes associated with a particular compute node share a value for PBScrayorder. Non-consumable.
	Vnodes for the first compute node listed are assigned a value of 1 for PBScrayorder. The vnodes for each subsequent compute node listed are assigned a value one greater than the previous value.
	Do not use this resource in a resource request.
	Format: Integer. Default: None
PBScrayseg	Tracks the segment ordinal of the associated NUMA node. For the first NUMA node of a compute host, the segment ordinal is *0*, and the value of PBScrayseg for the associated vnode is *0*. For the second NUMA node, the segment ordinal is *1*, PBScrayseg is *1*, and so on. Non-consumable. Format: String. Default: None

5.6   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 5-3: Values for resources_available.arch

OS	Resource Label
AIX 5	aix4
CLE	XT
HP-UX 11	hpux11
Linux	linux
Linux with cpusets	linux_cpuset
Solaris	solaris7

Table 5-3: Values for resources_available.arch

OS	Resource Label
Unicos	unicos
Unicos MK2	unicosmk2
Unicos SMP	unicossmp

Chapter 6

Attributes

This chapter lists all of the PBS attributes. Attributes are listed by the PBS object they modify. For example, all attributes of jobs are listed in section 6.11, "Job Attributes", on page 375. Attributes are case-sensitive.

6.1   When Attribute Changes Take Effect

When you set the value of most attributes, the change takes place immediately. You do not need to restart any daemons in order to make the change.

6.2   How To Set Attributes

Most attributes are set using the `qmgr` command. However, some vnode attributes must be set using the `pbs_mom -s insert` command, to create a Version 2 configuration file. For information about these requirements, see section 3.5.2, "Choosing Configuration Method" on page 48 in the PBS Professional Administrator's Guide. The following are the instructions for setting all other attributes.

Attributes

To set the value of an attribute, use the `qmgr` command, either from the command line or within `qmgr`:

> `qmgr -c 'set <object> <attribute> = <value>'`
>
> `Qmgr: set <object> <attribute> = <value>`

To unset the value of an attribute:

> `qmgr -c 'unset <object> <attribute>'`
>
> `Qmgr: unset <object> <attribute>`

where *<object>* is one of *server*, *queue*, *hook*, *node*, or *sched*.

For example, to set resources_max.walltime at the server to be 24 hours:

> `Qmgr: set server resources_max.walltime = 24:00:00`

See ".

6.3   Viewing Attribute Values

If you want to view attribute values, the following commands are helpful:

    qstat; see "qstat" on page 194 of the PBS Professional Reference Guide

    qmgr; see "qmgr" on page 151 of the PBS Professional Reference Guide

    pbs_rstat; see "pbs_rstat" on page 78 of the PBS Professional Reference Guide

- To see server attributes, use one of the following:

    qstat –B –f

    Qmgr: list server

- To see queue attributes, use one of the following:

    qstat –Q –f <queue name>

    Qmgr: list queue <queue name>

- To see job attributes:

    qstat –f <job ID>

- To see hook attributes:

    Qmgr: list hook <hook name>

- To see scheduler attributes:

    Qmgr: list sched

- To see vnode attributes:

    Qmgr: list node <node name>

- To see reservation attributes:

    Qmgr: pbs_rstat –F

6.4   Attribute Table Format

In the following tables, the columns contain the following information:

Name

The name of the attribute

Description

A description of the attribute's function

Format

The attribute's format

Val/Opt

If the attribute can take only specific values or options, each is listed here

Value/Option Description

If the attribute can take only specific values or options, the behavior of each value or option is described here

Default Value, Def Val

The attribute's default value, if any

Python Type

The attribute's Python attribute value type

User, Oper, Mgr

Indicates the actions allowed for unprivileged users, Operators, and Managers

The following table shows the operations allowed and their symbols:

Table 6-1: User, Operator, Manager Actions

Symbol	Explanation
r	Entity can read attribute
w	Entity can directly set or alter attribute
s	Entity can set but not alter attribute
a	Entity can alter but not set attribute
i	Entity can indirectly set attribute
-	Entity cannot set or alter attribute, whether directly or indirectly

6.5   Caveats

- The Python types listed as Python dictionaries support a restricted set of operations. They can reference values by index. Other features, such as `has_key()`, are not available.

- Do not use `qmgr` to set attributes for reservation queues.

6.6　　Server Attributes

Server attributes are divided into these groups:

- Those that can be set by an operator or manager

- Those that are read-only

Server Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Defaul t Value	Python Type	User	Oper	Mgr
The following server attributes can be set by an operator or manager:									
acl_host_ena ble	Specifies whether the server obeys the host access control list in the acl_hosts server attribute.	*Boolean.*		When this attribute is *True*, the server limits host access according to the access control list.	*False*; all hosts allowed access	bool	r	r	r, w
acl_hosts	List of hosts from which services can be requested of this server. Requests from the Server's host always honored whether or not that host is in the list. This list contains the fully qualified domain names of the hosts. List is evaluated left-to-right; first match in list is used.	*String.* Form*: [+|-]host- name.domai n[,...]*			None; all hosts allowed access	pbs.acl	r	r	r, w
acl_resv_gro up_enable	Specifies whether the server obeys the group reservation access control list in the acl_resv_groups server attribute.	*Boolean*		When this attribute is *True*, the server limits group access according to the access control list.	*False*; all groups allowed access	bool	r	r	r, w

Server Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
acl_resv_gro ups	List of groups allowed or denied permission to create reservations in this PBS complex. The groups in the list are groups on the server host, not submission hosts. List is evaluated left-to-right; first match in list is used.	*String.* Form*: [+|-]group_nam e[,...]*				pbs.acl	r	r	r, w
acl_resv_host _enable	Specifies whether the server obeys the host reservation access control list in the acl_resv_hosts server attribute.	*Boolean*		When this attribute is *True*, the server limits host access according to the access control list.	*False*; access allowed from all hosts	bool	r	r	r, w
acl_resv_host s	List of hosts from which reservations can be created in this PBS complex. This list is made up of the fully-qualified domain names of the hosts. List is evaluated left-to-right; first match in list is used.	*String.* Form*: [+|-]host-name.domai n[,...]*			None; access allowed from all hosts	pbs.acl	r	r	r, w
acl_resv_user _enable	Specifies whether the server limits which users are allows to create reservations, according to the access control list in the acl_resv_users server attribute.	*Boolean*		When this attribute is *True*, the server limits user reservation creation according to the access control list.	*False*; all users are allowed to create reservations	bool	r	r	r, w

Attributes

Server Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
acl_resv_users	List of users allowed or denied permission to create reservations in this PBS complex. List is evaluated left-to-right; first match in list is used.	*String.* Form: *[+|-]user [@host][,...]*			None	pbs.acl	r	r	r, w
acl_user_enable	Specifies whether the server limits which users are allowed to run commands at the server, according to the control list in the acl_users server attribute.	*Boolean*		When this attribute is *True*, the server limits user access according to the access control list.	*False*; all users have access	bool	r	r	r, w
acl_users	List of users allowed or denied permission to run commands at this server. List is evaluated left-to-right; first match in list is used.	*String.* Form: *[+|-]user [@host][,...]*			None; all users allowed access	pbs.acl	r	r	r, w
backfill_depth	Modifies backfilling behavior. Sets the number of jobs that are to be backfilled around. Recommendation: set this to less than 100.	Integer. Must be >=1.	>=1	PBS backfills around the specified number of jobs.	Unset; backfill depth is *1*	int	r	r, w	r, w
			Unset	Backfill depth is set to *1*					
comment	Informational text.	*String* of any form			None	str	r	r, w	r, w

Server Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Defaul t Value	Python Type	User	Oper	Mgr
default_chunk	The list of resources which will be inserted into each chunk of a job's select specification if the corresponding resource is not specified by the user. This provides a means for a site to be sure a given resource is properly accounted for even if not specified by the user.	*String. Form: default_chu nk.<res>=<v al>,default_ chunk.<res> =<val>,...*			None	Dictionary: default_chunk[<resource name>]=<value > where <resource name> is any built-in or custom resource	r	r, w	r, w
default_node	No longer used.						-	-	-
default_qdel_ arguments	Argument to qdel command. Automatically added to all qdel commands. See qdel(1B). Overrides standard defaults. Overridden by arguments given on the command line.	*String. Form: - Wsuppress _mail=<N> .*			None	pbs.args	r	r, w	r, w
default_qsub _arguments	Arguments that are automatically added to the qsub command. Any valid arguments to qsub command, such as job attributes. Setting a job attribute via default_qsub_arguments sets that attribute for each job which does not explicitly override it. See qsub(1B). Settable by the administrator via the qmgr command. Overrides standard defaults. Overridden by arguments given on the command line and in script directives.	*String. Form: <option> <value> <option> <value> , e.g. -r y -N MyJob* To add to existing: Qmgr: s s default_qsub _arguments +=”<option> <value>”			None	pbs.args	r	r, w	r, w

Attributes

Server Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
default_queue	The name of the default target queue. Used for requests that do not specify a queue name. Must be set to an existing queue.	*Queue name*			None; must be set to an existing queue	pbs.queue	r	r, w	r, w
eligible_time_enable	Controls starving behavior. Viewable via `qstat` by job owner, Operator and Manager.	*Boolean*	*True*	The value of the job's eligible_time attribute is used for its starving time.	*False*	bool	r	r	r, w
			False	The value of *now() - etime* is used for the job's starving time.					
est_start_time_freq	Interval at which PBS calculates estimated start times and vnodes for all jobs. Best value is workload dependent. Recommendation: set this to two hours.	*Duration.* Expressed as an integer number of seconds or a *Duration*. See section , "Duration", on page 405	*>0*	PBS calculates estimated start times and vnodes for all jobs at the specified interval.	Unset	pbs.duration	r	r, w	r, w
			0	PBS calculates estimated start times and vnodes for all jobs at every scheduling cycle.					
			Unset	PBS does not calculate estimated start times or vnodes for all jobs; PBS calculates these only for the top N jobs specified in backfill_depth.					

Server Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
flatuid	Used for authorization allowing users to submit and alter jobs. Specifies whether user names are treated as being the same across the PBS server and all submission hosts in the PBS complex. Can be used to allow users without accounts at the Server host to submit jobs. If UserA has an account at the Server host, PBS requires that `UserA@<server host>` is the same as `UserA@<execution host>`.	*Boolean*	*True*	PBS assumes that `UserA@<submithost>` is same user as `UserA@<server>`. Jobs that run under the name of the job owner do not need authorization. A job submitted under a different username, by using the **u** option to the **qsub** command, requires authorization. Entries in `.rhosts` or `hosts.equiv` are not checked, so even if UserA@host1 has an entry for UserB@host2, UserB@host2 cannot operate on UserA@host1's jobs. User without account on Server can submit jobs.	*False*; authori-zation is required	bool	r	r	r, w
			False	PBS does not assume that `UserA@<submission host>` is the same user as `UserA@<server host>`. Jobs that run under the name of the job owner need authorization. Users must have accounts on the Server host to submit jobs.					
job_history_d uration	Specifies the length of time PBS will keep each job's history.	*Duration*			*Two weeks*	int	r	r	r, w

Attributes

Server Attributes										
Name	Description	Format	Val / Opt	Value/Option Description	Defaul t Value	Python Type	User	Oper	Mgr
job_history_e nable	Enables job history management. Setting this attribute to *True* enables job history management.	*Boolean*			*False*	bool	r	r	r, w
job_requeue_ timeout	Specifies the length of time that can be taken while requeueing a job. Minimum allowed value: 1 second. Maximum allowed value: 3 hours.	Duration			45 sec-onds	pbs.duration	r	r, w	r, w

				Server Attributes					
Name	Description	Format	Val / Opt	Value/Option Description	Defaul t Value	Python Type	User	Oper	Mgr
job_sort_form ula	Formula for computing job priorities. Described in the **PBS Professional Administrator's Guide**. If the attribute job_sort_formula is set, the scheduler will use the formula in it to compute job priorities. If it is unset, 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. Can be set by Manager or Operator. Viewable by users, Manager or Operator. The formula can be made up of expressions, where expressions contain terms which are added, subtracted, multiplied, or divided, and which can contain parentheses, exponents, and unary plus and minus.	*String* containing mathematical formula			Unset	pbs.job_sort_for mula	r, w	r, w	r, w
log_events	Specifies the types of events which are logged.	*Integer* representation of bit string			*511*	int	r	r, w	r, w

Attributes

Server Attributes										
Name	Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
mail_from	The username from which server-generated mail is sent to users. Mail is sent to this address upon failover. On Windows, requires fully qualified mail address.	*String*			*adm*	str	r	r	r, w
managers	List of PBS managers.	*String.* Form: "*user@host. sub.domain[,user@host. sub.domain. ..]*". The host, sub-domain, or domain name may be wildcarded with an aster-isk (*).			*Root on the server host*	pbs.acl	r	r	r, w
max_array_si ze	The maximum number of sub-jobs allowed in any array job.	*Integer*			*10000*	int	r	r, w	r, w
max_concurr ent_provision	The maximum number of vnodes allowed to be in the pro-cess of being provisioned. Can-not be set to zero.	Integer	>0		5	int	r	r	r, w

Server Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
max_group_res	Old limit attribute. Incompatible with new limit attributes. The maximum amount of the specified resource that any single group may consume in this PBS complex.	*String. Form: max_group _res.<resource>=<value>*	Any PBS resource, e.g. "ncpus", "mem", "pmem"		None	Dictionary: max_group_res[<resource name>]=<value> where <resource name> is any built-in or custom resource	r	r, w	r, w
max_group_res_soft	Old limit attribute. Incompatible with new limit attributes. The soft limit for the specified resource that any single group may consume in this complex. If a group is consuming more than this amount of the specified resource, their jobs are eligible to be preempted by jobs from groups who are not over their soft limit.	*String. Form: "max_group _res_soft.re source_name=value"*	Any PBS resource, e.g. "ncpus", "mem", "pmem", etc.		None	Dictionary: max_group_res _soft[<resource name>]=<value> where <resource name> is any built-in or custom resource	r	r, w	r, w
max_group_run	Old limit attribute. Incompatible with new limit attributes. The maximum number of jobs owned by the users in one group allowed to be running within this complex at one time.	*Integer*			No limit	int	r	r, w	r, w

Attributes

Server Attributes										
Name	Description	Format	Val / Opt	Value/Option Description	Defaul t Value	Python Type	User	Oper	Mgr
max_group_r un_soft	Old limit attribute. Incompati-ble with new limit attributes. The maximum number of jobs owned by the users in one group allowed to be running in this complex at one time. If a group has more than this number of jobs running, their jobs are eligi-ble to be preempted by jobs from groups who are not over their soft limit.	*Integer*			None	int	r	r, w	r, w
max_queued	Limit attribute. The maximum number of jobs allowed to be queued or running in the com-plex. Can be specified for projects, users, groups, or all. Cannot be used with old limit attributes.	Limit specifi-cation. See Chapter 7, "Formats", on page 403.			None	None	r	r, w	r, w
max_queued _res.<resourc e>	Limit attribute. The maximum amount of the specified resource allowed to be allocated to jobs queued or running in the com-plex. Can be specified for projects, users, groups, or all. Cannot be used with old limit attributes.	Limit specifi-cation. See Chapter 7, "Formats", on page 403.			None	None	r	r, w	r, w

Server Attributes										
Name	Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
max_run	Limit attribute. The maximum number of jobs allowed to be running in the complex. Can be specified for projects, users, groups, or all. Cannot be used with old limit attributes.	Limit specification. See Chapter 7, "Formats", on page 403.			None	None	r	r, w	r, w
max_run_res. <resource>	Limit attribute. The maximum amount of the specified resource allowed to be allocated to jobs running in the complex. Can be specified for projects, users, groups, or all. Cannot be used with old limit attributes.	Limit specification. See Chapter 7, "Formats", on page 403			None	None	r	r, w	r, w
max_run_res _soft.<resour ce>	Limit attribute. Soft limit on the amount of the specified resource allowed to be allocated to jobs running in the complex. Can be specified for projects, users, groups, or all. Cannot be used with old limit attributes.	Limit specification. See Chapter 7, "Formats", on page 403			None	None	r	r, w	r, w
max_run_soft	Limit attribute. Soft limit on the number of jobs allowed to be running in the complex. Can be specified for projects, users, groups, or all. Cannot be used with old limit attributes.	Limit specification. See Chapter 7, "Formats", on page 403.			None	None	r	r, w	r, w

Attributes

Server Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
max_running	Old limit attribute. Incompatible with new limit attributes. The maximum number of jobs allowed to be selected for execution at any given time, from all possible jobs.	*Integer*			None	int	r	r, w	r, w
max_user_res	Old limit attribute. Incompatible with new limit attributes. The maximum amount of the specified resource that any single user may consume within this complex.	*String. Form: max_user_res.<resource>=<value>*	Any PBS resource, e.g. "ncpus", "mem", "pmem", etc.		None	Dictionary: max_user_res[<resource name>]= <value> where <resource name> is any built-in or custom resource	r	r, w	r, w
max_user_res_soft	Old limit attribute. Incompatible with new limit attributes. The soft limit on the amount of the specified resource that any single user may consume within a complex. If a user is consuming more than this amount of the specified resource, their jobs are eligible to be preempted by jobs from users who are not over their soft limit.	*String. Form: max_user_res_soft.resource_name =value*	Any valid PBS resource, e.g. "ncpus", "mem", "pmem", etc		None	Dictionary: max_user_res_soft[<resource name>]=<value > where <resource name> is any built-in or custom resource	r	r, w	r, w

Server Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
max_user_run	Old limit attribute. Incompatible with new limit attributes. The maximum number of jobs owned by a single user allowed to be running within the complex at one time.	*Integer*			None	int	r	r, w	r, w
max_user_run_soft	Old limit attribute. Incompatible with new limit attributes. The soft limit on the number of jobs owned by a single user that are allowed to be running within this complex at one time. If a user has more than this number of jobs running, their jobs are eligible to be preempted by jobs from users who are not over their soft limit.	*Integer*			None	int	r	r, w	r, w
node_fail_requeue	Controls whether running jobs are automatically requeued or are deleted when the primary execution vnode fails. Number of seconds to wait after losing contact with Mother Superior before requeueing or deleting jobs. Reverts to default value when server is restarted.	*Integer. Seconds.*	Unset, zero, less than zero	Jobs are not requeued; they are left in the *Running* state until the execution vnode is recovered.	*310*	int	r	r, w	r, w
			Greater than zero	Jobs are requeued if they are marked as rerunnable, or are deleted when the node has been down for the specified number of seconds.					

Attributes

Server Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
node_group_ enable	Specifies whether placement sets (which includes node grouping) are enabled. See node_group_key server attribute.	*Boolean*		When set to *True*, placement sets are enabled.	*False*	bool	r	r, w	r, w
node_group_ key	Specifies the resources to use for placement sets (node grouping). Overridden by queue's node_group_key attribute. See node_group_enable server attribute.	*string_array* When specifying multiple resources, enclose the value in double quotes.			Unset	pbs.node_group _key	r	r, w	r, w
operators	List of PBS operators.	*String. Form: user@host. sub.domain[,user@host. sub.domain. ..]. The host, sub-domain, or domain name may be wildcarded with an asterisk (*).*			None	pbs.acl	r	r	r, w
pbs_license_f ile_location	**Deprecated**. Do not use.	-	-	-	-	-	-	-	-

Server Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
pbs_license_info	Location of license information. Can be port and hostname of license server, or local pathname to the actual license file associated with a license server. Delimiter between items is colon (":") for UNIX/Linux and semicolon (";") for Windows.	*String*. Port and host-name form: *<port1>@<host1>:<port2>@<host2>:...:<portN>@<hostN>:<path to license file>* where <host1>, <host2>, ... <hostN> can be IP addresses, and the license file can be listed first or last.			None	str	r	r	r, w
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.	*Integer*. Seconds.			*31536000 seconds (1 year).*	pbs.duration	r	r	r, w

Attributes

Server Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
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.	*Integer*			Maximum value for an integer	int	r	r	r, w
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.	*Integer*			*Zero*	int	r	r	r, w
query_other_jobs	Controls whether unprivileged users are allowed to select or query the status of jobs owned by other users.	*Boolean*		When this attribute is *True*, unprivileged users can query or select other users' jobs.	*True*	bool	r	r	r, w
require_cred	Specifies the Kerberos credential authentication method required. All jobs submitted without the specified credential will be rejected. See require_cred_enable. Depends on optional Kerberos and DCE support. Not supported under Windows.	*String*	*krb5* / *dce*		Unset	str	r	r	r, w

Server Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Defaul t Value	Python Type	User	Oper	Mgr
require_cred_ enable	Specifies whether the server is to use the Kerberos credential authentication method given in the require_cred server attribute. Depends on optional Kerberos and DCE support. Not supported under Windows.	*Boolean*		*True* means use the Kerberos authentication method specified.	*False*; no Kerberos credential authentication used	bool	r	r	r, w
reserve_retry _cutoff	The time period before the reservation start time during which PBS does not attempt to reconfirm a degraded reservation. When this value is changed, all degraded reservations use the new value. Must be greater than zero.	*Integer.* Seconds.			*7200 (2 hours)*	int	-	-	r, w
reserve_retry _init	The amount of time after a reservation becomes degraded that PBS waits before attempting to reconfirm the reservation. When this value is changed, only reservations that become degraded after the change use the new value. Must be greater than zero.	*Integer.* Seconds.			*7200 (2 hours)*	int	-	-	r, w

Attributes

Server Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Defaul t Value	Python Type	User	Oper	Mgr
resources_av ailable	The list of available resources and their values defined on the server. Each resource is listed on a separate line.	*String. Form: resources_a vail- able.<resour ce>=<value >*			None	Dictionary: resources_avail able[<resource name>]=<value > where <resource name> is any built-in or cus- tom resource	r	r, w	r, w
resources_co st	No longer used.						-	-	-
resources_de fault	The list of default job-wide resource values that are set as limits for jobs in this complex when a) the job does not specify a limit, and b) there is no queue default. The value for a string array, e.g. resources_default.<string array resource>, can contain only one string. For host-level resources, see the default_chunk.<resource> server attribute.	*String. Form: resources_d efault.resour ce_name=v alue[,...]*			No limit	Dictionary: resources_defa ult[<resource name>]=<value > where <resource name> is any built-in or cus- tom resource	r	r, w	r, w

Server Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
resources_max	The maximum amount of each resource that can be requested by any single job in this complex, if there is not a resources_max value defined for the queue at which the job is targeted. This attribute functions as a gating value for jobs entering the PBS complex.	*String. Form: resources_ max.resourc e_name=val ue[,...]*			No limit	Dictionary: resources_max[<resource name>]=<value > where <resource name> is any built-in or cus-tom resource	r	r, w	r, w
resv_enable	Specifies whether or not advance and standing reservations can be created in this complex.	*Boolean*		When set to *True*, new reserva-tions can be created. When changed from *True* to *False*, new reservations cannot be cre-ated, but existing reservations are honored.	*True*	bool	r	r	r, w
resv_post_pr ocessing_tim e	The amount of time allowed for reservations to clean up after run-ning jobs. Reservation duration and end time are extended by this amount of time. Jobs are not allowed to run during the cleanup period.	Duration			Unset; behaves as if zero	int	r	r, w	r, w
rpp_highwate r	The maximum number of RPP packets that can be in transit at any time.	*Integer*	Greater than or equal to one		*64*	int	r	r	r, w

Attributes

Server Attributes										
Name	Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
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*	Greater than or equal to zero		*10*	int	r	r	r, w
scheduler_iteration	The time between scheduling iterations.	*Integer. Seconds.*			*10 minutes (600 seconds)*	pbs.duration	r	r, w	r, w
scheduling	Enables scheduling of jobs. Specified by value of -a option to pbs_server command. If -a is not specified, value is taken from previous invocation of pbs_server.	*Boolean*		When this attribute is set to *True*, scheduling is enabled.	*False* if never set via pbs_server command.	bool	r	r, w	r, w
single_signon _password_e nable	Only used on systems requiring passwords, such as Windows or HPCBP. Incompatible with other systems. Specifies whether or not users must give a password for each job. Can be enabled only when no jobs exist, or when all jobs have a bad password hold ("*p*" hold). Can be disabled only when no jobs exist.	*Boolean.*	*True*	Users submitting jobs must specify a password only once; PBS remembers it for future job execution.	UNIX: *False* Windows: *True*	bool	r	r, w	r, w
			False	Users submitting jobs must specify a password for each job.					

Server Attributes										
Name	Description	Format	Val / Opt	Value/Option Description	Defaul t Value	Python Type	User	Oper	Mgr
system_cost	No longer used.						-	-	-
The following server attributes can be set by root only:									
acl_roots	List of users with root privilege who can submit and run jobs in this PBS complex. For any job whose owner is root or Adminis- trator, the job owner must be listed in this access control list, or the job is rejected. List is evaluated left-to-right; first match in list is used. Can be set or altered by root only, and only at the server host.	*String. Form: [+	-]user [@host][,...]*			None; no root jobs allowed	pbs.acl	r	r	r
The following server attributes are read-only:									
FLicenses	The number of floating licenses currently available for allocation to unlicensed CPUs. One license is required for each virtual CPU.	*Integer*			None	int	r	r	r

Attributes

Server Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Defaul t Value	Python Type	User	Oper	Mgr
license_count	The license_count attribute is made up of these numbers: Avail_Global, Avail_Local, Used, High_Use, Avail_Sockets, Unused_Sockets.	*String. Form: Avail_Globa l:<val> Avail_Local: <val> Used:<val> High_Use:< val> Avail_Socke ts:<val> Unused_So ckets:<val>*	Avail_ Global	The number of PBS CPU licenses still kept by the Altair license server (checked in.)	*Avail_ Glo- bal:0 Avail_L ocal:0 Used:0 High_ Use:0 Avail_ Sock- ets:0 Unuse d_Soc kets:0*	pbs.license_cou nt	r	r	r
			Avail_ Local	The number of PBS CPU licenses still kept by PBS (checked out.)					
			Used	The number of PBS CPU licenses currently in use.					
			High_ Use	The highest number of PBS CPU licenses ever checked out and used by the current instance of the PBS server.					
			Avail_ Sock- ets	The total number of socket licenses in the license file.					
			Unuse d_Soc kets	The number of unused socket licenses.					
pbs_version	The version of PBS for this server.	*String*			None	pbs.version	r	r	r
resources_as signed	The total of each type of resource allocated to jobs running in this complex.	*String. Form: resources_a ssigned.<re s>=<val>[,re sources_as signed.<res >=<val>,...]*			None	Dictionary: resources_assig ned[<resource name>]=<value > where <resource name> is any built-in or cus- tom resource	r	r	r

Server Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Defaul t Value	Python Type	User	Oper	Mgr
server_host	The name of the server. The server name is the same as the host name.	Hostname. If the server is listening to a non-standard port, the port number is appended, with a colon, to the host name. Example: `host.doma in:9999`			None	str	r	r	r

Attributes

Server Attributes										
Name	Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr	
server_state	The current state of the server:	*String*	*Hot_Start*	The server will run first any jobs that were running when it was shut down.	None	Server state constant pbs.SV_STATE_HOT	r	r	r	
			Idle	The server is running. Scheduling has been turned off.		Server state constant pbs.SV_STATE_IDLE				
			Active	The server is running. The scheduler is not in a scheduling cycle.		Server state constant pbs.SV_STATE_ACTIVE				
			Scheduling	The server is running. The scheduler is in a scheduling cycle.		Server state constant pbs.SV_STATE_ACTIVE				
			Terminating	The server is terminating. No additional jobs will be run.		Server state constant pbs.SV_STATE_SHUTIMM or pbs.SV_STATE_SHUTSIG				
			Terminating_Delayed	Server is terminating in delayed mode. No new jobs will be run. Server will shut down after all running jobs are finished.		Server state constant pbs.SV_STATE_SHUTDEL				

Server Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
state_count	List of the number of jobs in each state in the complex. Suspended jobs are counted as running.	*String.* Form*: tran-siting=<X>, queued=<Y >, ...*			None	pbs.state_count	r	r	r
total_jobs	The total number of jobs in the complex. If the job_history_enable attribute is set to *True*, this includes jobs that are finished, deleted, and moved.	*Integer*			None	int	r	r	r

6.7   Scheduler Attributes

Scheduler Attributes										
Name	Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
The following attributes are settable.										
do_not_span_psets	Specifies whether or not the scheduler requires the job to fit within one existing placement set.	Boolean	*True*	The job must fit in one existing placement set. All existing placement sets are checked. If the job fits in an occupied placement set, the job waits for the placement set to be available. If the job can't fit within a single placement set, it won't run.	*False*		r	r, w	r, w
			False	The scheduler first attempts to place the job in a single placement set, but if it cannot, it allows the job to span placement sets, running on whichever vnodes can satisfy the job's resource request.					
sched_cycle_length	The scheduler's maximum cycle length. Overwritten by the **-a alarm** option to `pbs_sched` command.	*Duration*			*20:00* (20 minutes)	None	r	r, w	r, w
The following attributes are read-only.										
pbs_version	The version of PBS for this scheduler.	*String*			None	None	-	r	r
sched_host	The hostname of the machine on which the scheduler runs.	*String*			*Servers host*	None	-	r	r

6.8   Reservation Attributes

Reservation attributes are divided into these groups:

- Those that can be set by users, operators, or managers

- Those that are read-only

Reservation Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	User	Oper	Mgr
The following attributes can be set by user, operator, or manager:									
Account_ Name	No longer used.						-	-	-
Authorize d_Groups	List of groups who can or can-not submit jobs to this reserva-tion. Group names are inter-preted relative to the server, not the submission host. List is evaluated left-to-right; first match in list is used. This list is used to set the reservation queue's acl_groups attribute. See the G option to the pbs_rsub command.	*String. Form:* *[+|-]group_ name,... , [+|-]group_ name]*			Jobs can be sub-mit-ted by all grou ps	pbs.acl	r, w	r, w	r, w

Attributes

Reservation Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	User	Oper	Mgr
Authorized_Hosts	The list of hosts from which jobs can and cannot be submitted to this reservation. List is evaluated left-to-right; first match in list is used. This list is used to set the reservation queue's acl_hosts attribute. See the H option to the pbs_rsub command.	*String. Form: [+|-]host-name, ... , [+|-]host-name*			Jobs can be sub-mit-ted from all hosts	pbs.acl	r, w	r, w	r, w

Reservation Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	User	Oper	Mgr
Authorized_Users	The list of users who can or cannot submit jobs to this reservation. This list is used to set the reservation queue's **acl_users** attribute. List is evaluated left-to-right; first match in list is used. See the U option to the `pbs_rsub` command.	*String*. Form: "*[+|-]user[hostname.domain], ...,[+|-]...*" where, '-' means "deny" and '+' means "allow". In addition, a single '*' may be used to wildcard various list entries			Reservation owner only	pbs.acl	r, w	r, w	r, w
group_list	No longer used.						-	-	-

Attributes

Reservation Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	User	Oper	Mgr
interactive	Number of seconds that the `pbs_rsub` command will block while waiting for confirmation or denial of the reservation. See the -I block_time option to the `pbs_rsub` command.	*Integer*	Less than zero	The reservation is automatically deleted if it cannot be confirmed in the time specified.	*Zero*	int	r, w	r, w	r, w
			Zero or greater than zero	The reservation is not automatically deleted if it cannot be confirmed in the time specified.					
Mail_Points	Sets the list of events for which mail is sent by the server. Mail is sent to the list of users specified in the Mail_Users attribute. See the m mail_points option to the `pbs_rsub` command.	*String* consisting of 1) one or more of the letters "*a*", "*b*", "*c*", "*e*", or 2) the string "*n*". Cannot use "*n*" with any other letter	*a*	Notify when reservation is terminated	"*ac*"	pbs.mail_points	r, w	r, w	r, w
			b	Notify when reservation period begins					
			c	Notify when reservation is confirmed					
			e	Notify when reservation period ends					
			n	Do not send mail. Cannot be used with other letters.					

				Reservation Attributes						
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	User	Oper	Mgr
Mail_Users	The set of users to whom mail is sent for the reservation events specified in the Mail_Points attribute. See the M mail_list option to the `pbs_rsub` command.	*String.* Form*: user@h ost[,user @host, ...]*			Reservation owner only	pbs.user_list	r, w	r, w	r, w
Priority	No longer used.						-	-	-
reserve_count	The count of occurrences in the standing reservation.	*Integer*				int	r, w	r, w	r, w
reserve_duration	Reservation duration in seconds. For a standing reservation, this is the duration for one occurrence.	*Integer*				pbs.duration	r, w	r, w	r, w
reserve_end	The date and time when an advance reservation or the soonest occurrence of a standing reservation ends.	*Date*				long	r, w	r, w	r, w

Attributes

Reservation Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	User	Oper	Mgr
Reserve_ Name	The name assigned to the reservation dur- ing creation, if specified. See the N option to the pbs_rsub command.	*String* up to 15 charac- ters. First character is alpha- betic			None	str	r, w	r, w	r, w

Reservation Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	User	Oper	Mgr
reserve_rrule	The rule that describes the recurrence pattern of a standing reservation. See the **r** option to the `pbs_rsub` command.	Either of two forms: *"FREQ= freq_spec; COUNT = count_spec; interval_spec"* or *"FREQ= freq_spec; UNTIL= until_spec; interval_spec"*	freq_spec	Frequency with which the standing reservation repeats. Valid values are: *WEEKLY|DAILY|HOURLY*	None	str	r, s	r, w	r, w
			count_spec	The exact number of occurrences. Number up to 4 digits in length. Format: *integer.*	None				
			interval_spec	Specifies interval. Format is one or both of: *BYDAY = MO|TU|WE|TH|FR|SA|SU or BYHOUR = 0|1|2|...|23*	None				
			until_spec	Occurrences will start up to but not after date and time specified. Format: *YYYYMMDD[THHMMSS]* Year-month-day part and hour-minute-second part separated by a capital *T.*	None				

Attributes

Reservation Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	User	Oper	Mgr
reserve_start	The date and time when the reservation period for the reservation or soonest occurrence begins.	*Date*			None	long	r, w	r, w	r, w
Resource _List	The list of resources allocated to the reservation. Jobs running in the reservation cannot use in aggregate more than the specified amount of a resource.	*String. Form: Resource_List.<res>=<val>, Resource_List.<res>=<val>, ...*			None	Dictionary: Resource_List[<resource name>]=<resource value> where <resource> name is any built-in or custom resource	r, w	r, w	r, w
User_List	No longer used.						-	-	-
The following reservation attributes are read-only:									
ctime	The time that the reservation was created.	*Date. Seconds since epoch.*			None	int	r	r	r
hash-name	No longer used.						-	-	-
mtime	The time that the reservation was last modified.	*Date*				int	r	r	r

Reservation Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	User	Oper	Mgr
Queue	Name of the reservation queue. Jobs that are to use resources belonging to this reservation are submitted to this queue.	Format for an advance reservation: *R<unique integer>* Format for a standing reservation: *S<unique integer>*				pbs.queue	r	r	r

Attributes

Reservation Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	User	Oper	Mgr
reserve_ID	The reservation identifier.	For an advance reservation: string of the form *R<unique integer>.server_name* For a standing reservation: string of the form *S[unique integer].server_name*				str	r	r	r
reserve_index	The index of the soonest occurrence of a standing reservation.	*Integer*				int	r	r	r

Reservation Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	User	Oper	Mgr
Reserve_Owner	The login name on the submission host of the user who created the reservation.	*String. Username@ host*			Login name of creator	str	r	r	r
reserve_retry	Time at which reservation will be reconfirmed.	*Integer. Seconds since epoch.*			None	int	r	r	r

Attributes

Reservation Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	User	Oper	Mgr
reserve_state	The state of the reservation.		NO RESV_ NONE	No reservation yet.	None	reservation state constant: pbs.RESV_STATE_NONE	r	r	r
			UN RESV_ UNCONFIRMED	Reservation request is awaiting confirmation.		reservation state constant: pbs.RESV_STATE_UNCONFIRMED			
			CO RESV_ CON-FIRMED	Resv. confirmed. All occurrences of standing resv. confirmed.		reservation state constant: pbs.RESV_STATE_CONFIRMED			
			WT RESV_ WAIT	Unused.		reservation state constant: pbs.RESV_STATE_WAIT			
			TR RESV_ TIME_TO_RUN	Start of the reservation period.		reservation state constant: pbs.RESV_STATE_TIME_TO_RUN			
			RN RESV_ RUN-NING	Resv. period has started; reservation is running.		reservation state constant: pbs.RESV_STATE_RUNNING			
			FN RESV_ FIN-ISHED	End of the reservation period.		reservation state constant: pbs.RESV_STATE_FINISHED			
			BD RESV_ BEING_DELETED	Reservation is being deleted.		reservation state constant: pbs.RESV_STATE_BEING_DELETED			
			DE RESV_ DELETED	Reservation has been deleted.		reservation state constant: pbs.RESV_STATE_DELETED			
			DJ RESV_ DELETING_JOBS	Jobs belonging to the reservation are being deleted		reservation state constant: pbs.RESV_STATE_DELETING_JOBS			
			DG DEGRADED	Reservation is degraded.		reservation state constant: pbs.RESV_STATE_DEGRADED			

Reservation Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	User	Oper	Mgr
reserve_substate	The substate of the reservation or occurrence. The substate is used internally by PBS.	*Integer*			None	int	r	r	r
reserve_type	No longer used.						-	-	-
resv_nodes	The list of each vnode and the resources allocated from it to satisfy the chunks requested for this reservation or occurrence.	*String. Form: "(vnode _name:r esource =value[:r esource =value].. .) [+(vnod e_name :resourc e=value[:resourc e=value])+..."*			None	pbs.exec_vnode	r	r	r
server	Name of server.	*String*			None	pbs.server	r	r	r
Variable_List	Not used						-	-	-

6.9   Queue Attributes

Queue attributes are divided into the following groups:

- Those that apply to both execution and routing queues

- Those that apply only to execution queues

- Those that apply only to routing queues

In the following table, Q Type indicates the type of queue to which the attribute applies: R (routing), E (execution):

Queue Attributes										
Name	Description	Format	Q Type	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
The following attributes apply to both routing and execution queues:										
acl_group_ enable	Controls whether the queue obeys the access control list defined in the acl_groups queue attribute.	*Boolean*	R, E		When set to *True*, the queue limits group access according to the access control list.	*False*; all groups allowed access	bool	r	r, w	r, w
acl_groups	List of groups which are allowed or denied access to this queue. The groups in the list are groups on the server host, not submitting hosts. List is evaluated left-to-right; first match in list is used.	*String*. Form: *[+|-] group-name[,...]*	R, E			None	pbs.acl	r	r, w	r, w
acl_host_e nable	Controls whether the queue obeys the access control list defined in the acl_hosts queue attribute.	*Boolean*	R, E		When set to *True*, the queue limits host access according to the access control list.	*False*; all hosts allowed access.	bool	r	r, w	r, w

Queue Attributes										
Name	Description	Format	Q Type	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
acl_hosts	List of hosts from which jobs may be submitted to this queue. List is evaluated left-to-right; first match in list is used.	*String.* Form: *[+|-]hostname[...]*	R, E			None	pbs.acl	r	r, w	r, w
acl_user_e nable	Controls whether the queue obeys the access control list defined in the acl_users queue attribute.	*Boolean*	R, E		When set to *True*, the queue limits user access according to the access control list.	*False*; all users allowed access	bool	r	r, w	r, w
acl_users	List of users allowed or denied access to this queue. List is evaluated left-to-right; first match in list is used.	*String.* Form: *[+|-]user [@host][,...]*	R, E			None	pbs.acl	r	r, w	r, w
enabled	Specifies whether this queue accepts new jobs.	*Boolean*	R, E	*True*	The queue is *enabled*. The queue accepts new jobs; new jobs can be enqueued.	*False*	bool	r	r, w	r, w
				False	The queue does not accept new jobs.					
from_route _only	Specifies whether this queue accepts jobs only from routing queues, or from both execution and routing queues.	*Boolean*	R, E	*True*	This queue accepts jobs only from routing queues.	*False*	bool	r	r	r, w
				False	This queue accepts jobs from both execution and routing queues.					
max_array _size	The maximum number of sub-jobs that are allowed in an array job.	*Integer*	R, E			No limit	int	r	r, w	r, w

Attributes

Queue Attributes											
Name	Description	Format	Q Type	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
max_queu able	Old limit attribute. Incompatible with new limit attributes. The maximum number of jobs allowed to reside in the queue at any given time.	*Integer*	R, E			No limit	int	r	r, w	r, w
max_queu ed	Limit attribute. The maximum number of jobs allowed to be queued in or running from this queue. Can be specified for projects, users, groups, or all. Cannot be used with old limit attributes.	Limit specification. See Chapter 7, "Formats", on page 403	R, E			None	None	r	r, w	r, w
max_queu ed_res.<re source>	Limit attribute. The maximum amount of the specified resource allowed to be allocated to jobs queued in or running from this queue. Can be specified for projects, users, groups, or all. Cannot be used with old limit attributes.	Limit specification. See Chapter 7, "Formats", on page 403	R, E			None	None	r	r, w	r, w
max_runnin g	Old limit attribute. Incompatible with new limit attributes.For an execution queue, this is the largest number of jobs allowed to be running at any given time. For a routing queue, this is the largest number of jobs allowed to be transiting from this queue at any given time.	*Integer*	R, E			None	int	r	r, w	r, w

Queue Attributes										
Name	Description	Format	Q Type	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
node_group_key	Specifies the resources to use for placement sets (node grouping). Overrides server's node_group_key attribute. Resources used must be of type string_array.	*string_array.* Comma-separated list of resource names. When specifying multiple resources, enclose value in double quotes.	R, E			None	pbs.node_group_key	r	r, w	r, w
Priority	The priority of this queue compared to other queues of the same type in this PBS complex. The value of priority has no meaning for routing queues.	*Integer. -1024 to 1023*	R, E			None	int	r	r, w	r, w
queue_type	The type of this queue. This attribute must be explicitly set at queue creation.	*String*	R, E	*e , execution*	Execution queue	None	PBS queue type constants: pbs.QUEUETYPE_EXECUTION or pbs.QUEUETYPE_ROUTE	r	r, w	r, w
				r , route	Routing queue					
require_cred	Specifies the credential type required. All jobs submitted to the named queue without the specified credential will be rejected. Not supported under Windows.	*String*	R, E	*krb5*		unset	str	r	r	r, w
				dce						

Attributes

					Queue Attributes					
Name	Description	Format	Q Type	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
require_cred_enable	Specifies whether the credential authentication method specified in the require_cred queue attribute is required for this queue. Not supported under Windows.	*Boolean*	R, E		When set to *True*, the credential authentication method is required	*False*	bool	r	r	r, w
resources_default	The list of default resource values which are set as limits for a job residing in this queue and for which the job did not specify a limit. If not set, the default limit for a job is determined by the first of the following attributes which is set: server's resources_default, queue's resources_max, server's resources_max. If none of these is set, the job gets unlimited resource usage.	*String. Form: resources_default.<resource name>=<value>, resources_default.<resource_name>=<value>, ...*	R, E			None	Dictionary: resources_default[<resource name>]=<value> where <resource name> is any built-in or custom resource	r	r, w	r, w
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.	*String. Form: resources_max.<resource_name >=<value>, resources_max.<resource_name >=<value>, ...*	R, E			None; infinite usage	Dictionary: resources_max[<resource name>]=<value> where <resource name> is any built-in or custom resource	r	r, w	r, w

Queue Attributes										
Name	Description	Format	Q Type	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
resources_min	The minimum amount of each resource that can be requested by a single job in this queue.	*String.* Form*: resources_max.<resource_name>=<value>, resources_max.<resource_name>=<value>, ...*	R, E			Zero usage	Dictionary: resources_min[<resource name>]=<value> where <resource name> is any built-in or custom resource	r	r, w	r, w
started	Specifies whether jobs in this queue can be scheduled for execution.	*Boolean*	R, E		Set to *True*: jobs in this queue can run	*False*	bool	r	r, w	r, w
state_count	The number of jobs in each state currently residing in this queue.	*String.* Form*: transiting=<val>, exiting=<val>, ...*	R, E			None	pbs.state_count	r	r	r
total_jobs	The number of jobs currently residing in this queue.	*Integer*	R, E			None	int	r	r	r
The following attributes apply only to execution queues:										
checkpoint _min	Specifies the minimum number of minutes of CPU time or walltime allowed between checkpoints of a job. If a user specifies a time less than this value, this value is used instead. The value given in checkpoint_min is used for both CPU minutes and walltime minutes.	*Integer*	E			None	pbs.duration	r	r, w	r, w

Attributes

					Queue Attributes					
Name	Description	Format	Q Type	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
default_chunk	The list of resources which will be inserted into each chunk of a job's select specification if the corresponding resource is not specified by the user. This provides a means for a site to be sure a given resource is properly accounted for even if not specified by the user.	*String.* Form*: default_chunk.<resource>=<value>, default_chunk.<resource>=<value>, ...*	E			None	Dictionary: default_chunk[<resource name>]=<value> where <resource name> is any built-in or custom resource	r	r, w	r, w
hasnodes	Describes whether this queue has associated vnodes.	*Boolean*	E		This attribute is set to *True* if there are vnodes associated with this queue.	*False*; no vnodes are associated with this queue	bool	r	r	r, i
kill_delay	The time delay between sending SIGTERM and SIGKILL when a `qdel` command is issued against a running job.	*Integer. Seconds.* Must be greater than or equal to *zero*.	E			*10 seconds*	pbs.duration	r	r, w	r, w
max_group _res	Old limit attribute. Incompatible with new limit attributes. The maximum amount of the specified resource that any single group may consume in a complex.	*String.* Form*: max_group_res. resource_name= value* *Example: set server max_group_res. ncpus=6*	E	Any PBS resource, e.g. "ncpus", "mem", "pmem", etc.		None	Dictionary: max_group_res [<resource name>]=<value> where <resource name> is any built-in or custom resource	r	r, w	r, w

Queue Attributes										
Name	Description	Format	Q Type	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
max_group _res_soft	Old limit attribute. Incompatible with new limit attributes. The soft limit on the amount of the specified resource that any single group may consume in a complex. If a group is consuming more than this amount of the specified resource, their jobs are eligible to be preempted by jobs from groups who are not over their soft limit.	*String*. Form*: max_group_res _soft.resource_n ame=value* *Example: set server max_group_res_ soft.ncpus=3*	E	Any valid PBS resource , e.g. "ncpus ", "mem", "pmem ", etc.		None	Dictionary: max_group_res _soft[<resourc e name>]=<valu e> where <resource name> is any built-in or custom resource	r	r, w	r, w
max_group _run	Old limit attribute. Incompatible with new limit attributes. The maximum number of jobs owned by a group that are allowed to be running from this queue at one time.	*Integer*	E			None	int	r	r, w	r, w
max_group _run_soft	Old limit attribute. Incompatible with new limit attributes. The maximum number of jobs owned by users in a single group that are allowed to be running from this queue at one time. If a group has more than this number of jobs running, their jobs are eligible to be preempted by jobs from groups who are not over their soft limit.	*Integer*	E			None	int	r	r, w	r, w

Attributes

Queue Attributes										
Name	Description	Format	Q Type	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
max_run	Limit attribute. The maximum number of jobs allowed to be running from this queue. Can be specified for projects, users, groups, or all. Cannot be used with old limit attributes.	Format: Limit specification. See Chapter 7, "Formats", on page 403	E			None	None	r	r, w	r, w
max_run_res.<resource>	Limit attribute. The maximum amount of the specified resource allowed to be allocated to jobs running from this queue. Can be specified for projects, users, groups, or all. Cannot be used with old limit attributes.	Format: Limit specification. See Chapter 7, "Formats", on page 403.	E			None	None	r	r, w	r, w
rux_run_res_soft.<resource>	Limit attribute. Soft limit on the amount of the specified resource allowed to be allocated to jobs running from this queue. Can be specified for projects, users, groups, or all. Cannot be used with old limit attributes.	Format: Limit specification. See Chapter 7, "Formats", on page 403.	E			None	None	r	r, w	r, w
max_run_soft	Limit attribute. Soft limit on the number of jobs allowed to be running from this queue. Can be specified for projects, users, groups, or all. Cannot be used with old limit attributes.	Format: Limit specification. See Chapter 7, "Formats", on page 403.	E			None	None	r	r, w	r, w

Queue Attributes										
Name	Description	Format	Q Type	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
max_user_res	Old limit attribute. Incompatible with new limit attributes. The maximum amount of the specified resource that any single user may consume.	*String. Form: max_user_res.r esource_name= value Example: set server max_user_res.n cpus=6*	E	any PBS resource, e.g. "ncpus", "mem", "pmem", etc		None	Dictionary: max_user_res[<resource name>] =<value> where <resource name> is any built-in or custom resource	r	r, w	r, w
max_user_res_soft	Old limit attribute. Incompatible with new limit attributes. The soft limit on the amount of the specified resource that any single user may consume. If a user is consuming more than this amount of the specified resource, their jobs are eligible to be preempted by jobs from users who are not over their soft limit.	*String. Form: max_user_res_ soft.resource_na me=value Example: set server max_user_res_s oft.ncpus=3*	E	any valid PBS resource, e.g. "ncpus", "mem", "pmem", etc		None	Dictionary: max_user_res_ soft[<resource name>] =<value> where <resource name> is any built-in or custom resource	r	r, w	r, w
max_user_run	Old limit attribute. Incompatible with new limit attributes. The maximum number of jobs owned by a single user that are allowed to be running from this queue at one time.	*Integer*	E			None	int	r	r, w	r, w

Attributes

Queue Attributes											
Name	Description	Format	Q Type	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr	
max_user_ run_soft	Old limit attribute. Incompatible with new limit attributes. The soft limit on the number of jobs owned by a single user that are allowed to be running from this queue at one time. If a user has more than this number of jobs running, their jobs are eligible to be preempted by jobs from users who are not over their soft limit.	*Integer*	E			None	int	r	r, w	r, w	
resources_ assigned	The total for each kind of resource allocated to jobs running from this queue.	*String.* Form*: resources_assig ned.<res>=<val> <new- line>resources_ assigned.<res>= <val><new- line>...*	E				Dictionary: resources_assi gned[<resource name>]=<valu e> where <resource name> is any built-in or custom resource	r	r	r	
resources_ available	The list of resources 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. See qmgr(1B).	*String.* Form*: resources_availa ble.<resource_n ame>=<value>< new- line>resources_ avail- able.<resource_ name>=<value> <newline>...*	E				Dictionary: resources_avail able[<resource name>]=<valu e> where <resource name> is any built-in or custom resource	r	r, w	r, w	

Queue Attributes										
Name	Description	Format	Q Type	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
The following attributes apply only to routing queues:										
alt_router	No longer used.							-	-	-
route_desti nations	The list of destinations to which jobs may be routed.	*String* of comma-separated strings. Form: *queue_name [@server_host [:port]] Example: Q1, Q2@remote, Q3@remote:155 01*	R			None. Must be set to at least one valid destina-tion	pbs.route_desti nations	r	r	r, w
route_held _jobs	Specifies whether jobs in the held state can be routed from this queue.	*Boolean*	R		If *True*, jobs with a hold can be routed from this queue.	*False*	bool	r	r, w	r, w
route_lifeti me	The maximum time a job is allowed to reside in a routing queue. If a job cannot be routed in this amount of time, the job is aborted.	*Integer. Seconds.*	R	>0	Number of seconds specified	Unset; infinite	pbs.duration	r	r, w	r, w
				0	Infinite					
				unset	Infinite					
route_retry _time	Time delay between routing retries. Typically used when the network between servers is down.	*Integer. Seconds.*	R			*30 sec-onds*	pbs.duration	r	r, w	r, w
route_waiti ng_jobs	Specifies whether jobs whose execution_time attribute value is in the future can be routed from this queue.	*Boolean*	R		If set to *True*, jobs with a future execution_time attribute can be routed from this queue.	*False*	bool	r	r, w	r, w

6.10　　Vnode Attributes

Vnode attributes are divided into the following groups:

- Those that can be set by an operator or manager

- Those that are read-only

Vnode Attributes								
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
The following attributes can be set by a manager or operator:								
com-ment	Information about this vnode. This attribute may be set by the manager to any string to inform users of any information relating to the node. If this attribute is not explicitly set, the PBS Server will use the attribute to pass information about the node status, specifically why the node is down. If the attribute is explicitly set by the manager, it will not be modified by the Server.	*String*			None	str	r	r	r, w
current_aoe	This attribute identifies the AOE currently instantiated on this vnode. Case-sensitive. Cannot be set on Server's host.	String			Unset	str	r	r	r, w
hpcbp_enable	Enables HPCBP features in Linux MOM.	*Boolean*		When set to *True*, HPCBP features are enabled.	*False*	bool	r	r	r, w

				Vnode Attributes							
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr		
hpcbp_ stage_p rotocol	Protocol and port number for staging files to and from HPC Basic Profile Server.	*String*			*scp*	str	r	r	r, w
hpcbp_ user_na me	User account with limited privilege, used for request-ing job/node status from HPC Basic Profile Server.	*String*			None	str	r	r	r, w
hpcbp_ webser vice_ad dress	URL for HPC Basic Pro-file Server .	*String.* The address must start with *https://.*			None	str	r	r	r, w
lictype	No longer used. (Was **dep-recated**.) Gone from .c file.					None	-	-	-
max_gr oup_run	The maximum number of jobs owned by any users in a single group allowed to run on this vnode at one time.	*Integer*			None	int	r	r, w	r, w
max_ru nning	The maximum number of jobs allowed to be run on this vnode at any given time.	*Integer*			None	int	r	r, w	r, w

Attributes

Vnode Attributes										
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
max_us er_run	The maximum number of jobs owned by a single user allowed to run on this vnode at one time.	*Integer*			Non e	int	r	r, w	r, w
Mom	Hostname of host on which MOM daemon runs. Can be explicitly set by Manager only via qmgr, and only at vnode creation. The server can set this to the FQDN of the host on which MOM runs, if the vnode name is the same as the hostname.	*String*		.	Val ue of vno de reso urce (vn ode nam e.)	str	r	r	r, w
no_mult inode_j obs	Controls whether jobs which request more than one chunk are allowed to execute on this vnode. Used for cycle harvesting.	*Boolean*		When set to *True*, jobs requesting more than one chunk are not allowed to execute on this vnode	*Fal se*	bool	r	r	r, w
ntype	The type of the vnode.	*String*	PBS	Normal vnode, not Globus.	*PB S*	int	r	r	r, w
			globus	PBS no longer supports Globus. The Globus func-tionality has been **removed** from PBS.					
pnames	The list of resources being used for placement sets. Not used for scheduling; advisory only.	*String*. Comma-separated list of resource names.			Non e	str	r	r	r, w

Vnode Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
Port	Port number on which MOM daemon listens. Can be explicitly set only via qmgr, and only at vnode creation.	*Integer*			*150 02*	int	-	r, w	r, w
Priority	The priority of this vnode compared with other vnodes.	*Integer.*	*[-1024, +1023]* inclusive		Non e	int	r	r, w	r, w
provisio n_enabl e	Controls whether this vnode can be provisioned. If set to *True*, this vnode may be provisioned. Cannot be set on Server's host.	*Boolean*			Uns et	bool	r	r	r, w
queue	The queue with which this vnode is associated. Each vnode can be associated with at most 1 queue. Queues can be associated with multiple vnodes. Any jobs in a queue that has associated vnodes can run only on those vnodes. If a vnode has an associated queue, only jobs in that queue can run on that vnode.	*String*	Name of queue	Only jobs in specified queue may run on this vnode.	Non e	pbs.qu eue	r	r	r, w
			Unset	Any job in any queue that does not have associated vnodes can run on this vnode.					

Attributes

Vnode Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
resourc es_avail able	The list of resources and the amounts available on this vnode. If not explicitly set, the amount shown is that reported by the **pbs_mom** running on the vnode. If a resource value is explicitly set, that value is retained across restarts.	*String.* Form: *resource s_availab le.<resou rce name>=< value>, resource s_availab le.<resou rce name> = <value>, ...*			None	Dictionary: resourc es_avai lable['<r esourc e>'] = < val> where *<resou rce>* is any custom or built-in resour ce	r	r, w	r, w
resv_en able	Controls whether the vnode can be used for advance and standing reservations. Reservations are incompatible with cycle harvesting.	*Boolean*		When set to *True*, this vnode can be used for reservations. Existing reservations are honored when this attribute is changed from *True* to *False*.	*True*	bool	r	r	r, w

Vnode Attributes									
Name	Description	Format	Val / Opt	Value/Option Description		Def Val	Python Type	Usr	Opr Mgr
sharing	Specifies whether more than one job at a time can use the resources of the vnode or the vnode's host. Either (1) the vnode or host is allocated exclusively to one job, or (2) the vnode's or host's unused resources are available to other jobs. Behavior is determined by a combination of vnode **sharing** attribute and job's placement directive.	String. Example: *vnode-name: shar-ing=force _excl*	*default_shar ed*	Defaults to *shared*		*def ault _sh are d*	int	r	r, w r, w
			default_excl	Defaults to *exclusive*					
			default_exclh ost	Entire host is assigned to the job unless the job's sharing request specifies otherwise					
			ignore_excl	Overrides any job *place=excl* setting					
			force_excl	Overrides any job *place=shared* setting					
			force_exclho st	The entire host is assigned to the job, regardless of the job's sharing request					
			Unset	Defaults to *shared*					

Behavior of vnode is shown in table below:

Value of **sharing**	Placement Request (`-lplace=`)				
	Vnode			Host	
	not specified	place=*shared*	place=*excl*	place=*exclhost*	place**!**=*exclhost*
not set	shared	shared	exclusive	exclusive	depends on place
default_shared	shared	shared	exclusive	exclusive	depends on place
default_excl	exclusive	shared	exclusive	exclusive	depends on place
default_exclhost	exclusive	shared	exclusive	exclusive	depends on place
ignore_excl	shared	shared	shared	shared	not exclusive
force_excl	exclusive	exclusive	exclusive	exclusive	not exclusive

Attributes

Vnode Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
state	Shows or sets the state of the vnode.	*String.* Comma-separated list of one or more states: st*ate[, state ...]*	*free*	Node is up and capable of accepting new job(s). Cannot be combined with other states.		int	r	r	r
			offline	Jobs are not to be assigned to this vnode. Can combine: *busy, job-busy, job-exclusive, resv_exclusive.*			r	r, w	r, w
			down	Node is not responding to queries from the Server. Cannot be combined with *free.*			r	r	r
			job-busy	All CPUs on the vnode are allocated to jobs. Can combine with: *offline, resv_exclusive.*			r	r	r
			job-exclusive	Entire vnode is exclusively allocated to one job at the job's request. Can combine with *offline* and *resv_exclusive*			r	r	r
			resv-exclusive	Running reservation has requested exclusive use of vnode. Can combine with *job-exclusive* and *offline*					
			busy	Vnode is reporting load average greater than allowed max. Can combine with *offline.*			r	r	r
			provisioning	Vnode is being provisioned. Cannot be combined with any other states.			r	r	r
			wait-provisioning	Vnode needs to be provisioned, but can't: limit reached for concurrent provisioning vnodes. Cannot be combined with other states. See max_concurrent_provision.					
			stale	Vnode was previously reported to server, but is no longer reported to server. Cannot combine with *free.*			r	r	r
			state-unknown	The Server has never been able to contact the vnode. Either MOM is not running on the vnode, the vnode hardware is down, or there is a network problem.			r	r	r

Vnode Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
The following attributes are read-only:									
in_multi vnode_ host	Specifies whether a vnode is part of a multi-vnoded host. Used internally. Do not set.	*Integer*	*unset*	Not part of a multi-vnode host		int			r, w
			1	Part of a multi-vnode host					
jobs	List of jobs running on the vnode.	*String*. Form: *#ljobid,...* where *#* is the num- ber of the processor.				str	r	r	r
license	Indicates whether a vnode is socket-licensed. Set by PBS.	*character*	*ll*	Indicates that vnode is socket licensed.	Uns et	str	r	r	r
license_ info	Indicates number of socket licenses assigned to vnode. Set by PBS.	*Integer*			Uns et	int	r	r	r
pbs_ver sion	The version of PBS for this MOM	*String*			Non e	str	r	r	r
pcpus	The number of physical CPUs on the vnode. This is set to the number of CPUs available when MOM starts. For a multi- ple-vnode MOM, only the natural vnode has pcpus.	*Integer*			Nu mbe r of CP Us on star- tup	int	r	r	r

Attributes

Vnode Attributes										
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
resv	List of advance and standing reservations pending on the vnode.	*String*. Comma-separated list of reservation IDs. Form: *RNNNN.<server>, RNNNN.<server>, ...*			None	str	r	r	r
resources_assigned	The total amount of each resource allocated to jobs running on this vnode.	*String*. Form: *resources_assigned.<resource>=<value>[,resources_ assigned.<resource>=<value>*			None	Dictionary: resources_available['<resource>'] = <val> where *<resource> is* any custom or built-in resource	r	r	r

Vnode Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
topolog y_info	Contains information intended to be used in hooks. Visible in and usable by hooks only.	*XML string*			Uns et	str	-	-	-

6.11 Job Attributes

Job attributes are divided into the following groups:

- Those that can be set by users, operators, or managers

- Those that are read-only

Job Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
The following job attributes can be set by users, operators, or managers:									
Account_N ame	Account to which the job is charged.	*String*			None	str	r, w	r, w	r, w

Attributes

Job Attributes										
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
argument_list	JSDL-encoded listing of arguments to job's executable.	Example: if arguments are "A B": `<jsdl-hpcpa:Argument>A</jsdl-hpcpa:Argument>` `<jsdl-hpcpa:Argument>B</jsdl-hpcpa:Argument>`			None	str	r, w	r, w	r, w
array	Indicates whether this is a job array.	*Boolean*		Set to *True* if this is an array job.	*False*	bool	r, s	r	r
array_indices_submitted	Applies only to job arrays. Complete list of indices of subjobs given at submission time.	*String*. Given as range, e.g. `1–100`			None	pbs.range	r, s	r	r
block	Specifies whether `qsub` will wait for the job to complete, and return the exit value of the job.	*Boolean*			*False*	int	r, s	r	r

Job Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
Checkpoint	Determines when the job will be check-pointed. An $action script is required to checkpoint the job.	*String*	c	Checkpoint at intervals, measured in CPU time, set on job's execution queue. If no interval set at queue, job is not checkpointed.	*u*	pbs.check-point	r, w	r, w	r, w
			c = *minutes of CPU time*	Checkpoint at intervals of specified number of minutes of job CPU time. This value must be > 0. If interval specified is less than that set on job's execution queue, queue's interval is used. Format: *integer*.					
			w	Checkpoint at intervals, measured in walltime, set on job's execution queue. If no interval set at queue, job is not checkpointed.					
			w = *min-utes of walltime*	Checkpoint at intervals of the specified number of minutes of job walltime. This value must be greater than zero. If the interval specified is less that that set on the execution queue in which the job resides, the queue's interval is used. Format: *integer*.					
			n	No checkpointing.					
			s	Checkpoint only when the server is shut down.					
			u	Unset. Defaults to behavior when interval argument is set to *s*.					
comment	Comment about job. Informational only.	*String*			None	str	r	r, w	r, w

Attributes

Job Attributes										
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr	
depend	Specifies inter-job dependencies.	*String.* Form: *type:arg_list [,type:arg_list ...]* arg_list is one or more PBS job IDs in the form: *jobid[:jobid ...]* Must be enclosed in double quotes if it contains commas. Example: "`type:jobid [,jobid...]`"	after: arg_list	This job may run at any point after all jobs in arg_list have started execution.	None; no depen-dencies	pbs.depend	r, w	r, w	r, w	
			afterok: arg_list	This job may run only after all jobs in arg_list have terminated with no errors.						
			afternotok: arg_list	This job may run only after all jobs in arg_list have terminated with errors.						
			afterany: arg_list	This job can run after all jobs in arg_list have finished execution, with or without errors. This job will not run if a job in the arg_list was killed.						
			before: arg_list	Jobs in arg_list may start once this job has started.						
			beforeok: arg_list	Jobs in arg_list may start once this job terminates without errors.						
			beforeno-tok: arg_list	If this job terminates execution with errors, then jobs in arg_list may begin.						
			beforeany: arg_list	Jobs in arg_list may begin execution once this job terminates execution, with or without errors.						
			on: count	This job may run after *count* depen-dencies on other jobs have been satis-fied. This type is used with one of the *before* types listed. Count is an integer greater than 0.						

Job Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
eligible_time	The amount of wall clock wait time a job has accrued while the job is blocked waiting for resources. For a job currently accruing *eligible_time*, if we were to add enough of the right type of resources, the job would start immediately. Viewable via `qstat -f`.	*Duration*			*Zero*	pbs.duration	r	r, w	r, w
Error_Path	The final path name for the file containing the job's standard error stream. If the output path is specified, but does not include a filename, the default filename is `jobid.ER`. If the path name is not specified, the default filename is `<job name>.e<sequence number>`. See the `qsub` and `qalter` commands.	*String*. Form: *[host-name:]path-name*	path_name; relative path	Path is relative to the current working directory of command executing on current host.	Path is current working directory where `qsub` is run. Filename is `<job name>.e<job number>`.	str	r, w	r, w	r, w
			path_name; absolute path	Path is absolute path on current host where command is executing.					
			host-name:path_name; relative path	Path is relative to user's home directory on specified host.					
			host-name:path_name; absolute path	Path is absolute path on named host.					
			No path	Path is current working directory where `qsub` is executed.					

Attributes

Job Attributes										
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
executable	JSDL-encoded listing of job's executable.	Example: if the executable is `ping`: `<jsdl-hpcpa:Exe-cut-able>ping</jsdl-hpcpa:Exe-cutable>`			None	str	r, w	r, w	r, w
Execution_ Time	Time after which the job may execute. Before this time, the job remains queued in the (*W*)ait state. Can be set when stage-in fails and PBS moves job start time out 30 minutes to allow user to fix problem.	*Datetime*. See Chapter 7, "For-mats", on page 403.			Unset; no delay	int	r, w	r, w	r, w

Job Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
exec_host	If the job is running, this is set to the name of the host or hosts on which the job is executing.	*String.* Form: *host/N[*C][+...]*, where "host" is the name of the host, "N" is task slot number starting at 0, on that node, and "C" is the number of CPUs allocated to the job. "*C" does not appear if its value is one.			None	pbs.exec_host	r	r, i	r, i

Job Attributes										
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr

exec_vnode	List of chunks for the job. Each chunk shows the name of the vnode(s) from which it is taken, along with the host-level, consumable resources allocated from that vnode, and any AOE provisioned on this vnode for this job.
	
	If a vnode is allocated to the job but no resources from the vnode are used by the job, the vnode name appears alone.
	
	If a chunk is split across vnodes, the name of each vnode and its resources appear inside one pair of parentheses, joined with a plus ("+") sign.
		Each chunk is enclosed in parentheses. Chunks are connected by plus signs. For a job which requested two chunks satisfied by resources from three vnodes, exec_vnode is:
		`(vnodeA:ncpus=N:mem=X)`
		`+`
		`(nodeB:ncpus=P:mem=Y+nodeC:mem=Z)`.
					None	pbs.exec_vnode	r	r, w	r, w

Job Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
group_list	A list of group names used to determine the group under which the job runs. When a job runs, the server selects a group name from the list according to the following ordered set of rules: 1. Select the group name for which the associated host name matches the name of the server host. 2. Select the group name which has no associated host name. 3. Use the login group for the user name under which the job will be run.	*String*. Form: *group_name [@host] [,group_name [@host]...]* Must be enclosed in double quotes if it contains commas.			None	pbs.group_list	r, w	r, w	r, w
Hold_Types	The set of holds currently applied to the job. If the set is not null, the job will not be scheduled for execution and is said to be in the hold state. The hold state takes precedence over the wait state.	*String*, made up of the letters 'n', 'o', 'p', 's', 'u'	*n*	No hold	*n*	pbs.hold_types	r, w	r, w	r, w
			o	Other hold					
			p	Bad password					
			s	System hold					
			u	User hold					

Attributes

Job Attributes										
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
interactive	Specifies whether the job is interactive. Can be set, but not altered, by unprivileged user.	*Boolean*		Set to *True* if this is an interactive job.	*False*	int	r, w	r	r
Job_Name	The job name. See the `qalter` and `qsub` commands.	*String* up to 15 characters, first character must be alphabetic or numeric			Base name of job script, or STDIN	str	r, w	r, w	r, w
Join_Path	Specifies whether the job's standard error and standard output streams are to be merged and placed in the file specified in the Output_Path job attribute.	*Boolean*		When set to *True*, the job's standard error and standard output streams are merged.	*False*	pbs.join_path	r, w	r, w	r, w

Job Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
Keep_Files	Specifies whether the standard output and/or standard error streams are retained on the execution host in the job's staging and execution directory after the job has executed. Otherwise these files are returned to the submission host. Keep_Files overrides the Output_Path and Error_Path attributes.	*String*. Can be "*o*", "*e*", "*oe*", "*eo*", or "*n*".	o	The standard output stream is retained. The filename is: `job_name.o<sequence num-ber>`	n	pbs.keep_files	r, w	r, w	r, w
			e	The standard error stream is retained. The filename is: `job_name.e<sequence num-ber>`					
			eo, oe	Both standard output and standard error streams are retained.					
			n	Neither stream is retained. Files returned to submission host.					
Mail_Points	Specifies state changes for which the server sends mail about the job.	*String*. Can be any of "*a*", "*b*", "*e*", or "*n*".	a	Mail is sent on job abort	a	pbs.mail_points	r, w	r, w	r, w
			b	Mail is sent at beginning of job					
			e	Mail is sent at end of job					
			n	No mail is sent. Cannot be used with other options.					
Mail_Users	The set of users to whom mail is sent when the job makes state changes specified in the Mail_Points job attribute.	*String*. Form: "*user@host[,user@host]*" Must be enclosed in double quotes if it contains commas.			Job owner only	pbs.email_list	r, w	r, w	r, w
no_stdio_sockets	Not used.						-	-	-

Attributes

Job Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
Output_Path	The final path name for the file containing the job's standard output stream. If the output path is specified, but does not include a file-name, the default file-name is `jobid.OU`. If the path name is not specified, the default filename is `<job name>.o<job num-ber>`. See the `qsub` and `qalter` com-mands.	*String*. Form: *[host-name:]path-name*	path_name ; relative path	Path is relative to the current working directory of command executing on current host.	Path is current work-ing direc-tory where `qsub` is run. File-name is `<job name>.o<job number>`.	str	r, w	r, w	r, w
			path_name ; absolute path	Path is absolute path on current host where command is executing.					
			host-name:path_name; rel-ative path	Path is relative to user's home directory on specified host.					
			host-name:path_name; absolute path	Path is absolute path on named host.					
			No path	Path is current working directory where `qsub` is executed.					
Priority	The scheduling priority for the job. Higher val-ues indicate greater pri-ority.	*Integer*. Form: *[+|-]nnnn*	[-1024, +1023] inclusive		Unset	int	r, w	r, w	r, w

Job Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
project	The job's project. A project is a way to tag jobs. Each job can belong to at most one project.	*String*. Can contain any characters except for the following: Slash ("/"), left bracket ("["), right bracket ("]"), double quote ("""), semico-lon (";"), colon (":"), vertical bar ("|"), left angle bracket ("<"), right angle bracket (">"), plus ("+"), comma (","), question mark ("?"), and aster-isk ("*").			*_pbs_project_default*	str	r, w	r, w	r, w
pset	Shows name of place-ment set used by the job. On BlueGene, specifies which partition should be used.	*String*				str	r	r	r, w

Job Attributes								
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
Rerunnable	Specifies whether the job can be rerun. Does not affect how a job is treated if the job could not begin execution. See section 3.13.8, "Marking a Job as "Rerunnable" or Not", on page 69 of the PBS Professional User's Guide. Job arrays are required to be rerunnable and are rerunnable by default.	*Character*: "y" or "n"	*y*	The job can be rerun.	y	bool	r, w	r, w	r, w
			n	Once the job starts running, it can never be rerun.					
Resource_ List	The list of resources required by the job. List is a set of name=value strings. The meaning of name and value is dependent upon defined resources. Each value establishes the limit of usage of that resource. If not set, the value for a resource may be deter-mined by a queue or server default estab-lished by the administra-tor. See Chapter 5, "Resources", on page 297.	*String. Form: Resource_List. <res>=<value> , Resource_List. <res>=<value> , ...*			None	Dictionary: Resource_Li st[<resource name>]=<va lue> where <resource name> is any built-in or custom resource	r, w	r, w	r, w

Job Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
sandbox	Specifies type of location PBS uses for job staging and execution. User-settable via `qsub -Wsandbox=<value>` or via a PBS directive. See the $jobdir_root MOM configuration option in `pbs_mom.8B`.	*String*	*PRIVATE*	PBS creates job-specific staging and execution directories under the directory specified in the $jobdir_root MOM configuration option.	Unset	str	r, w	r, w	r, w
			HOME or unset	PBS will use the job owner's home directory for staging and execution.					
sched_hint	No longer used.						-	-	-
Shell_Path _List	One or more absolute paths to the program(s) to process the job's script file.	*String*. Form: "*path[@host][, path[@host]...]*" Must be enclosed in double quotes if it contains commas.			User's login shell on execution host	pbs.path_list	r, w	r, w	r, w
stagein	The list of files to be staged in prior to job execution.	*String*. Form: "*execution_path@storage_host:storage_path*" Must be enclosed in double quotes if it contains commas.			None	pbs.staging_list	r, w	r, w	r, w

Attributes

				Job Attributes				
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr Opr Mgr	
stageout	The list of files to be staged out after job execution.	*String*. Form: *"execution_path@storage_host:storage_path"* Must be enclosed in double quotes if it contains commas.			None	pbs.staging_list	r, w / r, w / r, w	
Submit_arguments	Job submission arguments given on the `qsub` command line.	*String*			None	str	r, w / r, w / r, w	
umask	The initial umask of the job is set to the value of this attribute when the job is created. This may be changed by `umask` commands in the shell initialization files such as .`profile` or .`cshrc`.	*Decimal integer*			*077*	int	r, w / r, w / r, w	

Job Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
User_List	The list of users which determines the user name under which the job is run on a given host. When a job is to be executed, the server selects a user name from the list according to the following ordered set of rules: 1. Select the user name from the list for which the associated host name matches the name of the server. 2. Select the user name which has no associated host name; the wild card name. 3. Use the Job_Owner as the user name.	*String*. Form: "*user@host [,user@host...]*" Must be enclosed in double quotes if it contains commas. May be up to 15 characters in length.			Value of Job_O wner job attribut e	pbs.user_list	r, w	r, w	r, w
Variable_Li st	List of environment variables set in the job's execution environment. See the `qsub(1B)` command.	*String*. Form: "*name=value [,name=value...]*" Must be enclosed in double quotes if it contains commas.			None	Dictionary: Variable_List[<variable name>]=<va lue> where <resource name> is any built-in or custom resource	r, w	r, w	r, w

Attributes

Job Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
The following job attributes are read-only:									
accounting_id	Accounting ID for tracking accounting data not produced by PBS.	*String*			None	str	r	r	r
accrue_type	Indicates what kind of time the job is accruing.	*Integer*	*0 (initial_time)*	Job is accruing initial time.	2 (eligible_time)	int	-	-	r
			1 (ineligible_time)	Job is accruing ineligible time.					
			2 (eligible_time)	Job is accruing eligible time.					
			3 (run_time)	Job is accruing run time.					

Job Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
alt_id	For a few systems, the session id is insufficient to track which processes belong to the job. Where a different identifier is required, it is recorded in this attribute. If set, it will also be recorded in the end-of-job accounting record. For jobs running in CPU sets, the alt_id holds the set name in a form usable by the cpuset(1) command. For HPCBP, holds HPCBP server job ID. On Windows, holds PBS home directory.	*String.* May contain white spaces.			None	str	r	r	r
array_id	Applies only to subjobs. Array identifier of subjob.	*String*			None	str	r	r	r
array_index	Applies only to subjobs. Index number of subjob.	*String*			None	int	r	r	r
array_indices_remaining	Applies only to job arrays. List of indices of subjobs still queued.	*String.* Range or list of ranges, e.g. `500, 552, 596-1000.`			None	str	r	r	r

Attributes

Job Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
array_state _count	Applies only to job arrays. Lists number of subjobs in each state.	*String*			None	pbs.state_co unt	r	r	r
ctime	The time that the job was created.	*Integer.* Sec-onds since epoch.			None	int	r	r	r
egroup	If the job is queued, this attribute is set to the group name under which the job is to be run.	*String*			None	str	-	-	r
estimated	List of values associ-ated with job's esti-mated start time. Used to report job's **exec_vnode** and **start_time**. Can be set in a hook or via `qalter`, but PBS will overwrite the values.	*String.* Form*: esti-mated.<res>=< value>, esti-mated.<res>=< value>*	*exec_vnod e*	The estimated vnodes used by this job.	Unset	Dictionary: estimated. [<resource name>]=<valu e> where <resource name> is any resource.	r	r, w	r, w
			start_time	The estimated start time for this job.	Unset		r	r, w	r, w
etime	The time that the job became eligible to run when queued in an exe-cution queue.	*Integer.* Sec-onds since epoch.			None	int	r	r	r
euser	If the job is queued, this attribute is set to the user name under which the job is to be run.	*String*			None	str	-	-	r

Job Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
Exit_status	Exit status of job. Set to zero for successful execution. If any subjob of an array job has non-zero exit status, the array job has non-zero exit status.	*Integer*			None	int	r	r	r
forward_x11_cookie	Contains the X authorization cookie.	*String*			None	str	r	r	r
forward_x11_port	Contains the number of the port being listened to by the port forwarder on the submission host.	*Integer*			None	int	r	r	r
hashname	No longer used.						-	-	-
jobdir	Path of the job's staging and execution directory on the primary execution host. Either user's home, or private sandbox. Depends on value of **sandbox** attribute. Viewable via `qstat -f`.	*String*				str	r	r	r
Job_Owner	The login name on the submitting host of the user who submitted the batch job.	*String. <Username>@<submission host>*				str	r	r	r

Attributes

Job Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
job_state	The state of the job.	Character	*E* (Exiting)	The job has finished, with or without errors, and PBS is cleaning up post-execution.	None	PBS job state constant	r, i	r, i	r, i
			H (Held)	The job is held.					
			Q (Queued)	The job resides in an execution or routing queue pending execution or routing. It is not in held or waiting state.					
			R (Running)	The job is in an execution queue and is running.					
			S (Suspend)	The job was executing and has been suspended. The job retains its assigned resources but does not use CPU cycles or walltime.					
			T (Transiting)	The job is being routed or moved to a new destination.					
			U (User suspend)	The job was running on a workstation configured for cycle harvesting and the keyboard/mouse is currently busy. The job is suspended until the workstation has been idle for a configured amount of time.					
			W (Waiting)	The Execution_Time attribute contains a time in the future. Can be set when stage-in fails and PBS moves job start time out 30 minutes to allow user to fix problem.					

Job Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
mtime	The time that the job was last modified, changed state, or changed locations.	*Integer*. Seconds since epoch.			None	int	r	r	r
qtime	The time that the job entered the current queue.	*Integer*. Seconds since epoch.			None	int	r	r	r
Queue	The name of the queue in which the job currently resides.	*String*			None	pbs.queue	r	r	r
queue_rank	A number indicating the job's position within its queue. Only used internally by PBS.	*Integer*			None	int	-	-	r
queue_type	The type of queue in which the job is currently residing.	*Character*	E	Execution queue	None	PBS queue type constant.	-	-	r
			R	Routing queue					
resources_used	The amount of each resource used by the job.	*String.* Form: List of name=value pairs; format is *resources_used.<res>=<val>, resources_used.<res>=<val>*			None	Dictionary: resources_used [<resource name>]= <value> where <resource name> is any built-in or custom resource	r	r	r

Attributes

Job Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
run_count	The number of times the job has been executed.	*Integer*			Zero	int	-	-	r
schedselect	The union of the select specification of the job, and the queue and server defaults for resources in a chunk.				None	pbs.select	-	-	r
server	The name of the server which is currently managing the job. When the secondary server is running during failover, shows the name of the primary server. After a job is moved to another server, either via qmove or peer scheduling, shows the name of the new server.	*String*			None	pbs.server	r	r	r
session_id	If the job is running, this is set to the session id of the first executing task.				None	int	r	r	r

Job Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
Stageout_st atus	Status of stageout. If stageout succeeded, this is set to *1*. If stageout failed, this is set to *0*. Available only for finished jobs. Displayed only if set. If stageout fails for any subjob of an array job, the value of Stageout_status is zero for the array job.	*Integer*			None	int	r	r	r
stime	The time when the job started execution. Changes when job is restarted.	*Integer*. Seconds since epoch.			None	int	r	r	r
substate	The substate of the job. The substate is used internally by PBS.	*Integer*			None	int	r	r	r
sw_index	No longer used. Gone from .c file.						-	-	-

6.12   **Hook Attributes**

An unset hook attribute takes the default value for that attribute.

Hook attributes can be set by root only.

Hook Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	Usr	Opr	Mgr
The following attributes are settable and readable by root or the Admin at the local server only.									
Type	The type of the hook.	*String*	*site*		*site*				
enabled	Determines whether or not a hook is run when its triggering event occurs.	*Boolean*	*True*	Hook runs when triggering event occurs	*True*				
			False	Hook is not run when triggering event occurs					
freq	Frequency with which exechost_periodic hook runs	*Integer*		Number of seconds between triggers	*120*				
user	Specifies who executes the hook.	*String*	*pbsadmin*	Hook runs as root	*pbsadmin*				
			pbsuser	Hook runs as owner of job					

Hook Attributes								
Name	Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	Usr Opr Mgr	
event	List of events that trigger the hook. Can be operated on with the "=", "+=", and "-=" operators. The *provision* event cannot be combined with any other events.	*string_array*	*queuejob*	Triggered when job is queued	"" meaning hook is not triggered			
			modifyjob	Triggered when job is modified				
			movejob	Triggered when job is moved				
			resvsub	Triggered when reservation is created				
			runjob	Triggered when job is run				
			provision	Hook is master provisioning hook				
			execjob_begin	Triggered when MoM receives job				
			execjob_end	Triggered after job finishes or is killed				
			execjob_preterm	Triggered just before job is killed				
			execjob_prologue	Triggered just before first job process				
			execjob_epilogue	Triggered just after job runs successfully				
			exechost_periodic	Triggered at periodic interval on execution hosts				
			""	Hook is not triggered				
order	Indicates relative order of hook execution, for hooks sharing a trigger. Hooks with lower order values execute before those with higher values.	*Integer*			1			

Attributes

Hook Attributes									
Name	Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	Usr	Opr	Mgr
alarm	Specifies the number of seconds to allow a hook to run before the hook times out.	*Integer. Must be greater than zero.*			*30*				

Chapter 7

Formats

This chapter describes the formats used in PBS Professional.

7.1 List of Formats

PBS NAME

This is a generic term, used to describe various PBS entities. For example, attribute names are PBS NAMEs.

Must start with an alpha-numeric character, and may contain the following: alpha-numeric , . _ - # : / \ @

Specifically excluded: ` ~ ! $ % ^ & * ( ) + = < > ? ; ' " |

Accounting Log Entry

logfile-date-time;record-type;id-string;message-text
where

logfile-date-time
   Date and time stamp in the format:
   mm/dd/yyyy hh:mm:ss

record-type
   A single character indicating the type of record

id-string
   The job or reservation identifier

message-text
   Format: blank-separated *keyword=value* fields.

Message text is ASCII text.

Content depends on the record type.

Attribute Name

PBS NAME

Boolean

Name of resource is a string.

Values:

TRUE, True, true, T, t, Y, y, 1

FALSE, False, false, F, f, N, n, 0

Date

<Day of week> <Name of month> <Day of month> <HH:MM:SS> <YYYY>

Datetime

[[[[CC]YY]MM]DD]hhmm[.SS]

where

Table 7-1: Datetime Symbols

Symbol	Meaning
CC	Century
YY	Year
MM	Month
DD	Day of month
hh	Hour
mm	Minute
SS	Second

When setting the value, each portion of the date defaults to the current date, as long as the next-smaller portion is in the future. For example, if today is the 3rd of the month and the specified day DD is the 5th, the month MM will be set to the current month.

If a specified portion has already passed, the next-larger portion will be set to one after the current date. For example, if the day DD is not specified, but the hour hh is specified to be 10:00 a.m. and the current time is 11:00 a.m., the day DD will be set to tomorrow.

Unspecified portions default to *now*.

Destination Identifier

String used to specify a particular destination. The identifier may be specified in one of three forms:

<queue name>@<server name>

<queue name>

@<server name>

where *<queue name>* is an ASCII character string of up to 15 characters.

Valid characters are alphanumerics, the hyphen and the underscore. The string must begin with a letter.

Duration

A period of time, expressed either as

integer seconds

or

[[hours:]minutes:]seconds[.milliseconds]
in the form:

[[HH:]MM:]SS[.milliseconds]
Note that milliseconds are rounded to nearest second.

Float

Floating point. Allowable values: [+-] 0-9 [[0-9] ...][.][[0-9] ...]

Host Name

String of the form

name.domain

where *domain* is a hierarchical, dot-separated list of subdomains. Therefore, a host name cannot contain a dot, "." as a legal character other than as a subdomain separator.

The name must not contain the commercial at sign, "@", as this is often used to separate a file from the host in a remote file name.

A host name cannot contain a colon, ":".

The maximum length of a host name supported by PBS is defined by PBS_MAXHOSTNAME, currently set to *64*.

Job Identifier

sequence_number[.server_name][@server]

Format:

[0-9]+[\[[0-9]+\]].IP_HOSTNAME[.IP_DOMAINNAME]

Job Array Identifier

Job array identifiers are a sequence number followed by square brackets:

sequence_number[][.server_name][@server]

Example:

1234[ ]

Note that some shells require that you enclose a job array ID in double quotes.

Job Array Range

sequence_number[<first>-<last>][.server_name][@server]

first and *last* are the first and last indices of the subjobs.

Job Name, Job Array Name

A job name or job array name can be at most 15 characters. It must consist of printable, non-whitespace characters. The first character must be alphabetic, numeric, plus sign ("+"), dash or minus ("-"), or underscore ("_"). The name must not contain special characters.

Limit Specification

limit-spec=limit[, limit-spec=limit, ...]

where *limit-spec* is:

Table 7-2: Limit Specification Syntax

limit-spec
o:PBS_ALL
p:PBS_GENERIC
p:<project name>
u:PBS_GENERIC

Table 7-2: Limit Specification Syntax

limit-spec
u:<user name>
g:PBS_GENERIC
g:<group name>

See the PBS Professional Administrator's Guide for an explanation of what each limit-spec means.

logfile-date-time

Date and time stamp in the format:

mm/dd/yyyy hh:mm:ss

long

Long integer. Allowable values: 0-9 [[0-9] ...]

Queue Identifier

To specify a queue at the default server:

<queue name>

To specify all queues at a server:

@<server name>

To specify a queue at a specific server:

<queue name>@<server name>

Queue Name

PBS NAME

Resource Name

PBS NAME

Resource names are case-insensitive.

Resource Value

- PBS NAME, or

- Anything inside double quotes

The format of each data type is defined for that data type. For example, float resources are defined above, in "Float" on page 405.

Size

Number of bytes or words. The size of a word is the word size on the execution host.

Default: bytes

Format:

   integer[suffix]

where *suffix* can be one of the following:

Table 7-3: Size in Bytes

Suffix	Meaning	Size
b or w	Bytes or words	1
kb or kw	Kilobytes or kilowords	2 to the 10th, or 1024
mb or mw	Megabytes or mega-words	2 to the 20th, or 1,048,576
gb or gw	Gigabytes or giga-words.	2 to the 30th, or 1,073,741,824
tb or tw	Terabytes or tera-words.	2 to the 40th, or 1024 gigabytes
pb or pw	Petabytes or peta-words.	2 to the 50th, or 1,048,576 gigabytes

String (resource value)

Any character, including the space character.

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 resource values are case-sensitive.

string_array

Comma-separated list of strings. Strings in string_array may not contain commas.

Subjob Identifier

Subjob identifiers are a sequence number followed by square brackets enclosing the subjob's index:

sequence_number[index][.server_name][@server]

Example:

```
1234[99]
```

User Name

String up to 16 characters in length. PBS supports names containing any printable, non-whitespace character except the at sign ("@"). Your platform may place additional limitations on user names.

User Name, Windows

Must conform to the POSIX-1 standard for portability:

- The username must contain only alphanumeric characters, dot (.), underscore (_), and/or hyphen "-".

- The hyphen must not be the first letter of the username.

- If "@" appears in the username, then it will assumed to be in the context of a Windows domain account: username@domainname.

- An exception to the above rule is the space character, which is allowed. If a space character appears in a username string, then it will be displayed quoted and must be specified in a quoted manner.

Vnode Name

- For the natural vnode, the vnode name must conform to legal name for a host

- For other vnodes, the vnode name is a PBS NAME

Chapter 8

States

This chapter lists and describes the states in PBS Professional.

8.1   Job States

Job states are abbreviated to one character.

Table 8-1: Job States

State	Description
B	Job arrays only: job array has started
E	Job is exiting after having run
F	Job is finished. Job has completed execution, job failed during execution, or job was deleted.
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.
M	Job was moved to another server
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.

Table 8-1: Job States

State	Description
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.)

8.1.1   Job Substates

Job substates are numeric:

Table 8-2: 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

Table 8-2: Job Substates

Substate Number	Substate Description
37	job held - file stage in failed
41	job sent to MOM to run
42	Running
43	Suspended by Operator or Manager
44	PBS no longer supports Globus. The Globus functionality has been **removed** from PBS. 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
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

Table 8-2: Job Substates

Substate Number	Substate Description
71	Job is waiting for vnode(s) to be provisioned with its requested AOE.
153	(Set by MOM) Mother Superior waiting for delete ACK from sisters

8.2   Job Array States

Job array states map closely to job states except for the '*B*' state. The '*B*' state applies to job arrays and indicates that at least one subjob has left the queued state and is running or has run, but not all subjobs have run. Job arrays will never be in the '*R*', '*S*' or '*U*' states.

Table 8-3: Job Array States

State	Indication
B	The job array has started
E	All subjobs are finished and the server is cleaning up the job array
F	The job array is finished
H	The job array is held
Q	The job array is queued, or has been qrerun
T	The job array is in transit between servers
W	The job array has a wait time in the future

8.3 Subjob States

Subjobs can be in one of six states, listed here.

Table 8-4: Subjob States

State	Indication
E	Ending
F	Finished
Q	Queued
R	Running
S	Suspended
U	Suspended by keyboard activity
X	Expired or deleted; subjob has completed execution or been deleted

8.4 Server States

The state of the server is shown in the **server_state** server attribute. Possible values are:

Table 8-5: Server States

Hot_Start	The Server has been started so that it will run first any jobs that were running when the Server was shut down. Python type: pbs.SV_STATE_HOT
Idle	The Server is running. The Scheduler is between scheduling cycles. Python type: pbs.SV_STATE_IDLE
Scheduling	The Server is running. The Scheduler is in a scheduling cycle. Python type: pbs.SV_STATE_ACTIVE

Table 8-5: Server States

Terminating	The Server is terminating. Python type: pbs.SV_STATE_SHUTIMM or pbs.SV_STATE_SHUTSIG
Terminating_Delayed	The Server is terminating in delayed mode. No new jobs will be run, and the Server will shut down when the last running job finishes. Python type: pbs.SV_STATE_SHUTDEL

8.5 Vnode States

If a vnode's state attribute is unset, that is equivalent to the state being *free*. A vnode's state is shown in its state attribute, which can take on zero or more of the values listed here. Some vnode state values can be set simultaneously. Values are:

Table 8-6: Vnode States

State	Set By	Description	Can Combine With these States
busy	Server	Node is up and has load average greater than max_load, or is showing keyboard or mouse activity. 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, or when the mouse and keyboard have not shown any activity for a specified amount of time, the *busy* mark is removed. Consult your OS documentation to determine values that make sense.	*offline*

Table 8-6: Vnode States

State	Set By	Description	Can Combine With these States
down	Server	Node is not usable. Existing communication lost between Server and MOM.	Cannot be set with *free*
free	Server	Node is up and has available CPU(s). Server will mark a vnode "*free*" on first successful ping after vnode was "*down*".	None
job-busy	Server	Node is up and all CPUs are allocated to jobs.	*offline* *resv-exclusive*
job-exclusive	Server	Node is up and has been allocated exclusively to a single job.	*offline* *resv-exclusive*
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.	*busy* *job-busy* *job-exclusive* *resv-exclusive*
provisioning	Server	A vnode is in the provisioning state while it is in the process of being provisioned. No jobs are run on vnodes in the provisioning state.	Cannot be set with any other states
resv-exclusive	Server	Reservation has requested exclusive use of vnode, and reservation is running.	*job-exclusive,* *offline*

Table 8-6: Vnode States

State	Set By	Description	Can Combine With these States
stale	Server	MOM managing vnode is not reporting any information about this vnode, but was reporting it previously. Server can still communicate with MOM. A vnode becomes *stale* when: 1. A vnode is defined in the Server 2. MOM starts or restarts and reports a set of vnodes according to her configuration 3. A vnode which existed in the server earlier is not in the set being reported now by MOM 4. That vnode is marked "*stale*"	Cannot be set with *free*
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.	
unresolvable	Server	Server cannot resolve name of vnode	

Table 8-6: Vnode States

State	Set By	Description	Can Combine With these States
wait-provisioning	Server	There is a limit on the maximum number of vnodes that can be in the provisioning state. This limit is specified in the Server's max_concurrent_provision attribute. If a vnode is to be provisioned, but cannot because the number of concurrently provisioning vnodes has reached the specified maximum, the vnode goes into the *wait-provisioning state*. No jobs are run on vnodes in the *wait-provisioning* state.	Cannot be set with any other states

8.6   Reservation States

The following table shows the list of possible states for a reservation. The states that you will usually see are *CO, UN, BD*, and *RN*, although a reservation usually remains unconfirmed for too short a time to see that state.

Table 8-7: Reservation States

Code	Numeric	State	Description
BD	7	*RESV_BEING_DELETED*	Transitory state; reservation is being deleted
CO	2	*RESV_CONFIRMED*	Reservation confirmed
DG	10	*RESV_DEGRADED*	Vnode(s) allocated to reservation unavailable

Table 8-7: Reservation States

Code	Numeric	State	Description
DE	*8*	*RESV_DELETED*	Transitory state; reservation has been deleted
DJ	*9*	*RESV_DELETING_JOBS*	Jobs remaining after reservation's end time being deleted
FN	*6*	*RESV_FINISHED*	Transitory state; reservation's end time has arrived and reservation will be deleted
NO	*0*	*RESV_NONE*	No reservation yet
RN	*5*	*RESV_RUNNING*	Time period from reservation's start time to end time is being traversed
TR	*4*	*RESV_TIME_TO_RUN*	Transitory state; reservation's start time has arrived
UN	*1*	*RESV_UNCONFIRMED*	Reservation not confirmed
WT	*3*	*RESV_WAIT*	Unused

8.6.1   Degraded Reservation Substates

The following table shows states and substates for degraded reservations:

Table 8-8: Degraded Reservation States and Substates

		Reservation Time in Future	Reservation Time Is Now
Advance Reservation	State	*RESV_DEGRADED*	*RESV_RUNNING*
	Substate	*RESV_DEGRADED*	*RESV_DEGRADED*

Table 8-8: Degraded Reservation States and Substates

		Reservation Time in Future	Reservation Time Is Now
Soonest Occurrence	State	*RESV_DEGRADED*	*RESV_RUNNING*
	Sub-state	*RESV_DEGRADED*	*RESV_DEGRADED*
Non-soonest Occurrence Only	State	*RESV_CONFIRMED*	N/A
	Sub-state	*RESV_DEGRADED*	N/A

Chapter 9

Accounting Log

This chapter describes the accounting log. The PBS accounting log is a text file with each entry terminated by a newline.   There is no limit to the size of an entry.

9.1   Log Entry Format

The format of an entry is:

logfile-date-time;record-type;id-string;message-text

where

logfile-date-time

> Date and time stamp in the format:
>
> *mm/dd/yyyy hh:mm:ss*

record-type

> A single character indicating the type of record

id-string

> The job or reservation identifier

message-text

> Message text format is blank-separated *keyword=value* fields.
>
> Message text is ASCII text.
>
> Content depends on the record type.
>
> There is no dependable ordering of the content of each message.

9.1.1 **Resources**

Values for requested resources are written in the same units as those in the resource requests. Values for resources_used and resources_assigned are written in kb. A suffix is always written unless the quantity is measured in bytes.

For Resource_List and resources_used, there is one entry per resource, corresponding to the resources requested and used, respectively.

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.

9.2 **Record Types**

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 reservation. Possible information includes:

Table 9-1: Reservation Information

Attribute	Explanation
owner=*ownername*	Name of party who submitted the reservation request.
name=*reservation_name*	If submitter supplied a name string for the reservation.
queue=*queue_name*	The name of the reservation queue.
ctime=*creation_time*	Time at which the reservation was created; seconds since the epoch.
start=*period_start*	Time at which the reservation period is to start, in seconds since the epoch.
end=*period_end*	Time at which the reservation period is to end, in seconds since the epoch.

Table 9-1: Reservation Information

Attribute	Explanation
duration = *reservation_duration*	The duration specified or computed for the 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 users who are and are not authorized to submit jobs to the reservation.
Authorized_Groups= *group_list*	The list of groups who are and are not authorized to submit jobs to the reservation.
Authorized_Hosts= *host_list*	The list of hosts from which jobs may and may not be submitted to the reservation.
Resource_List= *resources_list*	List of resources requested by the reservation. Resources are listed individually as, for example: `Resource_List.ncpus = 16 Resource_List.mem = 1048676.`

C

Job was checkpointed and held.

D

Job was deleted by request. The *message-text* will contain `requestor=user@host` to identify who deleted the job.

E

Job ended (terminated execution). 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 9-2: Job End Record Contents

Attribute	Explanation
Exit_status= *<value>*	The exit status of the job. See [section 12.10, "Job Exit Codes" on page 829 in the PBS Professional Administrator's Guide]().
account= *<account_string>*	If job has an "account name" string.
accounting_id=*<jidvalue>*	CSA JID, job container ID
alt_id=*<ID>*	Optional alternate job identifier. Included only for certain systems: On Altix machines with supported versions of ProPack or Performance Suite, the alternate ID will show the path to the job's cpuset, starting with /PBSPro/.
ctime=*<time>*	Time in seconds when job was created (first submitted), seconds since epoch.
eligible_time=*<time>*	Amount of time job has waited while blocked on resources.
end=*<time>*	Time in seconds since epoch when this accounting record was written.
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.

Table 9-2: Job End Record Contents

Attribute	Explanation
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.
exec_vnode= *<vnode_list>*	List of each vnode with vnode-level, consumable resources from that vnode. (vnodeA+resource_name=P+..)+(vnodeB+resource_name=P+..) P is the amount of that resource allocated from that vnode.
group=*<groupname>*	The group name under which the job executed.
jobname=*<job_name>*	The name of the job.
project=*<project name>*	The job's project name at the end of the job.
qtime=*<time>*	Time in seconds when job was queued into current queue.
queue=*<queue_name>*	The name of the queue in which the job executed.
resources_used.<resource>=<*amount>*	Resources used by the job as reported by MOM. Typically includes ncpus, mem, vmem, cput, walltime, cpupercent. Walltime does not include suspended time.

Table 9-2: Job End Record Contents

Attribute	Explanation
resource_assigned.<resource> = <*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.
Resource_List.<resource>= <*amount*>	List of resources requested by the job. Resources are listed individually as, for example: `Resource_List.ncpus =16` `Resource_List.mem =1048676`.
run_count=<*value*>	The number of times the job has been executed.

Table 9-2: Job End Record Contents

Attribute	Explanation
session=*<sessionID>*	Session number of job.
start=*<time>*	Time when job execution started.
user=*<username>*	The user name under which the job executed.

F

Resource reservation period finished.

K

Scheduler or server requested removal of the reservation. The *message-text* field contains: `requestor=Server@host` or `requestor=Sched-uler@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: `requestor=user@host` to identify who deleted the resource reservation.

L

License information. This line in the log will have the following fields:

Log date; record type; keyword; specification for floating license; hour; day; month; max

The following table explains each field:

Table 9-3: Licensing Information 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

Table 9-3: Licensing Information in Accounting Log

Field	Explanation
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.

M

Information about moved jobs. Contains date, time, record type, job ID, destination.

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.

In this case, the *message-text* field contains information about the job. Possible information includes:

Table 9-4: Job Rerun Record Contents

Attribute	Explanation
Exit_status=*<value>*	The exit status of the previous start of the job. See section 12.10, "Job Exit Codes" on page 829 in the PBS Professional Administrator's Guide.
account= *<account_string>*	If job has an "account name" string.
accounting_id=*<jidvalue>*	CSA JID, job container ID

Table 9-4: Job Rerun Record Contents

Attribute	Explanation
alt_id=*<ID>*	Optional alternate job identifier. Included only for certain systems: On Altix machines with supported versions of ProPack or Performance Suite, the alternate ID will show the path to the job's cpuset, starting with /PBSPro/.
ctime=*<time>*	Time in seconds when job was created (first submitted), seconds since epoch.
eligible_time=*<time>*	Amount of time job has waited while blocked on resources, starting at creation time.
end=*<time>*	Time in seconds since epoch when this accounting record was written.
etime=*<time>*	Time in seconds when job most recently 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.
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 9-4: Job Rerun Record Contents

Attribute	Explanation
exec_vnode= *<vnode_list>*	List of each vnode with vnode-level, consumable resources from that vnode. (vnodeA+resource_name=P+..)+(vnodeB+resource_name=P+..) P is the amount of that resource allocated from that vnode.
group=*<groupname>*	The group name under which the job executed.
jobname=*<job_name>*	The name of the job.
project=*<project name>*	The job's project name.
qtime=*<time>*	Time in seconds when job was queued into current queue.
queue=*<queue_name>*	The name of the queue in which the job is enqueued.
Resource_List.<resource>= *<amount>*	List of resources requested by the job. Resources are listed individually as, for example: `Resource_List.ncpus =16 Resource_List.mem =1048676`.
run_count=*<value>*	The number of times the job has been executed.
session=*<sessionID>*	Session number of job.
start=*<time>*	Time when job execution started most recently.
user=*<username>*	The user name under which the job executed.

S

Job execution started. The *message-text* field contains:

Table 9-5: Job Start Record Contents

Attribute	Explanation
accounting_id= *identifier_string*	An identifier that is associated with system-generated accounting data. In the case where accounting is CSA, *identifier_string* is a job container identifier or JID created for the PBS job.
ctime=*time*	Time in seconds when job was created (first submitted).
etime=*time*	Time in seconds when job became eligible to run; no holds, etc.
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.
group=*groupname*	The group name under which the job will execute.
jobname=*job_name*	The name of the job.
project=<*project name*>	The job's project name at the start of the job.
qtime=*time*	Time in seconds when job was queued into current queue.
queue=*queue_name*	The name of the queue in which the job resides.
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.

Table 9-5: Job Start Record Contents

Attribute	Explanation
Resource_List.resource= *amount*	List of resources requested by the job. Resources are listed individually as, for example: `Resource_List.ncpus= 16 Resource_List.mem= 1048676.`
session=*sessionID*	Session number of job.
start=*time*	Time in seconds when job execution started.
user=*username*	The user name under which the job will execute.

T

Job was restarted from a checkpoint file.

U

Created unconfirmed reservation on Server. The *message-text* field contains `requestor=user@host` to identify who requested the reservation.

Y

Reservation confirmed by the Scheduler. The *message-text* field contains `requestor=user@host` to identify who requested the reservation.

Chapter 10

Example Configurations

This chapter shows some configuration-specific scenarios which will hopefully clarify any 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

10.1   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.
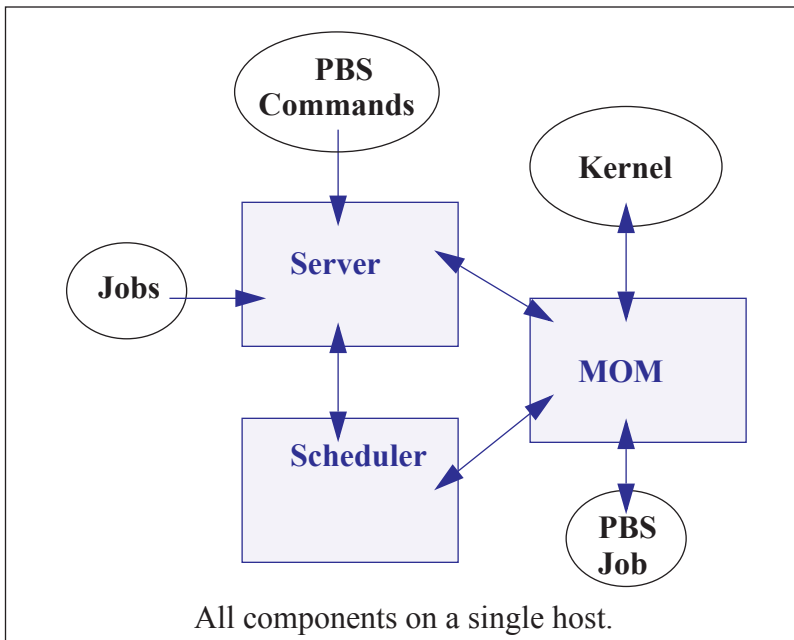
For this example, let's assume we have a 32-CPU server machine named "mars". We want users to log into mars and jobs will be run via PBS on mars.

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.

10.2   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.
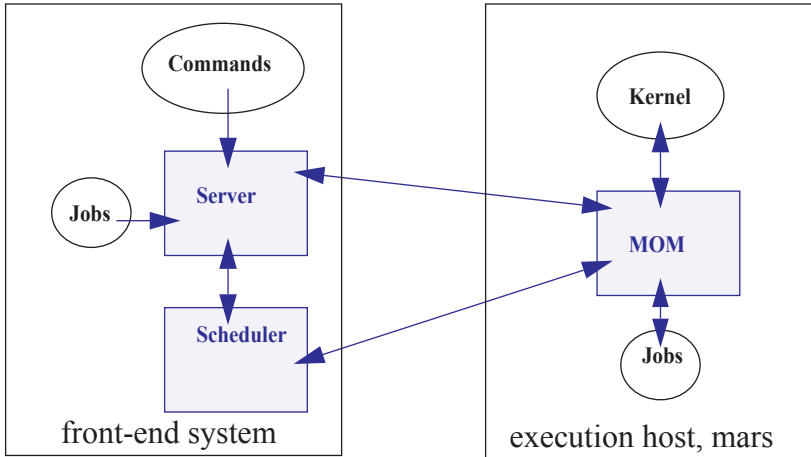
In this case, the PBS `server_priv/nodes` file would contain the name of our execution server mars, but this may not be what was written to the file during installation, depending on which options were selected. It is possible the hostname of the machine on which the Server was installed was added to 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:

```
$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
```

10.3   **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
```
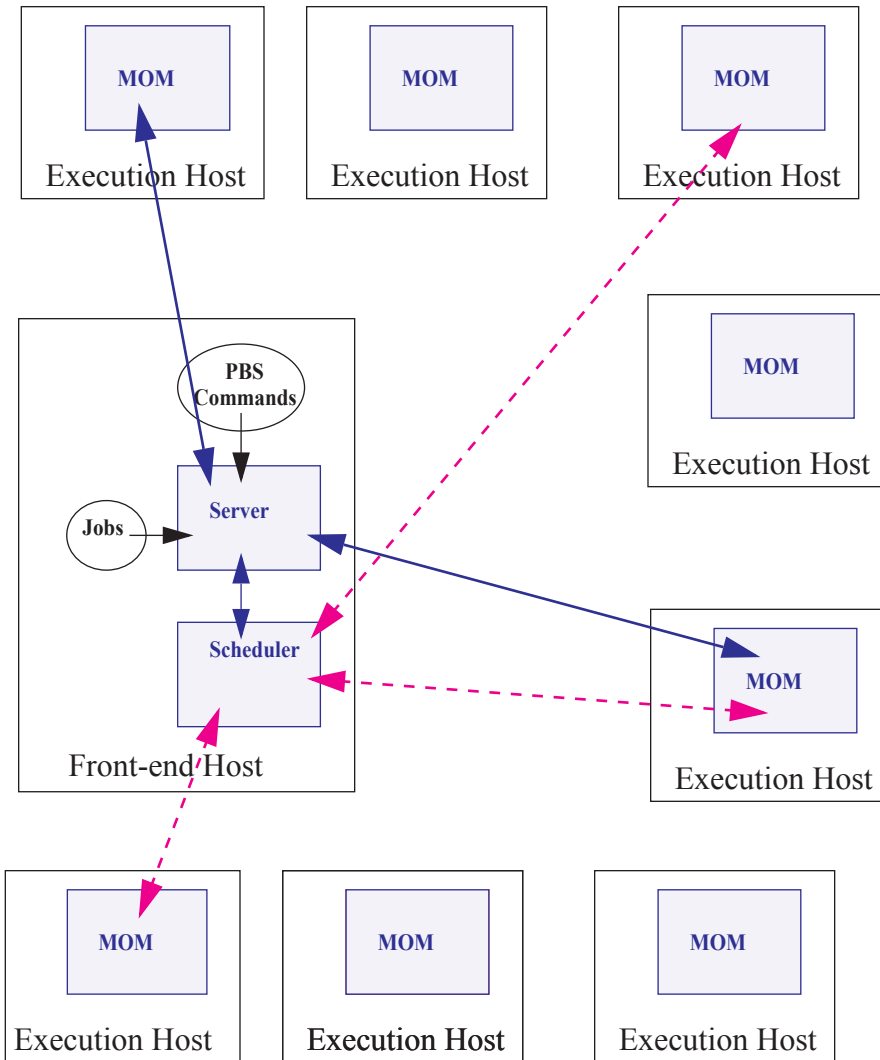
This diagram illustrates a multi-vnode complex configuration wherein the Scheduler and Server communicate with the MOMs on the execution hosts. Jobs are submitted to the Server, scheduled for execution by the Scheduler, and then transferred to a MOM when it's time to be run. MOM periodically sends status information back to the Server, and answers resource requests from the Scheduler.

10.4   **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: 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 = linux
Qmgr: set queue server_1 resources_min.arch = linux
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
```

10.5   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 5.14.7.3.ii, "Example of Floating, Externally-managed License with Features" on page 343 in the PBS Professional Administrator's Guide.

10.6 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: 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
```

Chapter 11

Run Limit Error Messages

This chapter lists the error messages generated when limits are exceeded. See <u>section 5.15.1, "Managing Resource Usage By Users, Groups, and Projects, at Server & Queues" on page 360 in the PBS Professional Administrator's Guide</u>.

11.1   Run Limit Error Messages

When a job would exceed a limit by running, the job's comment field is set to one of the following messages. The following table shows the limit attribute, where the limit is applied, to whom the limit is applied, and the message.

<div align="center">

Table 11-1: Job Run Limit Error Messages

</div>

Attribute	Where Applied	To Whom Applied	Message
max_run	queue	o: PBS_ALL	Not Running: Queue <Q> job limit has been reached.
max_run	server	o: PBS_ALL	Not Running: Server job limit has been reached.
max_run	server	p:PBS_GENERIC	Not Running: Project has reached server running limit.
max_run	queue	p:PBS_GENERIC	Not Running: Project has reached queue<queue-name>'s running limit.

Table 11-1: Job Run Limit Error Messages

Attribute	Where Applied	To Whom Applied	Message
max_run	server	p:<project name>	Not Running: Server job limit reached for project <project name>
max_run	queue	p:<project name>	Not Running: Queue <queue-name> job limit reached for project <project name>
max_run	queue	g: PBS_GENERIC	Not Running: Group has reached queue <Q> running limit.
max_run	server	g: PBS_GENERIC	Not Running: Group has reached server running limit.
max_run	queue	u: PBS_GENERIC	Not Running: User has reached queue <Q> running job limit.
max_run	server	u: PBS_GENERIC	Not Running: User has reached server running job limit.
max_run	queue	g:<group name>	Queue <Q> job limit reached for group <G>
max_run	server	g:<group name>	Server job limit reached for group <G>
max_run	queue	u:<user name>	Queue <Q> job limit reached for user <U>
max_run	server	u:<user name>	Server job limit reached for user <U>
max_run_res	queue	o: PBS_ALL	Queue <Q> job limit reached on resource <res>
max_run_res	server	o: PBS_ALL	Server job limit reached on resource <res>
max_run_res	queue	p:PBS_GENERIC	Not Running: Queue <queue-name> per-project limit reached on resource <res>
max_run_res	server	p:PBS_GENERIC	Not Running: Server per-project limit reached on resource <res>

Table 11-1: Job Run Limit Error Messages

Attribute	Where Applied	To Whom Applied	Message
max_run_res	server	p:<project name>	Not Running: would exceed project <project_name>'s limit on resource <res> in complex
max_run_res	queue	p:<project name>	Not Running: would exceed project <project_name>'s limit on resource <res> in queue <queue-name>
max_run_res	queue	g: PBS_GENERIC	Queue <Q> per-group limit reached on resource <res>
max_run_res	server	g: PBS_GENERIC	Server per-group limit reached on resource <res>
max_run_res	queue	u: PBS_GENERIC	Queue <Q> per-user limit reached on resource <res>
max_run_res	server	u: PBS_GENERIC	Server per-user limit reached on resource <res>
max_run_res	queue	g:<group name>	would exceed group <G>'s limit on resource <res> in queue <Q>
max_run_res	server	g:<group name>	would exceed group <G>'s limit on resource <res> in complex
max_run_res	queue	u:<user name>	would exceed user <U>'s limit on resource <res> in queue <Q>
max_run_res	server	u:<user name>	would exceed user <U>'s limit on resource <res> in complex

Error Codes

The following table lists all the PBS error codes, their textual names, and a description of each.

Table 12-1: Error Codes

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

Error Codes

Table 12-1: Error Codes

Error Name	Error Code	Description
PBSE_SYSTEM	15010	System error occurred
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 attribute 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. Returned under following conditions: 1. User does not have a password entry (getpwnam() returns null). 2. User's UID is zero and root isn't allowed to run jobs (acl_roots). 3. PBS_O_HOST is not set in the job.
PBSE_HOPCOUNT	15024	Max hop count exceeded
PBSE_QUEEXIST	15025	Queue already exists
PBSE_ATTRTYPE	15026	Incompatible queue attribute type

Table 12-1: Error Codes

Error Name	Error Code	Description
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. Message is distorted or truncated.
PBSE_BADATLST	15032	Bad attribute list structure
PBSE_NOCONNECTS	15033	No free connections
PBSE_NOSERVER	15034	No Server to connect to
PBSE_UNKRESC	15035	Unknown resource
PBSE_EXCQRESC	15036	Job exceeds Queue resource limits
PBSE_QUENODFLT	15037	No Default Queue Defined
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

Error Codes

Table 12-1: Error Codes

Error Name	Error Code	Description
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 (**Obsolete**)	15057	Cannot execute there (Obsolete; no longer used.)
PBSE_SISREJECT	15058	Sister rejected
PBSE_SISCOMM	15059	Sister could not communicate
PBSE_SVRDOWN	15060	Request rejected -server shutting down
PBSE_CKPSHORT	15061	Not all tasks could checkpoint
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

Table 12-1: Error Codes

Error Name	Error Code	Description
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
Reserved	15076	Not used.
PBSE_TOOLATE	15077	Reservation submitted with a start time that has already passed
PBSE_genBatchReq	15082	Batch request generation failed
PBSE_mgrBatchReq	15083	`qmgr` batch request failed
PBSE_UNKRESVID	15084	Unknown reservation ID
PBSE_delProgress	15085	Delete already in progress
PBSE_BADTSPEC	15086	Bad time specification(s)
PBSE_RESVMSG	15087	So reply_text can return a msg
PBSE_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

Error Codes

Table 12-1: Error Codes

Error Name	Error Code	Description
	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_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
PBSE_MaxArraySize	15107	max array size exceeded
PBSE_INVALSELECTRESC	15108	resource invalid in select spec
PBSE_INVALJOBRESC	15109	invalid job resource
PBSE_INVALNODEPLACE	15110	node invalid w/place|select
PBSE_PLACENOSELECT	15111	cannot have place w/o select
PBSE_INDIRECTHOP	15112	too many indirect resource levels
PBSE_INDIRECTBT	15113	target resource undefined
PBSE_NGBLUEGENE	15114	No node_group_enable on Blue-Gene
PBSE_NODESTALE	15115	Cannot change state of stale vnode
PBSE_DUPRESC	15116	cannot dup resource within a chunk
PBSE_CONNFULL	15117	server connection table full
PBSE_LICENSE_MIN_BADVAL	15118	bad value for pbs_license_min
PBSE_LICENSE_MAX_BADVAL	15119	bad value for pbs_license_max

Table 12-1: Error Codes

Error Name	Error Code	Description
PBSE_LICENSE_LINGER_BADVAL	15120	bad value for pbs_license_linger_time
PBSE_LICENSE_SERVER_DOWN	15121	License server is down
PBSE_LICENSE_BAD_ACTION	15122	Not allowed action with licensing
PBSE_BAD_FORMULA	15123	invalid sort formula
PBSE_BAD_FORMULA_KW	15124	invalid keyword in formula
PBSE_BAD_FORMULA_TYPE	15125	invalid resource type in formula
PBSE_BAD_RRULE_YEARLY	15126	reservation duration exceeds 1 year
PBSE_BAD_RRULE_MONTHLY	15127	reservation duration exceeds 1 month
PBSE_BAD_RRULE_WEEKLY	15128	reservation duration exceeds 1 week
PBSE_BAD_RRULE_DAILY	15129	reservation duration exceeds 1 day
PBSE_BAD_RRULE_HOURLY	15130	reservation duration exceeds 1 hour
PBSE_BAD_RRULE_MINUTELY	15131	reservation duration exceeds 1 minute
PBSE_BAD_RRULE_SECONDLY	15132	reservation duration exceeds 1 second
PBSE_BAD_RRULE_SYNTAX	15133	invalid recurrence rule syntax
PBSE_BAD_RRULE_SYNTAX2	15134	invalid recurrence rule syntax
PBSE_BAD_ICAL_TZ	15135	Undefined timezone info directory
PBSE_HOOKERROR	15136	error encountered related to hooks
PBSE_NEEDQUET	15137	need queue type set

Error Codes

Table 12-1: Error Codes

Error Name	Error Code	Description
PBSE_ETEERROR	15138	not allowed to alter attribute when eligible_time_enable is off
PBSE_HISTJOBID	15139	History job ID
PBSE_JOBHISTNOTSET	15140	job_history_enable not SET
PBSE_MIXENTLIMS	15141	mixing old and new limit enforcement
	15145	Server host not allowed to be provisioned
	15146	While provisioning, provisioning attributes can't be modified
	15147	State of provisioning vnode can't be changed
	15148	Vnode can't be deleted while provisioning
	15149	Attempt to set an AOE that is not in resources_available.aoe
	15150	Illegal job/reservation submission/ alteration
PBSE_MOM_INCOMPLETE_HOOK	15167	Execution hooks not fully transferred to a particular MoM
PBSE_MOM_REJECT_ROOT_SCRIPT S	15168	A MoM has rejected a request to copy a hook-related file, or a job script to be executed by root
PBSE_HOOK_REJECT	15169	A MoM received a reject result from a mom hook
PBSE_HOOK_REJECT_RERUNJOB	15170	Hook rejection requiring a job to be rerun

Table 12-1: Error Codes

Error Name	Error Code	Description
PBSE_HOOK_REJECT_DELETEJOB	15171	Hook rejection requiring a job to be deleted
Resource monitor specific error codes		
PBSE_RMUNKNOWN	15201	Resource unknown
PBSE_RMBADPARAM	15202	Parameter could not be used
PBSE_RMNOPARAM	15203	A needed parameter did not exist
PBSE_RMEXIST	15204	Something specified didn't exist
PBSE_RMSYSTEM	15205	A system error occurred
PBSE_RMPART	15206	Only part of reservation made
PBSE_SSIGNON_BAD_TRANSITION2	15207	bad attempt: false to true
PBSE_ALPSRELERR	15209	PBS is unable to release the ALPS reservation

Error Codes

Chapter 13

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.

Table 13-1: Request Codes

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

Request Codes

Table 13-1: Request Codes

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

Table 13-1: Request Codes

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
75	PBS_BATCH_ConfirmResv
80	PBS_BATCH_DefSchReply
81	PBS_BATCH_StatusSched
82	PBS_BATCH_StatusRsc

Chapter 14

PBS Environment Variables

14.1 PBS Environment Variables

The following table lists the PBS environment variables:

Table 14-1: PBS Environment Variables

Variable	Meaning
NCPUS	Number of threads, defaulting to number of CPUs, on the vnode
OMP_NUM_THREADS	Same as NCPUS.
PBS_ARRAY_ID	Identifier for job arrays. Consists of sequence number.
PBS_ARRAY_INDEX	Index number of subjob in job array.
PBS_CONF_FILE	Path to `pbs.conf`
PBS_CPUSET_DEDICATED	Set by `mpiexec` to assert exclusive use of resources in assigned cpuset.
PBS_ENVIRONMENT	Indicates job type: **PBS_BATCH** or **PBS_INTERACTIVE**
PBS_JOBCOOKIE	Unique identifier for inter-MOM job-based communication.

Table 14-1: PBS Environment Variables

Variable	Meaning
PBS_JOBDIR	Pathname of job-specific staging and execution directory
PBS_JOBID	The job identifier assigned to the job or job array by the batch system.
PBS_JOBNAME	The job name supplied by the user.
PBS_MOMPORT	Port number on which this job's MOMs will communicate.
PBS_NODEFILE	The filename containing a list of vnodes assigned to the job.
PBS_NODENUM	Logical vnode number of this vnode allocated to the job.
PBS_O_HOME	Value of **HOME** from submission environment.
PBS_O_HOST	The host name on which the qsub command was executed.
PBS_O_LANG	Value of **LANG** from submission environment
PBS_O_LOGNAME	Value of **LOGNAME** from submission environment
PBS_O_MAIL	Value of **MAIL** from submission environment
PBS_O_PATH	Value of **PATH** from submission environment
PBS_O_QUEUE	The original queue name to which the job was submitted.
PBS_O_SHELL	Value of **SHELL** from submission environment
PBS_O_SYSTEM	The operating system name where qsub was executed.
PBS_O_TZ	Value of **TZ** from submission environment
PBS_O_WORKDIR	The absolute path of directory where qsub was executed.

Table 14-1: PBS Environment Variables

Variable	Meaning
PBS_QUEUE	The name of the queue from which the job is executed.
PBS_TASKNUM	The task (process) number for the job on this vnode.
TMPDIR	The job-specific temporary directory for this job.

Chapter 15

File Listing

The following table lists all the PBS files and directories; owner and permissions are specific to UNIX systems.

Table 15-1: File Listing

Directory / File	Owner	Permission	Average Size
`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/pbsdsh`	root	-rwxr-xr-x	111837
`PBS_EXEC/bin/pbsnodes`	root	-rwxr-xr-x	153004
`PBS_EXEC/bin/pbs_dataservice`	root	-rwx------	
`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

Table 15-1: File Listing

Directory / File	Owner	Permission	Average Size
PBS_EXEC/bin/pbs_wish	root	-rwxr-xr-x	1592236
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
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

Table 15-1: File Listing

Directory / File	Owner	Permission	Average Size
`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_habitat`	root	-rwx------	10059
`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
`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/liblog.a`	root	-rw-r--r--	101230

Table 15-1: File Listing

Directory / File	Owner	Permission	Average Size
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.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
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

Table 15-1: File Listing

Directory / File	Owner	Permission	Average Size
`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/` `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/` `Downarrow.bmp`	root	-rw-r--r--	299
`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

Table 15-1: File Listing

Directory / File	Owner	Permission	Average Size
`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_bitmaps/` `Uparrow.bmp`	root	-rw-r--r--	293
`PBS_EXEC/lib/xpbs/pbs_box.tk`	root	-rw-r--r--	25912
`PBS_EXEC/lib/xpbs/` `pbs_button.tk`	root	-rw-r--r--	18795
`PBS_EXEC/lib/xpbs/` `pbs_checkpoint.tk`	root	-rw-r--r--	6892
`PBS_EXEC/lib/xpbs/` `pbs_common.tk`	root	-rw-r--r--	25940
`PBS_EXEC/lib/xpbs/` `pbs_concur.tk`	root	-rw-r--r--	8445
`PBS_EXEC/lib/xpbs/` `pbs_datetime.tk`	root	-rw-r--r--	4533
`PBS_EXEC/lib/xpbs/` `pbs_email_list.tk`	root	-rw-r--r--	3094
`PBS_EXEC/lib/xpbs/pbs_entry.tk`	root	-rw-r--r--	12389
`PBS_EXEC/lib/xpbs/` `pbs_fileselect.tk`	root	-rw-r--r--	7975
`PBS_EXEC/lib/xpbs/pbs_help`	root	drwxr-xr-x	4096
`PBS_EXEC/lib/xpbs/pbs_help/` `after_depend.hlp`	root	-rw-r--r--	1746

Table 15-1: File Listing

Directory / File	Owner	Permission	Average Size
PBS_EXEC/lib/xpbs/pbs_help/ auto_update.hlp	root	-rw-r--r--	776
PBS_EXEC/lib/xpbs/pbs_help/ before_depend.hlp	root	-rw-r--r--	1413
PBS_EXEC/lib/xpbs/pbs_help/ concur.hlp	root	-rw-r--r--	1383
PBS_EXEC/lib/xpbs/pbs_help/ datetime.hlp	root	-rw-r--r--	698
PBS_EXEC/lib/xpbs/pbs_help/ delete.hlp	root	-rw-r--r--	632
PBS_EXEC/lib/xpbs/pbs_help/ email.hlp	root	-rw-r--r--	986
PBS_EXEC/lib/xpbs/pbs_help/ fileselect.hlp	root	-rw-r--r--	1655
PBS_EXEC/lib/xpbs/pbs_help/ hold.hlp	root	-rw-r--r--	538
PBS_EXEC/lib/xpbs/pbs_help/ main.hlp	root	-rw-r--r--	15220
PBS_EXEC/lib/xpbs/pbs_help/ message.hlp	root	-rw-r--r--	677
PBS_EXEC/lib/xpbs/pbs_help/ misc.hlp	root	-rw-r--r--	4194
PBS_EXEC/lib/xpbs/pbs_help/ modify.hlp	root	-rw-r--r--	6034
PBS_EXEC/lib/xpbs/pbs_help/ move.hlp	root	-rw-r--r--	705

Table 15-1: File Listing

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

Table 15-1: File Listing

Directory / File	Owner	Permission	Average Size
`PBS_EXEC/lib/xpbs/pbs_help/` `signal.hlp`	root	-rw-r--r--	675
`PBS_EXEC/lib/xpbs/pbs_help/` `staging.hlp`	root	-rw-r--r--	3702
`PBS_EXEC/lib/xpbs/pbs_help/` `submit.hlp`	root	-rw-r--r--	9721
`PBS_EXEC/lib/xpbs/pbs_help/` `terminate.hlp`	root	-rw-r--r--	635
`PBS_EXEC/lib/xpbs/pbs_help/` `trackjob.hlp`	root	-rw-r--r--	2978
`PBS_EXEC/lib/xpbs/pbs_hold.tk`	root	-rw-r--r--	3539
`PBS_EXEC/lib/xpbs/` `pbs_jobname.tk`	root	-rw-r--r--	3375
`PBS_EXEC/lib/xpbs/` `pbs_listbox.tk`	root	-rw-r--r--	10544
`PBS_EXEC/lib/xpbs/pbs_main.tk`	root	-rw-r--r--	24147
`PBS_EXEC/lib/xpbs/pbs_misc.tk`	root	-rw-r--r--	14526
`PBS_EXEC/lib/xpbs/` `pbs_owners.tk`	root	-rw-r--r--	4509
`PBS_EXEC/lib/xpbs/pbs_pbs.tcl`	root	-rw-r--r--	52524
`PBS_EXEC/lib/xpbs/pbs_pref.tk`	root	-rw-r--r--	3445
`PBS_EXEC/lib/xpbs/` `pbs_preferences.tcl`	root	-rw-r--r--	4323
`PBS_EXEC/lib/xpbs/` `pbs_prefsave.tk`	root	-rw-r--r--	1378

Table 15-1: File Listing

Directory / File	Owner	Permission	Average Size
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

Table 15-1: File Listing

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/hourglass.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/maximize.bmp	root	-rw-r--r--	287
PBS_EXEC/lib/xpbsmon/ pbs_box.tk	root	-rw-r--r--	15607
PBS_EXEC/lib/xpbsmon/ pbs_button.tk	root	-rw-r--r--	7543
PBS_EXEC/lib/xpbsmon/ pbs_cluster.tk	root	-rw-r--r--	44406

File Listing

Table 15-1: File Listing

Directory / File	Owner	Permission	Average Size
PBS_EXEC/lib/xpbsmon/ pbs_color.tk	root	-rw-r--r--	5634
PBS_EXEC/lib/xpbsmon/ pbs_common.tk	root	-rw-r--r--	5716
PBS_EXEC/lib/xpbsmon/ pbs_dialog.tk	root	-rw-r--r--	8398
PBS_EXEC/lib/xpbsmon/ pbs_entry.tk	root	-rw-r--r--	10697
PBS_EXEC/lib/xpbsmon/ pbs_expr.tk	root	-rw-r--r--	6163
PBS_EXEC/lib/xpbsmon/pbs_help	root	drwxr-xr-x	4096
PBS_EXEC/lib/xpbsmon/pbs_help/ auto_update.hlp	root	-rw-r--r--	624
PBS_EXEC/lib/xpbsmon/pbs_help/ main.hlp	root	-rw-r--r--	15718
PBS_EXEC/lib/xpbsmon/pbs_help/ notes.hlp	root	-rw-r--r--	296
PBS_EXEC/lib/xpbsmon/pbs_help/ pref.hlp	root	-rw-r--r--	1712
PBS_EXEC/lib/xpbsmon/pbs_help/ prefQuery.hlp	root	-rw-r--r--	4621
PBS_EXEC/lib/xpbsmon/pbs_help/ prefServer.hlp	root	-rw-r--r--	1409
PBS_EXEC/lib/xpbsmon/ pbs_listbox.tk	root	-rw-r--r--	10640
PBS_EXEC/lib/xpbsmon/ pbs_main.tk	root	-rw-r--r--	6760

Table 15-1: File Listing

Directory / File	Owner	Permission	Average Size
`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/xpb-` `smon.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`	root	-rw-r--r--	3276
`PBS_EXEC/man/man1/pbs.1B`	root	-rw-r--r--	5376
`PBS_EXEC/man/man1/pbsdsh.1B`	root	-rw-r--r--	2978

Table 15-1: File Listing

Directory / File	Owner	Permission	Average Size
PBS_EXEC/man/man1/pbs_rdel.1B	root	-rw-r--r--	2342
PBS_EXEC/man/man1/pbs_rstat.1B	root	-rw-r--r--	2682
PBS_EXEC/man/man1/pbs_rsub.1B	root	-rw-r--r--	9143
PBS_EXEC/man/man1/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

Table 15-1: File Listing

Directory / File	Owner	Permission	Average Size
`PBS_EXEC/man/man3/` `pbs_default.3B`	root	-rw-r--r--	2150
`PBS_EXEC/man/man3/` `pbs_deljob.3B`	root	-rw-r--r--	3081
`PBS_EXEC/man/man3/` `pbs_disconnect.3B`	root	-rw-r--r--	1985
`PBS_EXEC/man/man3/` `pbs_geterrmsg.3B`	root	-rw-r--r--	2473
`PBS_EXEC/man/man3/` `pbs_holdjob.3B`	root	-rw-r--r--	3006
`PBS_EXEC/man/man3/` `pbs_manager.3B`	root	-rw-r--r--	4337
`PBS_EXEC/man/man3/` `pbs_movejob.3B`	root	-rw-r--r--	3220
`PBS_EXEC/man/man3/` `pbs_msgjob.3B`	root	-rw-r--r--	2912
`PBS_EXEC/man/man3/` `pbs_orderjob.3B`	root	-rw-r--r--	2526
`PBS_EXEC/man/man3/` `pbs_rerunjob.3B`	root	-rw-r--r--	2531
`PBS_EXEC/man/man3/` `pbs_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

Table 15-1: File Listing

Directory / File	Owner	Permission	Average Size
`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
`PBS_EXEC/man/man7/` `pbs_node_attributes.7B`	root	-rw-r--r--	7973

Table 15-1: File Listing

Directory / File	Owner	Permission	Average Size
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/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/ 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

Table 15-1: File Listing

Directory / File	Owner	Permission	Average Size
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_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
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/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

Table 15-1: File Listing

Directory / File	Owner	Permission	Average Size
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/pgsql	root	-rwxr-xr-x	
PBS_EXEC/sbin	root	drwxr-xr-x	4096
PBS_EXEC/sbin/pbs-report	root	-rwxr-xr-x	68296
PBS_EXEC/sbin/pbsfs	root	-rwxr-xr-x	663707
PBS_EXEC/sbin/pbs_demux	root	-rwxr-xr-x	38688
PBS_EXEC/sbin/pbs_idled	root	-rwxr-xr-x	99373
PBS_EXEC/sbin/pbs_iff	root	-rwsr-xr-x	133142
PBS_EXEC/sbin/pbs_mom	root	-rwx------	839326
PBS_EXEC/sbin/pbs_mom.cpuset	root	-rwx------	0
PBS_EXEC/sbin/pbs_mom.standard	root	-rwx------	0
PBS_EXEC/sbin/pbs_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/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

Table 15-1: File Listing

Directory / File	Owner	Permission	Average Size
`PBS_EXEC/tcltk/include`	root	drwxr-xr-x	4096
`PBS_EXEC/tcltk/include/tcl.h`	root	-rw-r--r--	57222
`PBS_EXEC/tcltk/include/tclDe-cls.h`	root	-rw-r--r--	123947
`PBS_EXEC/tcltk/include/tk.h`	root	-rw-r--r--	47420
`PBS_EXEC/tcltk/include/tkDe-cls.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/tclCon-fig.sh`	root	-rw-r--r--	7076
`PBS_EXEC/tcltk/lib/tk8.3`	root	drwxr-xr-x	4096
`PBS_EXEC/tcltk/lib/tkConfig.sh`	root	-rw-r--r--	3822
`PBS_EXEC/tcltk/license.terms`	root	-rw-r--r--	2233
`PBS_HOME`	root	drwxr-xr-x	4096
`PBS_HOME/aux`	root	drwxr-xr-x	4096
`PBS_HOME/checkpoint`	root	drwx------	4096

Table 15-1: File Listing

Directory / File	Owner	Permission	Average Size
`PBS_HOME/datastore`	data service account	-rwx------	
`PBS_HOME/mom_logs`	root	drwxr-xr-x	4096
`PBS_HOME/mom_priv`	root	drwxr-x--x	4096
`PBS_HOME/mom_priv/config`	root	-rw-r--r--	18
`PBS_HOME/mom_priv/jobs`	root	drwxr-x--x	4096
`PBS_HOME/mom_priv/mom.lock`	root	-rw-r--r--	4
`PBS_HOME/pbs_environment`	root	-rw-r--r--	0
`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/ resource_group`	root	-rw-r--r--	0
`PBS_HOME/sched_priv/sched.lock`	root	-rw-r--r--	4
`PBS_HOME/sched_priv/ sched_config`	root	-rw-r--r--	6370
`PBS_HOME/sched_priv/sched_out`	root	-rw-r--r--	0
`PBS_HOME/server_logs`	root	drwxr-xr-x	4096
`PBS_HOME/server_priv`	root	drwxr-x---	4096
`PBS_HOME/server_priv/account-ing`	root	drwxr-xr-x	4096

Table 15-1: File Listing

Directory / File	Owner	Permission	Average Size
PBS_HOME/server_priv/ acl_groups	root	drwxr-x---	4096
PBS_HOME/server_priv/acl_hosts	root	drwxr-x---	4096
PBS_HOME/server_priv/acl_svr	root	drwxr-x---	4096
PBS_HOME/server_priv/acl_svr/ managers	root	-rw-------	13
PBS_HOME/server_priv/acl_users	root	drwxr-x---	4096
PBS_HOME/server_priv/jobs	root	drwxr-x---	4096
PBS_HOME/server_priv/ license_file	root	-rw-r--r--	34
PBS_HOME/server_priv/queues/ newqueue	root	-rw-------	303
PBS_HOME/server_priv/queues/ workq	root	-rw-------	303
PBS_HOME/server_priv/resource- def	root		
PBS_HOME/server_priv/ server.lock	root	-rw-------	4
PBS_HOME/server_priv/svrlive	root	-rw-------	
PBS_HOME/server_priv/tracking	root	-rw-------	0
PBS_HOME/spool	root	drwxrwxrwt	4096
PBS_HOME/undelivered	root	drwxrwxrwt	4096

Chapter 16

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 13.4, "Event Logging" on page 857 in the PBS Professional Administrator's Guide.

A few log messages are listed here.

RPP Retries

Logs	Server, scheduler, MOM
Level	0x0002
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)

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

Logs	Mother Superior
Level	0x0100
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

Logs	MOM
Level	Event level 0x0002, 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

Logs	Scheduler
Level	0x0080
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: Message Indicating Whether It Is Prime Time

Logs	Scheduler
Level	0x0100 (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

Logs	Scheduler
Level	0x0080

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.

Resource Permission Flag Error

Logs	Server
Level	0x0080, 0x0100
Form	"It is invalid to set both flags 'r' and 'i'. Flag 'r' will be ignored."
Example	"It is invalid to set both flags 'r' and 'i'. Flag 'r' will be ignored."
Explanation	The "i" and "r" flags are incompatible. The "i" flag takes precedence.

Error During Evaluation of Tunable Formula

Logs	Scheduler
Level	0x0100
Form	1234.mars;Formula evaluation for job had an error. Zero value will be used
Example	1234.mars;Formula evaluation for job had an error. Zero value will be used
Explanation	Tunable formula produced error when evaluated.

Creation of Job-specific Directory

Logs	MOM
Level	0x0008
Form	"created the job directory <jobdir_root><unique_job_dir_name>"
Example	"created the job directory /Users/user1/pbsjobs/345.myhost"
Explanation	PBS created a job-specific execution and staging directory

Failure to Create Job-specific Directory

Logs	MOM
Level	0x0001
Form	"unable to create the job directory <unique_job_dir_name>"
Example	"unable to create the job directory /Users/user1/pbsjobs/345.myhost"
Explanation	The MOM was unable to create the job-specific staging and execution directory

Failure to Validate MOM's $jobdir_root

Logs	MOM
Level	0x0001
Form	"<file>[<linenum>]command "$jobdir_root <full path>" failed, aborting"
Example	"config[3] command "$jobdir_root /foodir" failed, aborting"
Explanation	$jobdir_root exists in the MOM's configuration file, and the MOM was unable to validate $jobdir_root; MOM has aborted

Job Eligible Time

Logs	Server
Level	0x0400
Form	MM/DD/YYYY hh:mm:ss;log event;Server@hostname;Job;jobID;job accrued 23 secs of <previous sample type>, new accrue_type=<next_sample_type>, eligible_time=<current amount of eligible_time>
Example	08/07/2007 13:xx:yy;0040;Server@myhost;Job;163.myhost;job accrued 23 secs of eligible_time, new accrue_type=run_time, eligible_time=00:00:23
Explanation	Previous sample was of eligible time; next sample will be of run_time, job has accrued 23 seconds of eligible_time.

Log Messages

Invalid Syntax for Standing Reservation

Logs	Server
Level	0x0080
Form	pbs_rsub error: Undefined iCalendar syntax
Example	pbs_rsub error: Undefined iCalendar syntax
Explanation	Invalid syntax given to pbs_rsub for recurrence rule for standing reservation

Appendix A: License Agreement

CAUTION!

PRIOR TO INSTALLATION OR USE OF THE SOFTWARE YOU MUST CONSENT TO THE FOLLOWING SOFTWARE LICENSE TERMS AND CONDITIONS BY CLICKING THE "I ACCEPT" BUTTON BELOW. YOUR ACCEPTANCE CREATES A BINDING LEGAL AGREEMENT BETWEEN YOU AND ALTAIR. IF YOU DO NOT HAVE THE AUTHORITY TO BIND YOUR ORGANIZATION TO THESE TERMS AND CONDITIONS, YOU MUST CLICK "I DO NOT ACCEPT" AND THEN HAVE AN AUTHORIZED PARTY IN THE ORGANIZATION THAT YOU REPRESENT ACCEPT THESE TERMS.

IF YOU, OR THE ORGANIZATION THAT YOU REPRESENT, HAS A MASTER SOFTWARE LICENSE AGREEMENT ("MASTER SLA") ON FILE AT THE CORPORATE HEADQUARTERS OF ALTAIR ENGINEERING, INC. ("ALTAIR"), THE MASTER SLA TAKES PRECEDENCE OVER THESE TERMS AND SHALL GOVERN YOUR USE OF THE SOFTWARE.

MODIFICATION(S) OF THESE SOFTWARE LICENSE TERMS IS EXPRESSLY PROHIBITED. ANY ATTEMTED MODIFICATION(S) WILL BE NONBINDING AND OF NO FORCE OR EFFECT UNLESS EXPRESSLY AGREED TO IN WRITING BY AN AUTHORIZED CORPORATE OFFICER OF ALTAIR. ANY DISPUTE RELATING TO THE VALIDITY OF AN ALLEGED MODIFICATION SHALL BE DETERMINED IN ALTAIR'S SOLE DISCRETION.

Altair Engineering, Inc. - Software License Agreement

THIS SOFTWARE LICENSE AGREEMENT, including any Additional Terms (together with the "Agreement"), shall be effective as of the date of YOUR acceptance of these software license terms and conditions (the "Effective Date") and is between ALTAIR ENGINEERING, INC., 1820 E. Big Beaver Road, Troy, MI 48083-2031, USA, a Michigan corporation ("Altair"), and YOU, or the organization on whose behalf you have authority to accept these terms (the "Licensee"). Altair and Licensee, intending to be legally bound, hereby agree as follows:

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:

Additional Terms. Additional Terms are those terms and conditions which are determined by an Altair Subsidiary to meet local market conditions.

Documentation. Documentation provided by Altair or its resellers on any media for use with the Products.

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.

ISV/Independent Software Vendor. A software company providing its products, ("ISV Software") to Altair's Licensees through the Altair License Management System using Altair License Units.

License Log File. A computer file providing usage information on the Software as gathered by the Software.

License Management System. The license management system (LMS) that accompanies the Software and limits its use in accordance with this Agreement, and which includes a License Log File.

License (Network) Server. A network file server that Licensee owns or leases located on Licensee's premises and identified by machine serial number and/or HostID on the Order Form.

License Units. A parameter used by the LMS to determine 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 Products 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.

Products. Products include Altair Software, ISV Software, and/or Suppliers' software; and Documentation related to all of the forgoing.

Proprietary Rights Notices. Patent, copyright, trademark or other proprietary rights notices applied to the Products, packaging or media.

Software. The Altair software identified in the Order Form and any Updates or Maintenance Releases.

Subsidiary. Subsidiary means any partnership, joint venture, corporation or other form of enterprise in which a party possesses, directly or indirectly, an ownership interest of fifty percent (50%) or greater, or managerial or operational control.

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 term of licenses granted under this Agreement. Annual licenses shall have a 12-month term of use unless stated otherwise on the Order Form. Perpetual licenses shall have a term of twenty-five years. Maintenance agreements for perpetual licenses have a 12-month term.

Update. A new version of the Products made generally available by Altair to its Licensees that includes additional features or functionalities but is substantially the same computer code as the existing Products.

2.   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 Products 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 Products on Licensed Workstations in accordance with the LMS 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 iden-

tified on the Order Form and for the term identified on the Order Form; c) make backup copies of the Products, provided that Altair's and its Suppliers' and ISV's Proprietary Rights Notices are reproduced on each such backup copy; d) freely modify and use Templates, and create interfaces to Licensee's proprietary software for internal use only using APIs 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 and its onsite Contractors who have agreed to be bound by the terms of this Agreement. 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 for 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).

3.   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 Products, except as expressly authorized in this Agreement; b) use any back-up copies of the Products 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 or ISV Software for any reason; d) sublicense, sell, lend, assign, rent, distribute, publicly display or publicly perform the Products 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 Products such as through a service bureau, wide area network, Internet location or time-sharing arrangement except as expressly provided in Section 2(b); g) remove any Proprietary Rights Notices from the Products; h) disable or circumvent the LMS provided with the Products; or (i) link any software developed, tested or supported by Licensee or third parties to the Products (except for Licensee's own proprietary software solely for Licensee's internal use).

4.   OWNERSHIP AND CONFIDENTIALITY. Licensee acknowledges that all applicable rights in patents, copyrights, trademarks, service marks, and trade secrets embodied in the Products are owned by Altair and/or its Suppliers or ISVs. Licensee further acknowledges that the Products, and all copies thereof, are and shall remain the sole and exclusive property of Altair and/or its Suppliers and ISVs. This Agreement is a license and not a sale of the Products. Altair retains all rights in the Products not expressly granted to Licensee herein. Licensee acknowledges that the Products are confidential and constitute valuable assets and trade secrets of Altair and/or its Suppliers and ISVs. Licensee agrees to take the same precautions necessary to protect and maintain the confidentiality of the Products as it does to protect its own information of a confidential nature but in any event, no less than a reasonable degree of care, 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 Products. If Licensee is required by any governmental

authority or court of law to disclose Altair's or its ISV's or its Suppliers' 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 3 and Section 4 of this Agreement shall survive termination of this Agreement for any reason. Altair's Suppliers and ISVs, 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 and its resellers providing support and training to licensed end users of the Products shall keep confidential all Licensee information provided to Altair in order that Altair may provide Support and training to Licensee. Licensee information shall be used only for the purpose of assisting Licensee in its use of the licensed Products. Altair agrees to take the same precautions necessary to protect and maintain the confidentiality of the Licensee information as it does to protect its own information of a confidential nature but in any event, no less than a reasonable degree of care, and shall not disclose or make them available to any person or entity except as expressly provided in this Agreement.

5.   MAINTENANCE AND SUPPORT. **Maintenance**. Altair will provide Licensee, at no additional charge for annual licenses and for a maintenance fee for paid-up licenses, with Maintenance Releases and Updates of the Products that are generally released by Altair during the term of the licenses granted under this Agreement, except that this shall not apply to any Term or 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 agrees to install Maintenance Releases and Updates promptly after receipt from Altair. Maintenance Releases and Updates are subject to this Agreement. Altair shall only be obligated to provide support and maintenance for the most current release of the Software and the 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 with 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, unless stated otherwise on the Order Form. **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 not provided by Altair; (b) any version of the Software other than the current version of the Software or the immediately prior release of the Software; (c) problems caused 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.

6.   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. Supplier software included in the Software and ISV Software provided to Licensee shall be warranted as stated by the Supplier or the ISV. Copies of the Suppliers' and ISV's terms and conditions of warranty 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 PRODUCTS WILL MEET LICENSEE'S REQUIREMENTS OR THAT THEIR OPERATION WILL BE UNINTERRUPTED OR ERROR-FREE, OR THAT IT WILL BE COMPATIBLE WITH ANY PARTICULAR HARDWARE OR SOFTWARE. ALTAIR EXCLUDES AND DISCLAIMS 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 PRODUCTS 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 PRODUCTS.

7.   INDEMNITY. Altair will defend and indemnify, 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 which are based on Supplier software or ISV 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 bind-

ing 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 reasonable 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 7, 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 Licensee 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 7 states Altair's entire liability to Licensee in the event a Claim is made. No indemnification is made for Supplier and/or ISV Software.

8.   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 6 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 6 for ninety (90) days after delivery to Licensee. The warranties hereunder are void if the Software has been improperly installed, misused, 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), 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, except for claims under Section 7. ALTAIR AND ITS SUPPLIERS AND ISVS 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 OR ITS ISVS 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 and ISVs 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.

9. UNITED STATES GOVERNMENT RESTRICTED RIGHTS. This section applies to all acquisitions of the Products by or for the United States government. By accepting delivery of the Products except as provided below, the government or the party procuring the Products under government funding, hereby agrees that the Products qualify 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 Products 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 Products for a full refund.

For procurements governed by DFARS Part 227.72 (OCT 1998), the Software, except as described below, 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.

10. 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.

11.   [RESERVED]

12.   Notice. All notices given by one party to the other under the Agreement or these Additional Terms 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 upon actual receipt.

Written notice shall be made to:

Altair: Licensee Name & Address:

Altair Engineering, Inc._____

1820 E. Big Beaver Rd_____

Troy, MI 48083_____

Attn: Tom M. PerringAttn: _____

13.   TERM. For annual licenses, or Support provided for perpetual licenses, renewal shall be automatic for each 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 licenses procured by Licensee may be made coterminous at the written request of Licensee and the consent of Altair.

14.   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.

15.   GENERAL PROVISIONS. Export Controls. Licensee acknowledges that the Products may be subject to the export control laws and regulations of the United States and other countries, and any amendments thereof. Licensee agrees that Licensee will not directly or indirectly export the Products into any country or use the Products in any manner except in compliance with all applicable U.S. and other countries 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 upon actual receipt. **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 Licensee Purchase Order or comparable document, the terms of this Agreement shall prevail. Moreover, each party agrees any additional terms on any Purchase Order or comparable document 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. In the event of a conflict between the terms of this Agreement, and the Additional Terms, the Agreement shall take precedence. **Entire Agreement**. This Agreement, the Additional Terms, and the Order Form(s) attached hereto 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. **Execution**. Copies of this Agreement executed via original signatures, facsimile or email shall be deemed binding on the parties.

Index

Index

Active Directory 2
Admin 2
administrator 2
Administrators 2
advance reservation 2, 424, 425, 425, 429, 453
aggressive_provision 293
AOE 3
aoe 300
API 3
application checkpoint 3
application operating environment 3
arch 300
array job 3, 10
attribute
    definition 3
    rerunnable 18
avoid_provision 293

B

backfill 282
backfill_prime 282
Backfilling 3
batch job 10
batch processing 3
borrowing vnode 3
built-in resource 3
busy 416
by_queue 283

C

checkpoint 269, 425, 451, 472, 474, 486
    restart 18
    restart file 19
    restart script 19
checkpoint and abort 4
checkpoint and restart 4
checkpoint/restart 4
checkpoint_abort 4, 269
chunk 4
cluster 4

commands 4
complex 4
configuration file
    Version 1 23
    Version 2 23
consumable resource 4
CPU 5
cpus_per_ssinode 283
cpuset_error_action 270
cput 300
custom resource 5

D

dedicated_prefix 283
degraded reservation 18
delegation 5
destination
    definition 5
    identifier 5
directive 6
Domain Admin Account 6
Domain Admins 6
Domain User Account 6
Domain Users 6
down 417

E

eligible_time 426, 431
eligible_time_enable 318
Enterprise Admins 6
entity 6
entity share 6
Environment Variables 463
error codes 449
est_start_time_freq 318
estimated 394
exec_host 425
exec_vnode 300
executable 380
execution host 6
execution queue 7

Index

Index

Index

Index

Index

Index