Altair PBS Professional™ 19.2.3

Reference Guide

Altair | PBS Works

You are reading the Altair PBS Professional 19.2.3

Reference Guide (RG)

Updated 7/31/19

Contact Us

For the most recent information, go to the PBS Works website, www.pbsworks.com, select "My PBS", and log in with your site ID and password.

Altair

Altair Engineering, Inc., 1820 E. Big Beaver Road, Troy, MI 48083-2031 USA www.pbsworks.com

Sales

pbssales@altair.com 248.614.2400

Please send any questions or suggestions for improvements to agu@altair.com.

Technical Support

Need technical support? We are available from 8am to 5pm local times:

Location	Telephone	e-mail
Australia	+1 800 174 396	anz-pbssupport@india.altair.com
China	+86 (0)21 6117 1666	pbs@altair.com.cn
France	+33 (0)1 4133 0992	pbssupport@europe.altair.com
Germany	+49 (0)7031 6208 22	pbssupport@europe.altair.com
India	+91 80 66 29 4500 +1 800 425 0234 (Toll Free)	pbs-support@india.altair.com
Italy	+39 800 905595	pbssupport@europe.altair.com
Japan	+81 3 6225 5821	pbs@altairjp.co.jp
Korea	+82 70 4050 9200	support@altair.co.kr
Malaysia	+91 80 66 29 4500 +1 800 425 0234 (Toll Free)	pbs-support@india.altair.com
North America	+1 248 614 2425	pbssupport@altair.com
Russia	+49 7031 6208 22	pbssupport@europe.altair.com
Scandinavia	+46 (0)46 460 2828	pbssupport@europe.altair.com
Singapore	+91 80 66 29 4500 +1 800 425 0234 (Toll Free)	pbs-support@india.altair.com
South Africa	+27 21 831 1500	pbssupport@europe.altair.com
South America	+55 11 3884 0414	br_support@altair.com
UK	+44 (0)1926 468 600	pbssupport@europe.altair.com

Contents

About PBS Documentation

The PBS Professional guides and release notes apply to the *commercial* releases of PBS Professional.

Document Conventions

`Abbreviation`

The shortest acceptable abbreviation of a command or subcommand is underlined

Attribute

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

`Command`

Commands such as `qmgr` and `scp`

Definition

Terms being defined

`File name`

File and path names

`Input`

Command-line instructions

Method

Method or member of a class

`Output`

Output, example code, or file contents

Syntax

Syntax, template, synopsis

`Utility`

Name of utility, such as a program

Value

Keywords, instances, states, values, labels

Notation

Optional arguments are enclosed in square brackets. For example:

```
qstat [-E]
```

Variables are enclosed in angle brackets. A variable is something the user must fill in with the correct value. In the following example, the user replaces *vnode name* with the name of the vnode:

```
pbsnodes -v <vnode name>
```

Optional variables are enclosed in angle brackets inside square brackets. For example:

```
qstat [<job ID>]
```

Literal terms appear exactly as they should be used. For example, to get the version of the `qstat` command, type the following exactly:

```
qstat --version
```

List of PBS Professional Documentation

PBS Professional Release Notes

Supported platforms, what's new and/or unexpected in this release, deprecations and interface changes, open and closed bugs, late-breaking information. For administrators and job submitters.

PBS Professional Installation & Upgrade Guide:

How to install and upgrade PBS Professional. For the administrator.

PBS Professional Administrator s Guide:

How to configure and manage PBS Professional. For the PBS administrator.

PBS Professional Hooks Guide:

How to write and use hooks for PBS Professional. For the PBS administrator.

PBS Professional Reference Guide:

Covers PBS reference material.

PBS Professional User s Guide:

How to submit, monitor, track, delete, and manipulate jobs. For the job submitter.

PBS Professional Programmer s Guide:

Discusses the PBS application programming interface (API). For integrators.

PBS Professional Manual Pages:

PBS commands, resources, attributes, APIs.

Where to Keep the Documentation

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

Ordering Software and Licenses

To purchase software packages or additional software licenses, contact your Altair sales representative at pbssales@altair.com.

1
Glossary of Terms

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 "ACLs" on page 361 in the PBS Professional Administrator's Guide.

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.

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 a 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

Same as PBS Administrator.

Linux: person with Manager privilege and root access.

Windows: person with Manager privilege who is a member of the local Administrators group.

A person who administers PBS, performing functions such as downloading, installing, upgrading, configuring, or managing PBS.

Administrator is distinguished from “site administrator”, although often these are the same person.

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 hook

A hook that is supplied as part of PBS. These hooks cannot be created or deleted by administrators. See "Managing Built-in Hooks" on page 145 in the PBS Professional Hooks Guide.

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 "Built-in vs. Custom Resources" on page 231 in the PBS Professional Administrator's Guide.

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.

Chunk-level resource, host-level resource

A resource that is available at the host level, for example, CPUs or memory. Chunk resources are requested inside of a selection statement. The resources of a chunk are to be applied to the portion of the job running in that chunk.

Chunk resources are requested inside a select statement. A single chunk is requested using this form:

-l select=<resource name>=<value>:<resource name>=<value>

For example, one chunk might have 2 CPUs and 4GB of memory:

```
-l select=ncpus=2:mem=4gb
```

To request multiples of a chunk, prefix the chunk specification by the number of chunks:

-l select=[number of chunks]<chunk specification>

For example, to request six of the previous chunk:

```
-l select=6:ncpus=2:mem=4gb
```

To request different chunks, concatenate the chunks using the plus sign ("+"):

-l select=[number of chunks]<chunk specification>+[number of chunks]<chunk specification>

For example, to request two kinds of chunks, one with 2 CPUs per chunk, and one with 8 CPUs per chunk, both kinds with 4GB of memory:

```
-l select=6:ncpus=2:mem=4gb+3:ncpus=8:mem=4GB
```

Chunk set

An identical set of chunks requested in a select statement. The following is a chunk set: 4:ncpus=8:mem=4GB

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.

Communication daemon, comm

The daemon which handles communication between the server, scheduler, and MoMs. Executable is `pbs_comm`.

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 "Consumable vs. Non-consumable Resources" on page 232 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.

Creating a hook

When you "create a hook" using `qmgr`, you're telling PBS that you want it to make you an empty hook object that has no characteristics other than a name.

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 "Built-in vs. Custom Resources" on page 231 in the PBS Professional Administrator's Guide.

Data service account

Created by PBS on installation. Account that is internal to the data service, with its own data service password. Used by PBS to log into and do operations on the data service. PBS maps this account to the PBS data service management account. Must have same name as PBS data service management account.

Data service management account

Created by administrator. Linux or Windows account with a system pasword. Data service account maps to the PBS data service management account and both must have the same name.

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 identifier, destination queue, destination server

String. One or more queues or a server. Jobs may be queried at or sent to a destination queue (for an example, see "qmove" on page 167). Commands may be directed to a destination queue or server (for an example, see "qenable" on page 142). A destination may be at the default PBS server or at another server.

Destination queue format:

<queue name>

Indicates specified queue at default server.

@<server name>

When moving a job, indicates default queue at that server.

When operating on queues, can indicate all queues at that server.

<queue name>@<server name>

Indicates specified queue at specified server.

Destination server format:

(no server name)

Indicates default server.

@<server name>

Indicates specified server.

@default

Indicates default 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 "Changing the Directive Prefix", on page 16 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.

Endpoint

A PBS server, scheduler, or MoM daemon.

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.

Event

A PBS operation or state transition. Also called *action*. For a list of events, see "Event Types" on page 82 in the PBS Professional Hooks Guide.

Execution event hook

A hook that runs at an execution host. These hooks run after a job is received by MoM. Execution event hooks have names prefixed with "*execjob_*".

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 HPE SGI 8600 are also collections of hosts.

An execution host can be comprised of one or more vnodes. On the HPE SGI 8600, 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.

Failure action

The action taken when a hook fails to execute. Specified in the fail_action hook attribute. See "Using the fail_action Hook Attribute" on page 34 in the PBS Professional Hooks Guide.

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 "Using Fairshare" on page 139 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/complex

In peer scheduling, the queue/complex from which jobs are pulled to be run at another queue/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 "Global vs. Local Resources" on page 233 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 2.47, "qhold", on page 144.

HTT

Intel's Hyper-Threading Technology

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

Idle

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

Importing a hook

When you "import a hook" using `qmgr`, you're telling PBS which Python script to run when the hook is triggered.

Importing a hook configuration file

When you "import a hook configuration file" using `qmgr`, you're telling PBS which file should be stored as the configuration file for the specified hook.

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 "Job Arrays", on page 147 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 351.

Job Submission Description Language (JSDL)

Language for describing the resource requirements of jobs.

Job-wide resource, server resource, queue resource

A job-wide resource, also called a server-level or queue-level resource, is a resource that is available to the entire job at the server or queue.

A job-wide resource is available to be consumed or matched at the server or queue if you set the server or queue 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.

Examples of job-wide resources are shared scratch space, licenses, or walltime.

A job can request a job-wide resource for the entire job, but not for individual chunks. 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.

Leaf

An endpoint (a server, scheduler, or MoM daemon.)

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 number>@<hostname>" 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 number of jobs that can be queued
- The maximum number of jobs that can be running
- The maximum number of jobs that can be queued and running
- The maximum amount of a resource that can be allocated to queued jobs
- 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 and running jobs

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 a scheduler. See "Global vs. Local Resources" on page 233 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 "Manager" on page 359 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).

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 Linux, if the user's login shell is `csh`, 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 "Consumable vs. Non-consumable Resources" on page 232 in the PBS Professional Administrator's Guide.

Non-job event hook

A hook that is not directly related to a specific job. Non-job event hooks are periodic hooks, startup hooks, provisioning hooks, and reservation creation hooks.

nppcu

Number of Processors Per Compute Unit - Cray BASIL 1.3 attribute in the RESERVE XML for specifying how many processors of a compute unit should be used.

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 "Operator" on page 358 in the PBS Professional Administrator's Guide.

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.

pbshook

Keyword used by `qmgr` to operate on built-in hooks.

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

PBS Administrator

Same as Administrator.

Linux: person with Manager privilege and root access.

Windows: person with Manager privilege who is a member of the local Administrators group.

A person who administers PBS, performing functions such as downloading, installing, upgrading, configuring, or managing PBS.

PBS Administrator is distinguished from "site administrator", although often these are the same person.

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.

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 "Placement Sets" on page 171 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 "Placement Sets" on page 171 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 "Placement Sets" on page 171 in the PBS Professional Administrator's Guide.

Policy, Scheduling policy

The set of rules by which a 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.

Preemption target

A preemption target is a job in a specified queue or a job that has requested a specified resource. The queue and/or resource is specified in another job's Resource_List.preempt_targets.

Pre-execution event hook

A hook that runs before the job is accepted by MoM. These hooks do not run on execution hosts. Pre-execution event hooks are for job submission, moving a job, altering a job, or just before sending a job to an execution host.

Primary Scheduler

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

Primary Execution Host

The execution host where a job's top task runs, and where the MoM that manages the job runs.

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. HPE Performance Cluster Manager (HPCM).

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.

Reservation ID, reservation identifier

When a reservation is successfully submitted to PBS, PBS returns a unique identifier for the reservation. Format:

R<sequence number>[.server][@new server]

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 "Using PBS Resources" on page 227 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

A scheduler is a daemon which implements some or all of the site's job scheduling policy controlling when and where each job is run. A scheduler is a 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 ID, job array ID, or reservation ID, for example, *1234*. The largest value that can be used for a sequence number is set in the max_job_sequence_id job attribute.

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.

Site

A location which for our purposes uses (or will use) PBS. A site can employ one or more PBS complexes, each made up of any combination of hardware and software PBS supports.

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, 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 "Staging and Execution Directories for Job" on page 519 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 351.

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, 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.

TPP

TCP-based Packet Protocol. Protocol used by `pbs_comm`.

User

Has two meanings:

1. A person who submits jobs to PBS, as differentiated from Operators, Managers and administrators. See "User" on page 358 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 233.

Version 2 configuration file

Also called vnodedefs 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.

vnodedefs file

A Version 2 configuration file. Vnode configuration file containing vnode attribute settings. Created using `pbs_mom -s insert` command.

vp

Virtual processor. The smallest unit of execution resources that can be specified to run a job. Cray refers to these as a CPU (aka a BASIL PE, an Intel thread or an AMD core).

2 PBS Commands

In this chapter, we describe each PBS command, including any options, operands, etc.

2.1 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.

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	Action	Permission Required	Server Must Be Running?	MoM Must Be Running?
mpiexec	Runs MPI programs under PBS on Linux	Any	No	No
nqs2pbs	**Deprecated**. Converts NQS job scripts to PBS format.	Any	No	No
pbs	Start, stop, restart, or get the PIDs of PBS daemons	Root on Linux; Admin on Windows	No	No
pbsdsh	Distributes tasks to vnodes under PBS	Any	No	Yes
pbsfs	Show or manipulate PBS fairshare usage data	Any	Yes	No
pbsnodes	Query PBS host or vnode status, mark hosts free or offline, change the comment for a host, or output vnode information	Result depends on permission	Yes	No
pbsrun	General-purpose wrapper script for `mpirun`	Root or PBS administrator only	No	No
pbsrun_unwrap	Unwraps `mpirun`, reversing `pbsrun_wrap`	Root only	No	No
pbsrun_wrap	General-purpose script for wrapping `mpirun` in `pbsrun`	Root only	No	No
pbs_account	For Windows. Manage PBS service account	Admin on Windows	No	No

Table 2-1: Permission and Daemon Requirements for Commands

Command	Action	Permission Required	Server Must Be Running?	MoM Must Be Running?
pbs_attach	Attaches a session ID to a PBS job	Any	Yes	Yes
pbs_comm	Starts the PBS communication daemon	Root on Linux; Admin on Windows	No	No
pbs_dataservice	Start, stop, or check the status of PBS data service	Root on Linux; Admin on Windows	No	No
pbs_ds_password	Sets or changes data service user account or its password	Root on Linux; Admin on Windows	No	No
pbs_hostn	Reports hostname and network address(es)	Any	No	No
pbs_idled	Runs PBS daemon that monitors the console and informs pbs_mom of idle time	Root or PBS administrator only	No	No
pbs_iff	Tests authentication with the server	Any; useful only to root	Yes	No
pbs_interactive	For Windows. Register, unregister, or get the version of PBS_INTERACTIVE service	Administrator only	No	No
pbs_lamboot	PBS front end to LAM's lamboot program	Any	No	No
pbs_migrate_users	Transfers per-user or per-server passwords between PBS servers during a migration upgrade	Any	Yes	No
pbs_mkdirs	For Windows. Create, or fix the permissions of, the directories and files used by PBS	PBS administrator only	No	No
pbs_mom	Runs the PBS job monitoring and execution daemon	Root on Linux; Admin on Windows	No	No
pbs_mpihp	Runs an MPI application in a PBS job with HP MPI	Any	Yes	Yes
pbs_mpilam	Runs MPI programs under PBS with LAM MPI	Any	Yes	Yes
pbs_mpirun	Runs MPI programs under PBS with MPICH	Any	Yes	Yes
pbs_password	Sets or updates password of a PBS user	Any	Yes	No
pbs_probe	**Deprecated**. Reports PBS diagnostic information and fixes permission errors	Root or PBS administrator only	No	No

Table 2-1: Permission and Daemon Requirements for Commands

Command	Action	Permission Required	Server Must Be Running?	MoM Must Be Running?
pbs_python	Python interpreter for debugging a hook script from the command line	Any	No	No
pbs_ralter	Modify an existing advance or standing reservation	Job owner or PBS administrator	Yes	No
pbs_rdel	Deletes a PBS advance or standing reservation	Any	Yes	No
pbs_release_nodes	Releases vnodes assigned to a PBS job	Job owner, PBS Manager, Operator, administrator, root on Linux, Admin on Windows	Yes	Yes
pbs_rstat	Shows status of PBS advance or standing reservations	Any	Yes	No
pbs_rsub	Creates a PBS advance or standing reservation	Any	Yes	No
pbs_sched	Runs a PBS scheduler	Root on Linux; Admin on Windows	No	No
pbs_server	Starts a PBS batch server	Root on Linux; Admin on Windows	No	No
pbs_snapshot	Captures PBS data to be used for diagnostics	Root on Linux	Yes	No
pbs_tclsh	TCL shell with TCL-wrapped PBS API	Any	No	No
pbs_tmrsh	TM-enabled replacement for `rsh/ssh` for use by MPI implementations	Any	No	Yes
pbs_topologyinfo	Reports topological information used for licensing purposes	Root or Windows administrator only	No	No
pbs_wish	TK window shell with TCL-wrapped PBS API	Any	No	No
printjob	Prints job information	Root or Windows Administrator only	No	No
qalter	Alters a PBS job	Any	Yes	No
qdel	Deletes PBS jobs	Any	Yes	No
qdisable	Prevents a queue from accepting jobs	Manager or Operator only	Yes	No
qenable	Allows a queue to accept jobs	Manager or Operator only	Yes	No

Table 2-1: Permission and Daemon Requirements for Commands

Command	Action	Permission Required	Server Must Be Running?	MoM Must Be Running?
qhold	Holds PBS batch jobs	Some holds can be set by Operator, Manager, root, or administrator only	Yes	No
qmgr	Administrator's command interface for managing PBS	Any	Yes	No
qmove	Moves a PBS job from one queue to another	Any; managers and operators can move jobs in some cases where unprivileged users cannot	Yes	No
qmsg	Writes message string into one or more job output files	Any	Yes	No
qorder	Swaps queue positions of two PBS jobs	Any	Yes	No
qrerun	Requeues a PBS job	Manager or Operator only	Yes	No
qrls	Releases holds on PBS jobs	Some holds can be released by Operator, Manager, root, or administrator only	Yes	No
qrun	Runs a PBS job immediately	Operator or Manager only	Yes	No
qselect	Selects specified PBS jobs	Any	Yes	No
qsig	Send signal to PBS job	Operator or Manager required to send *admin-suspend*, *admin-resume*, *suspend*, and *resume*. Any privilege for other signals.	Yes	Yes
qstart	Turns on scheduling or routing for the jobs in a PBS queue	Operator or Manager only	Yes	No
qstat	Displays status of PBS jobs, queues, or servers	Result depends on permission	Yes	No
qstop	Prevents PBS jobs in the specified queue from being scheduled or routed	Operator or Manager only	Yes	No
qsub	Submits a job to PBS	Any	Yes	No

Table 2-1: Permission and Daemon Requirements for Commands

Command	Action	Permission Required	Server Must Be Running?	MoM Must Be Running?
qterm	Terminates one or both PBS servers, and optionally terminates scheduler and/or MoMs	Operator or Manager only	Yes	No
tracejob	Extracts and prints log messages for a PBS job	Root or PBS administrator only	No	No
win_postinstall.py	For Windows. Configures PBS services	Administrator	No	No

2.1.1 Windows Requirements

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

2.2 mpiexec

Runs MPI programs under PBS on Linux

2.2.1 Synopsis

mpiexec

mpiexec --version

2.2.2 Description

The PBS `mpiexec` command provides the standard `mpiexec` interface on a system running supported versions of HPE MPI. If executed on a different 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 HPE `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 run as they would normally. MPI jobs can be launched across multiple HPE SGI systems. PBS will manage, track, and cleanly terminate multi-host MPI jobs. PBS users can run MPI jobs within specific partitions.

If CSA has been configured and enabled, PBS will collect accounting information on all tasks launched by an MPI job. CSA information will be associated with the PBS job ID that invoked it, on each execution host.

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

2.2.3 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`

2.2.4 Options

--version

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

2.2.5 Requirements

- System running a supported version of HPE MPI.
- PBS uses HPE SGI's `mpirun(1)` command to launch MPI jobs. HPE SGI's `mpirun` must be in the standard location.
- The location of `pbs_attach()` on each vnode of a multi-vnode MPI job must be the same as it is on the mother superior vnode.
- In order to run multihost jobs, the HPE SGI Array Services must be correctly configured. HPE SGI systems communicating via HPE SGI's Array Services must all use the same version of the `sgi-arraysvcs` package. HPE SGI systems communicating via HPE SGI's Array Services must have been configured to interoperate with each other using the default array. See HPE SGI's `array_services(5)` man page.

2.2.6 Environment Variables

PBS_CPUSET_DEDICATED

The PBS `mpiexec` script sets the PBS_CPUSET_DEDICATED environment variable to assert exclusive use of the resources in the assigned cpuset.

PBS_ENVIRONMENT

The PBS_ENVIRONMENT environment variable is used to determine whether `mpiexec` is being called from within a PBS job.

PBS_MPI_DEBUG

The PBS `mpiexec` checks the PBS_MPI_DEBUG environment variable. If this variable has a nonzero length, debugging information is written.

PBS_MPI_SGIARRAY

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`.

PBS_O_PATH

The PBS `mpiexec` uses the value of PBS_O_PATH to search for the correct `mpiexec` if it was invoked by mistake.

2.2.7 Path

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

2.2.8 See Also

The PBS Professional Administrator's Guide, "pbs_attach" on page 55

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

2.3 nqs2pbs

Deprecated. Converts NQS job scripts to PBS format

2.3.1 Synopsis

nqs2pbs <NQS script> [<PBS script>]

nqs2pbs --version

2.3.2 Description

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

2.3.2.1 Specifying Time or Date

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.

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.

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

2.3.3 Options

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

2.3.4 Operands

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

PBS script
: If specified, this is the name of the new PBS script. If not specified, the new file name is `nqs_script.new`.

2.3.5 Errors

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

2.3.6 See Also

"qsub" on page 207

2.4 pbs

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

2.4.1 Synopsis

pbs [start | stop | restart | status]

2.4.2 Description

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

You can start, stop, restart, or status the PBS daemons using the `systemctl` command; see "Starting & Stopping PBS" on page 115 in the PBS Professional Installation & Upgrade Guide.

2.4.2.1 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`.

This command is typically placed in `/etc/init.d` so that PBS starts up automatically.

2.4.2.2 Required Privilege

Root privilege is required to use this command.

2.4.3 Arguments

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.

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.

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

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

2.4.4 See Also

The PBS Professional Administrator's Guide, "pbs_comm" on page 57, "pbs_mom" on page 71, "pbs_sched" on page 103, "pbs_server" on page 105

2.5 pbsdsh

Distributes tasks to vnodes under PBS

2.5.1 Synopsis

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

pbsdsh [-n <vnode index>] [-s] [-v] [-o] -- <program >[<program args>]

pbsdsh --version

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

2.5.2.1 Example

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
```

2.5.3 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 wraps around, running multiple instances on some vnodes. This option is mutually exclusive with `-n`.

-n <vnode index>

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

-o

No obit request is made for spawned tasks. The program does 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

2.5.4 Operands

program

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

program args

Additional operands, *program args*, are passed as arguments to the program.

2.5.5 Standard Error

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

2.5.6 See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide, "qsub" on page 207, "TM Library Routines" on page 69 in the PBS Professional Programmer's Guide

2.6 pbsfs

Show or manipulate PBS fairshare usage data

2.6.1 Synopsis

Showing usage data:

pbsfs [-c <entity1> <entity2>] [-g <entity>] [-I <scheduler name>] [-p] [-t]

Manipulating usage data:

pbsfs [-d] [-e] [-I <scheduler name>] [-s <entity> <usage value>]

Printing version:

pbsfs --version

2.6.2 Description

You can use the `pbsfs` command to print or manipulate a PBS scheduler's fairshare usage data. You can print the usage data in various formats, described below. Changes made using `pbsfs` take effect in the next scheduling cycle; you do not need to restart or HUP a scheduler for changes to take effect.

We recommend that if you use the options that manipulate usage data, you should do this when a scheduler is not scheduling jobs, because scheduling while changing fairshare usage data may give unwanted results.

2.6.2.1 Prerequisites

The server must be running in order to use the `pbsfs` command.

2.6.2.2 Permissions

You must be root to run the `pbsfs` command; if not, it will print the error message, "`Unable to access fairshare data`".

2.6.3 Options to `pbsfs`

You can safely use the following options while jobs are being scheduled:

(no options)

Same as `pbsfs -p`.

-c <entity1> <entity2>

Compares two fairshare entities.

-g <entity>

Prints a detailed listing for the specified entity, including the path from the root of the tree to the entity.

-I <scheduler name>

Specifies name of scheduler whose data is to be manipulated or shown. Required for multischeds; optional for default scheduler. Name of default scheduler is "default". If not specified, `pbsfs` operates on default scheduler.

-p

Prints the fairshare tree as a table, showing for each internal and leaf vertex the group ID of the vertex's parent, group ID of the vertex, vertex shares, vertex usage, and percent of shares allotted to the vertex.

-t

Prints the fairshare tree in a hierarchical format.

--version

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

It is not recommended to be scheduling jobs when you use the following options:

-d

Decays the fairshare tree by the amount specified in the fairshare_decay_factor scheduler parameter.

-e

Trims fairshare tree to just the entities in the resource_group file. Unknown entities and their usage are deleted; as a result the unknown group has no usage and no children.

-s <entity> <usage value>

Sets *entity*'s usage value to *usage value*. Editing a non-leaf entity is ignored. All non-leaf entity usage values are calculated each time you use the pbsfs command to make changes.

2.6.3.1 Output Formats for pbsfs

The pbsfs command can print output in three different formats:

pbsfs -g <entity>

Prints a detailed listing for the specified entity. Example:

```
pbsfs –g pbsuser3
fairshare entity: pbsuser3
Resgroup: 20
cresgroup: 22
Shares: 40
Percentage: 24.000000%
fairshare_tree_usage: 0.832973
usage: 1000 (cput)
usage/perc: 4167
Path from root:
TREEROOT  :     0       1201 / 1.000 = 1201
group2    :    20       1001 / 0.600 = 1668
pbsuser3  :    22       1000 / 0.240 = 4167
```

pbsfs,

pbsfs -p

Prints the entire tree as a table, with data in columns. Example:

```
pbsfs
Fairshare usage units are in: cput
TREEROOT  : Grp: -1     cgrp: 0    Shares: -1     Usage: 1201   Perc: 100.000%
group2    : Grp: 0      cgrp: 20   Shares: 60     Usage: 1001   Perc: 60.000%
pbsuser3  : Grp: 20     cgrp: 22   Shares: 40     Usage: 1000   Perc: 24.000%
pbsuser2  : Grp: 20     cgrp: 21   Shares: 60     Usage: 1      Perc: 36.000%
group1    : Grp: 0      cgrp: 10   Shares: 40     Usage: 201    Perc: 40.000%
pbsuser1  : Grp: 10     cgrp: 12   Shares: 50     Usage: 100    Perc: 20.000%
pbsuser   : Grp: 10     cgrp: 11   Shares: 50     Usage: 100    Perc: 20.000%
unknown   : Grp: 0      cgrp: 1    Shares: 0      Usage: 1      Perc: 0.000%
```

pbsfs -t

Prints the entire tree as a tree, showing group-child relationships. Example:

```
pbsfs -t
  TREEROOT(0)
      group2(20)
           pbsuser3(22)
           pbsuser2(21)
      group1(10)
           pbsuser1(12)
           pbsuser(11)
      unknown(1)
```

2.6.3.2 Data Output by pbsfs

cresgroup, cgrp

Group ID of the entity

fairshare entity

The specified fairshare tree entity

fairshare usage units

The resource for which a scheduler accumulates usage for fairshare calculations. This defaults to *cput* (CPU seconds) but can be set in a scheduler's configuration file.

fairshare_tree_usage

The entity's effective usage. See "Computing Effective Usage (fairshare_tree_usage)" on page 145 in the PBS Professional Administrator's Guide.

Path from root

The path from the root of the tree to the entity. A scheduler follows this path when comparing priority between two entities.

Percentage, perc

The percentage of the shares in the tree allotted to the entity, computed as fairshare_perc. See "Computing Target Usage for Each Vertex (fairshare_perc)" on page 144 in the PBS Professional Administrator's Guide.

Resgroup, Grp

Group ID of the entity's parent group

Shares
: The number of shares allotted to the entity

usage
: The amount of usage by the entity

usage/perc
: The value a scheduler uses to the pick which entity has priority over another. The smaller the number the higher the priority.

2.6.4 See Also

"Using Fairshare" on page 139 in the PBS Professional Administrator's Guide.

2.7 pbsnodes

Query PBS host or vnode status, mark hosts free or offline, change the comment for a host, or output vnode information

2.7.1 Synopsis

pbsnodes [-o | -r] [-s <server name>] [-C <comment>] <hostname> [<hostname> ...]

pbsnodes [-l] [-s <server name>]

pbsnodes -v <vnode> [<vnode> ...] [-s <server name>]

pbsnodes -a[v] [-S[j][L]] [-F json|dsv [-D <delimiter>]] [- s <server name>]

pbsnodes [-H] [-S[j][L]] [-F json|dsv [-D <delimiter>]] <hostname> [<hostname> ...]

pbsnodes --version

2.7.2 Description

The `pbsnodes` command is used to query the status of hosts or vnodes, to mark hosts *FREE* or *OFFLINE*, to edit a host's comment attribute, or to output vnode information. The `pbsnodes` command obtains host information by sending a request to the PBS server.

2.7.2.1 Using pbsnodes

To list all vnodes:

```
pbsnodes -av
```

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.

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

To offline a single vnode in a multi-vnoded system, use:

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

2.7.2.2 Output

The order in which hosts or vnodes are listed in the output of the `pbsnodes` command is undefined. Do not rely on output being ordered.

If you print attributes, `pbsnodes` prints out only those attributes which are not at default values.

2.7.2.3 Permissions

PBS Manager or Operator privilege is required to execute `pbsnodes` with the `-o` or `-r` options, to view custom resources which have been created to be invisible to users, and to see some output such as PBS version.

2.7.3 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 used with the -v option, lists all vnodes.

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>".

-C <comment>

Sets the comment attribute for the specified host(s) to the value of *comment*. Comments containing spaces must be quoted. The comment string is limited to 80 characters. Usage:

```
pbsnodes -C <comment> <hostname> [<hostname> ...]
```

To set the comment for a vnode:

```
qmgr -c "s n <vnode name> comment=<comment>"
```

-F dsv [-D <delimiter>]

Prints output in delimiter-separated value format. Optional delimiter specification. Default delimiter is vertical bar ("|").

-F json

Prints output in JSON format.

-H <hostname> [<hostname> ...]

Prints all non-default-valued attributes for specified hosts and all vnodes on specified hosts.

-j

Displays the following job-related headers for specified vnodes:

Table 2-2: Output for -j Option

Header	Width	Description
vnode	15	Vnode name
state	15	Vnode state
njobs	6	Number of jobs on vnode
run	5	Number of running jobs at vnode
susp	6	Number of suspended jobs at vnode
mem f/t	12	Vnode memory free/total
ncpus f/t	7	Number of CPUs at vnode free/total
nmics f/t	7	Number of MICs free/total
ngpus f/t	7	Number of GPUs at vnode free/total
jobs	No restriction	List of job IDs on vnode

Note that nmics is a custom resource that must be created by the administrator if you want it displayed here.

Each subjob is treated as a unique job.

-L

Displays output with no restrictions on column width.

-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*, it is listed. For hosts with multiple vnodes, only hosts where all vnodes are marked as *DOWN* or *OFFLINE* are listed.

-o <hostname> [<hostname> ...]

Marks listed hosts as *OFFLINE* even if currently in use. This is different from being marked *DOWN*. A host that is marked *OFFLINE* continues to execute the jobs already on it, but is removed from the scheduling pool (no more jobs are 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 all vnodes on a multi-vnoded machine:

```
pbsnodes -o <name of natural vnode>
```

To offline a single vnode on a multi-vnoded system, use:

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

Requires PBS Manager or Operator privilege.

-r <hostname> [<hostname> ...]

Clears *OFFLINE* from listed hosts.

-S

Displays the following vnode information:

Table 2-3: Output for -S Option

Header	Width	Description
name	15	Vnode name
state	15	Vnode state
OS	8	Value of OS custom resource, if any
hardware	8	Value of hardware custom resource, if any
host	15	Hostname
queue	10	Value of vnode's queue attribute
ncpus	7	Number of CPUs at vnode
nmics	7	Number of MICs at vnode
mem	8	Vnode memory
ngpus	7	Number of GPUs at vnode
comment	No restriction	Vnode comment

Note that nmics and OS are custom resources that must be created by the administrator if you want their values displayed here.

-s <server name>

Specifies the PBS server to which to connect.

-v [<vnode> [<vnode> ...]]

Lists all non-default-valued attributes for each specified vnode.

With no arguments, prints one entry for each vnode in the PBS complex.

With one or more vnodes specified, prints one entry for each specified vnode.

When used with `-a`, lists all vnodes.

--version

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

2.7.4 Operands

<server name>

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

<hostname> [<hostname> ...]

Specifies the host(s) to be queried or operated on.

<vnode> [<vnode> ...]

Specifies the vnode(s) to be queried or operated on.

2.7.5 Exit Status

Zero

Success

Greater than zero

- Incorrect operands are given
- `pbsnodes` cannot connect to the server
- There is an error querying the server for the vnodes

2.7.6 See Also

The PBS Professional Administrator's Guide, "qmgr" on page 146

2.8 pbsrun

General-purpose wrapper script for mpirun

2.8.1 Synopsis

pbsrun

pbsrun --version

2.8.2 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, it translates any mpirun call of the form:

mpirun [<options>] <executable> [<args>]

into

mpirun [<options>] pbs_attach [<special options to pbs_attach>] <executable> [<args>]

where *special options* refers to any option needed by pbs_attach to do its job (e.g. *-j $PBS_JOBID*).

If the wrapper script is executed outside of PBS, a warning is issued about "not running under PBS", but it proceeds as if the actual program had been called in standalone fashion.

The pbsrun wrapper script is not meant to be executed directly; 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 (The wrapper is **deprecated** as of 14.2.1)
- MPICH-MX's mpirun (mpirun.ch_mx) with rsh/ssh (The wrapper is **deprecated** as of 14.2.1)
- MPICH-GM's mpirun (mpirun.mpd) with MPD (The wrapper is **deprecated** as of 14.2.1)
- MPICH-MX's mpirun (mpirun.mpd) with MPD (The wrapper is **deprecated** as of 14.2.1)
- MPICH2's mpirun
- Intel MPI's mpirun (The wrapper is **deprecated** as of 13.0)
- MVAPICH1's mpirun (The wrapper is **deprecated** as of 14.2.1)
- MVAPICH2's mpiexec

2.8.3 Options

--version

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

2.8.4 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"
```

2.8.4.1 Initialization Script Options

options_to_retain

Space-separated list of options and values that pbsrun.<mpirun version/flavor> passes on to the actual mpirun call. Options must begin with "-" or "--", and option arguments must be specified by some arbitrary name with left and right arrows, as in "<val1>".

options_to_ignore

Space-separated list of options and values that pbsrun.<mpirun version/flavor> does not pass on to the actual mpirun call. Options must begin with "-" or "--", and option arguments must be specified by arbitrary names with left and right arrows, as in "<n>".

options_to_transform

Space-separated list of options and values that pbsrun modifies before passing on to the actual mpirun call.

options_to_fail

Space-separated list of options that will cause pbsrun to exit upon encountering a match.

options_to_configfile

Single option and value that refers to the name of the configuration file containing command line segments found in certain versions of mpirun.

options_with_another_form

Space-separated list of options and values that can be found in *options_to_retain*, *options_to_ignore*, or *options_to_transform*, whose syntax has an alternate, unsupported form.

pbs_attach

Path to pbs_attach, which is called before the *executable* argument of mpirun.

options_to_pbs_attach

Special options to pass to the pbs_attach call. You may pass variable references (e.g. *$PBS_JOBID*) and they are substituted by pbsrun to actual values.

If pbsrun encounters any option not found in *options_to_retain*, *options_to_ignore*, and *options_to_transform*, 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(), such as 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`.

2.8.4.2 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.

2.8.5 Versions/Flavors of `mpirun`

2.8.5.1 MPICH-GM `mpirun` (`mpirun.ch_gm`) with `rsh/ssh`: `pbsrun.ch_gm`

2.8.5.1.i Syntax

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

Deprecated. 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` were used.

2.8.5.1.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 $PBS_NODEFILE.

-np

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

2.8.5.1.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
```

2.8.5.2 MPICH-MX `mpirun` (`mpirun.ch_mx`) with `rsh/ssh`: `pbsrun.ch_mx`

2.8.5.2.i Syntax

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

The wrapper is **deprecated**. 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 PBS to track all MPICH-MX 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_mx` were used.

2.8.5.2.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 $PBS_NODEFILE.

-np

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

2.8.5.2.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
```

2.8.5.3 MPICH-GM `mpirun` (`mpirun.mpd`) with MPD: `pbsrun.gm_mpd`

2.8.5.3.i Syntax

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

The wrapper is **deprecated**. 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 PBS to track all MPICH-GM 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_gm` with MPD were used.

2.8.5.3.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 $PBS_NODEFILE.

-np

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

2.8.5.3.iii Startup/Shutdown

The script starts MPD daemons on each of the unique hosts listed in $PBS_NODEFILE, using either `rsh` or `ssh` based on the value of the environment variable RSHCOMMAND. The default is `rsh`.

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

2.8.5.3.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
```

2.8.5.4 MPICH-MX `mpirun` (`mpirun.mpd`) with MPD: `pbsrun.mx_mpd`

2.8.5.4.i Syntax

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

The wrapper is **deprecated**. 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 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 were used.

2.8.5.4.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 $PBS_NODEFILE.

-np

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

2.8.5.4.iii Startup/Shutdown

The script starts MPD daemons on each of the unique hosts listed in $PBS_NODEFILE, using either `rsh` or `ssh` , based on the value of the environment variable RSHCOMMAND. The default is `rsh`.

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

2.8.5.4.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
```

2.8.5.5 MPICH2 `mpirun`: `pbsrun.mpich2`

2.8.5.5.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 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` were used.

2.8.5.5.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 $PBS_NODEFILE.

MPICH2 s mpirun -localonly <num processes>

For specifying number of processes to run locally. Not supported. The user is advised instead to use the equivalent arguments: `-np <num processes> -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 one argument and "`-localonly <num processes>`" has two arguments).

-np

If the user does not specify the `-np` option, 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*.

2.8.5.5.iii Startup/Shutdown

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

2.8.5.5.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.

2.8.5.6 Intel MPI `mpirun`: `pbsrun.intelmpi`

Wrapping Intel MPI, and support for `mpdboot`, are **deprecated**.

2.8.5.6.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 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` were used.

2.8.5.6.ii Options Handling

If executed inside a PBS job script, all of the options to the PBS interface to Intel 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 $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 MPI's `mpirun -s` argument to `mpdboot` is 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 the -np option, 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*.

2.8.5.6.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.

2.8.5.6.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
```

2.8.5.7 MVAPICH1 mpirun: pbsrun.mvapich1

2.8.5.7.i Syntax

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

The wrapper is **deprecated**. 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 PBS to be aware of all MVAPICH1 ranks and to 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 were used.

2.8.5.7.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 $PBS_NODEFILE is used.

2.8.5.7.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
```

2.8.5.8 MVAPICH2 `mpiexec: pbsrun.mvapich2`

2.8.5.8.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 PBS to be aware of all MVAPICH2 ranks and to 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` were used.

2.8.5.8.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 <hostname>

The *hostname* argument contents are ignored.

-machinefile <file>

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

2.8.5.8.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
```

2.8.6 Requirements

The `mpirun` being wrapped must be installed and working on all the vnodes in the PBS cluster.

2.8.7 Errors

If `pbsrun` encounters any option not found in *options_to_retain*, *options_to_ignore*, and *options_to_transform*, it is flagged as an error.

2.8.8 See Also

The PBS Professional Administrator's Guide, "pbs_attach" on page 55, "pbsrun_wrap" on page 51, "pbsrun_unwrap" on page 50

2.9 pbsrun_unwrap

Unwraps `mpirun`, reversing `pbsrun_wrap`

2.9.1 Synopsis

pbsrun_unwrap pbsrun.<mpirun version/flavor>

pbsrun_unwrap --version

2.9.2 Description

The `pbsrun_unwrap` script is used to reverse the actions of the `pbsrun_wrap` script.

Use `pbsrun_wrap` to wrap `mpirun`.

Using `pbsrun_unwrap` for Intel MPI is **deprecated** as of 13.0.

2.9.2.1 Syntax

pbsrun_unwrap pbsrun.<mpirun version/flavor>

For example, running the following:

```
pbsrun_unwrap pbsrun.ch_gm
```

causes the following actions:

1. Checks for a link in `$PBS_EXEC/lib/MPI/pbsrun.ch_gm.link`; If one exists, get the pathname it points to, for example:

   `/opt/mpich-gm/bin/mpirun.ch_gm.actual`
2. rm $PBS_EXEC/lib/MPI/pbsrun.mpirun.ch_gm.link
3. rm /opt/mpich-gm/bin/mpirun.ch_gm
4. rm $PBS_EXEC/bin/pbsrun.ch_gm
5. mv /opt/mpich-gm/bin/mpirun.ch_gm.actual /opt/mpich-gm/bin/mpirun.ch_gm

2.9.3 Options

--version

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

2.9.4 See Also

The PBS Professional Administrator's Guide, "pbs_attach" on page 55, "pbsrun" on page 40, "pbsrun_wrap" on page 51

2.10 pbsrun_wrap

General-purpose script for wrapping mpirun in pbsrun

2.10.1 Synopsis

pbsrun_wrap [-s] <path to actual mpirun> pbsrun.<mpirun version/flavor>

pbsrun_wrap --version

2.10.2 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 "pbs_attach" on page 55 and "pbsrun" on page 40.

Use pbsrun_unwrap to reverse the process.

Using pbsrun_wrap for Intel MPI is **deprecated** as of 13.0.

2.10.2.1 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:

1. Save original mpirun.ch_gm script:
   ```
   mv /opt/mpich-gm/bin/mpirun.ch_gm /opt/mpich/gm/bin/mpirun.ch_gm.actual
   ```
2. Instantiate pbsrun wrapper script as pbsrun.ch_gm:
   ```
   cp $PBS_EXEC/bin/pbsrun $PBS_EXEC/bin/pbsrun.ch_gm
   ```
3. 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
   ```
4. 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
   ```

2.10.3 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 gets the error:

```
Not running under PBS exiting since strict_pbs is enabled; execute only in PBS
```

--version

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

2.10.4 Requirements

The mpirun being wrapped must be installed and working on all the vnodes in the PBS complex.

2.10.5 See Also

The PBS Professional Administrator's Guide, "pbs_attach" on page 55, "pbsrun" on page 40, "pbsrun_unwrap" on page 50

2.11 pbs_account

For Windows. Manage PBS service account

2.11.1 Synopsis

pbs_account [-a <PBS service account name>] [-c [<password>]] [--ci] [--instid <instance ID>] [-o <output path>] [-p [<password>]] [--reg <service path>] [-s] [--unreg <service path>]

2.11.2 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.

2.11.2.1 Permissions

This command can be run by administrators only.

2.11.2.2 Platforms

This command is available on Windows only.

2.11.2.3 Caveats

Using `pbs_account --unreg` and `pbs_account --reg` stops and restarts MoM, which can kill jobs.

2.11.3 Options

-a <account name>

Specifies service account name.

-c [<password>]

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

Gives necessary privileges to the specified account: *Create Token Object*, *Replace Process Level Token*, *Log on as a Service*, and *Act as Part of the Operating System*

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

--ci

Informational only. Prints actions taken by `pbs_account` while creating PBS service account when operations are performed.

--instid <instance ID>

Specifies the instance ID when registering or unregistering multiple instances of a service. Example:

```
pbs_account --reg "C:\Program Files (x86)\PBS Pro_2\exec\sbin\pbs_mom" --instid 2 -a <username>
    -p <password>

pbs_account --unreg "C:\Program Files (x86)\PBS Pro_2\exec\sbin\pbs_mom" --instid 2
```

-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.

--reg <path to service>

Registers the PBS service with the SCM, instructing it to run the services under the PBS service account. *path to service* must be in double quotes. Restarts MoM.

-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.

--unreg <path to service>

Unregisters the PBS service with the SCM. *path to service* must be in double quotes. Stops MoM.

(no options)

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

2.11.4 Examples

Example 2-1: To create the PBS service account:

```
pbs_account -c -s -p <password>
```

Example 2-2: To change the PBS service account:

```
pbs_account --reg <service path> -a <PBS service account name>
```

Example 2-3: To register the server, scheduler, MoM, comm, and `rshd` services:

```
pbs_account --reg "\Program Files\PBS\exec\sbin\pbs_server.exe" -p <password>
pbs_account --reg "\Program Files\PBS\exec\sbin\pbs_mom.exe" -p <password>
pbs_account --reg "\Program Files\PBS\exec\sbin\pbs_sched.exe" -p <password>
pbs_account --reg "\Program Files\PBS\exec\sbin\pbs_comm.exe" -p <password>
pbs_account --reg "\Program Files\PBS\exec\sbin\pbs_rshd.exe" -p <password>
```

2.11.5 Exit Value

Zero

Upon success

2.12 pbs_attach

Attaches a session ID to a PBS job

2.12.1 Synopsis

Linux

pbs_attach [-j <job ID>] [-m <port number>] -p <PID>

pbs_attach [-j <job ID>] [-m <port number>] [-P] [-s] <cmd> [<arg> ...]

pbs_attach --version

Windows

pbs_attach [-c <path to script>] [-j <job ID>] [-m <port number>] -p <PID>

pbs_attach [-c <path to script>] [-j <job ID>] [-m <port number>] [-P] [-s] <cmd> [<arg> ...]

pbs_attach --version

2.12.2 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.

2.12.3 Options to `pbs_attach`

-c <path to script>

Windows only. Specified command is invoked using a new command shell. In order to spawn and attach built-in DOS commands such as `set` or `echo`, it is necessary to open the task using a `cmd` shell. The new command shell, `cmd.exe`, is attached as a task to the PBS job. The `pbs_attach` command spawns a program using a new command shell when attaching a batch script, or when invoked with the `-c` option.

-j <job ID>

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

-m <port number>

The port at which to contact MoM. Default: value of $PBS_MANAGER_SERVICE_PORT from `pbs.conf`.

-p <PID>

Process ID whose session ID is to be attached to the job. Default: process ID of `pbs_attach`. Cannot be used with the `-P` or `-s` options or the *cmd* operand.

-P

Attach sessions of both `pbs_attach` and the parent of `pbs_attach` to job. When used with `-s` option, the sessions of the new `fork()`ed `pbs_attach` and its parent, which is `pbs_attach`, are attached to the job. Cannot be used with the `-p` or `-s` options or the *cmd* operand.

-s

Starts a new session and attaches it to the job; `pbs_attach` calls `fork()`, then the child `pbs_attach` first calls `setsid()` and then calls `tm_attach` to attach the new session to the job. 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.

2.12.4 Operands

cmd

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

2.12.5 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.

2.12.6 See Also

The PBS Professional Administrator's Guide, "pbs_mom" on page 71, "pbs_tmrsh" on page 117, "TM Library", on page 69 of the PBS Professional Programmer's Guide

2.13 pbs_comm

Starts the PBS communication daemon

2.13.1 Synopsis

Linux:

pbs_comm [-N] [-r <other routers>] [-t <number of threads>]

pbs_comm --version

Windows:

pbs_comm.exe [-R|-U|-N] [-r <other routers>] [-t <number of threads>]

pbs_comm --version

2.13.2 Description

The PBS communication daemon, `pbs_comm`, handles communication between daemons, except for scheduler-server and server-server communication, which uses TCP. The server, scheduler(s), and MoMs are connected by one or more `pbs_comm` daemons.

See "Communication" on page 47 in the PBS Professional Installation & Upgrade Guide.

2.13.3 Options to pbs_comm

-N

Runs the communication daemon in standalone mode.

-r <other routers>

List of other `pbs_comm` daemons to which this `pbs_comm` must connect. This is equivalent to the `pbs.conf` variable PBS_COMM_ROUTERS. The command line overrides the variable. Format:

<hostname>[:<port number>][,<hostname>[:<port number>]]

-R

Registers the `pbs_comm` process. Available under Windows only.

-t <number of threads>

Number of threads the `pbs_comm` daemon uses. This is equivalent to the `pbs.conf` variable PBS_COMM_THREADS. The command line overrides the variable. Format:

Integer

-U

Unregisters the `pbs_comm` process. Available under Windows only.

2.13.4 Configuration Parameters

PBS_LEAF_ROUTERS

Parameter in `/etc/pbs.conf`. Tells an endpoint where to find its communication daemon.

You can tell each endpoint which communication daemon it should talk to. Specifying the port is optional.

Format: *PBS_LEAF_ROUTERS=<hostname>[:<port number>][,<hostname>[:<port number>]]*

PBS_COMM_ROUTERS

Parameter in /etc/pbs.conf. Tells a pbs_comm where to find its fellow communication daemons.

When you add a communication daemon, you must tell it about the other pbs_comms in the complex. When you inform communication daemons about each other, you only tell one of each pair about the other. Do not tell both about each other. We recommend that an easy way to do this is to tell each new pbs_comm about each existing pbs_comm, and leave it at that.

Format: *PBS_COMM_ROUTERS=<hostname>[:<port number>][,<hostname>[:<port number>]]*

PBS_COMM_THREADS

Parameter in /etc/pbs.conf. Tells pbs_comm how many threads to start.

By default, each pbs_comm process starts four threads. You can configure the number of threads that each pbs_comm uses. Usually, you want no more threads than the number of processors on the host.

Maximum allowed value: *100*

Format: *Integer*

Example:

```
PBS_COMM_THREADS=8
```

PBS_COMM_LOG_EVENTS

Parameter in /etc/pbs.conf. Tells pbs_comm which log mask to use.

By default, pbs_comm produces few log messages. You can choose more logging, usually for troubleshooting. See "Logging and Errors with TPP" on page 56 in the PBS Professional Installation & Upgrade Guide for logging details.

Format: *Integer*

Default: *511*

Example:

```
PBS_COMM_LOG_EVENTS=<log level>
```

PBS_LEAF_NAME

Parameter in /etc/pbs.conf. Tells endpoint what name to use for network. The value does not include a port, since that is usually set by the daemon.

By default, the name of the endpoint's host is the hostname of the machine. You can set the name where an endpoint runs. This is useful when you have multiple networks configured, and you want PBS to use a particular network. TPP internally resolves the name to a set of IP addresses, so you do not affect how pbs_comm works.

Format: *String*

Example:

```
PBS_LEAF_NAME=host1
```

PBS_START_COMM

Parameter in /etc/pbs.conf. Tells PBS init script whether to start a pbs_comm on this host if one is installed. When set to *1*, pbs_comm is started.

Just as with the other PBS daemons, you can specify whether each host should start pbs_comm.

Format: *Boolean*

Default: *0*

Example:

```
PBS_START_COMM=1
```

2.13.5 Communication Daemon Logfiles

The pbs_comm daemon creates its log files under $PBS_HOME/comm_logs. This directory is automatically created by the PBS installer.

In a failover configuration, this directory is in the shared PBS_HOME, and is used by the pbs_comm daemons running on both the primary and secondary servers. This directory must never be shared across multiple pbs_comm daemons in any other case.

The log filename format is *yyyymmdd* (the same as for other PBS daemons).

The log record format is the same as used by other pbs daemons, with the addition of the thread number and the daemon name in the log record. The log record format is as follows:

<date and time>;<event code>;<daemon name>(<thread number>);<object type>;<object name>;<message>

Example:

```
03/25/2014 15:13:39;0d86;host1.example.com;TPP;host1.example.com(Thread 2);Connection from leaf
    192.168.184.156:19331, tfd=81 down
```

2.13.6 Signal Handling by Communication Daemon

The pbs_comm daemon handles the following signals:

HUP

Re-reads the value of $PBS_COMM_LOG_EVENTS from pbs.conf.

TERM

The pbs_comm daemon exits.

2.14 pbs_dataservice

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

2.14.1 Synopsis

pbs_dataservice [start | stop | status]

2.14.2 Description

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

2.14.2.1 Permission

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

2.14.3 Arguments

start
: Starts the PBS data service.

stop
: Stops the PBS data service.

Can be used only when the 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`

2.14.4 Exit Status

Zero
: Success

Non-zero
: Failure

2.15 pbs_ds_password

Sets or changes data service user account or its password

2.15.1 Synopsis

pbs_ds_password [-C <username>] [-r]

2.15.2 Description

You can use this command to change the user account or account password for the data service.

2.15.2.1 Passwords

Blank passwords are not allowed.

If you type in a password, make sure it does not contain restricted characters. The `pbs_ds_password` command generates passwords containing the following characters:

0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!@#$%^&()_+*

When creating a password manually, do not use \ (backslash) or ‘ (backquote). This can prevent certain commands such as `pbs_server`, `pbs_ds_password`, and `printjob` from functioning properly, as they rely on connecting to the database. The format is also described in "PBS Password” on page 347.

2.15.2.2 Permissions

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

2.15.2.3 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. If failover is configured, stop the secondary server, remove definitions for PBS_PRIMARY and PBS_SECONDARY from `pbs.conf` on the primary server host, start PBS, run `pbs_ds_password`, stop PBS, replace the definitions, and start PBS again.

2.15.3 Options to `pbs_ds_password`

-C <username>

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

On Linux-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 options)

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

2.15.4 Exit Status

Zero
: Success

Non-zero
: Failure

2.16 pbs_hostn

Reports hostname and network address(es)

2.16.1 Synopsis

pbs_hostn [-v] <hostname>

pbs_hostn --version

2.16.2 Description

The `pbs_hostn` command takes a hostname, and reports the results of both the `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.

2.16.3 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.

2.16.4 Operands

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

2.16.5 Standard Error

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

2.16.6 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.

2.17 pbs_idled

Runs PBS daemon that monitors the console and informs pbs_mom of idle time

2.17.1 Linux Synopsis

pbs_idled [-D <display>] [-r <reconnect delay>] [-w <wait time>]

pbs_idled --version

2.17.2 Windows Synopsis

pbs_idled [start | stop]

pbs_idled --version

2.17.3 Linux Description

On Linux, the pbs_idled program monitors 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 vnode is busy.

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

2.17.4 Windows Description

On Windows, pbs_idled reads its polling interval from a file called idle_poll_time which is created by MoM. The service 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.

2.17.5 Linux Options to pbs_idled

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

-r <reconnect delay>
: Time to wait before we try to reconnect to the X display if the previous attempt was unsuccessful

-w <wait time>
: Interval between times when the daemon checks for events or pointer movement

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

2.17.6 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.

2.17.7 See Also

The PBS Professional Administrator's Guide

xhost(1), xauth(1)

2.18 pbs_iff

Tests authentication with the server

2.18.1 Usage

pbs_iff [-t] <server host> <server port>

pbs_iff --version

2.18.2 Description

Called from the pbs_connect() IFL API to authenticate a connection with the PBS server. Designed to be called internally by PBS commands and components, to be used by our IFL layer to talk to the server.

If `pbs_iff` cannot authenticate, it returns an error message.

2.18.2.1 Required Privilege

Can be run by any user.

It's a setuid root binary so it runs as the user who requests a connection to a server but it becomes root so that it can grab a privileged port.

2.18.3 Options to pbs_iff

-t
: Test mode; means test whether `pbs_iff` can authenticate with the server

--version
: Reports version and exits; can only be used alone

2.18.4 Arguments to pbs_iff

daemon host
: Host where server is running

daemon port
: Port on which server is listening; default is 15001

2.18.5 Exit Status

Zero
: If `pbs_iff` is able to contact the server at the specified port

Non-zero
: If `pbs_iff` is unable to contact the server at the specified port

2.19 pbs_interactive

Windows. Register, unregister, or get the version of PBS_INTERACTIVE service

2.19.1 Synopsis

pbs_interactive [R | U]

pbs_interactive --version

2.19.2 Description

The pbs_interactive command registers, unregisters, or gets the version of 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.

2.19.2.1 Required Privilege

Admin privilege is required to use this command.

2.19.3 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.

2.20 pbs_lamboot

PBS front end to LAM's `lamboot` program

2.20.1 Synopsis

pbs_lamboot

pbs_lamboot --version

2.20.2 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 vnodes 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*.

2.20.3 Options

--version

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

2.20.4 Operands

The operands for `pbs_lamboot` are the same as for `lamboot`.

2.20.5 Environment Variables and Path

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

2.20.6 See Also

The PBS Professional Administrator's Guide, `lamboot(1)`, "TM Library", on page 69 of the PBS Professional Programmer's Guide

2.21 pbs_migrate_users

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

2.21.1 Synopsis

pbs_migrate_users <old server> <new server>

pbs_migrate_users --version

2.21.2 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.

2.21.3 Options

--version

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

2.21.4 Operands

The format of *old server* and *new server* is

<hostname>[:<port number>]

2.21.5 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

2.21.6 See Also

"pbs_password" on page 81

2.22 pbs_mkdirs

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

2.22.1 Synopsis

pbs_mkdirs

pbs_mkdirs [mom | sched | server]

2.22.2 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`

2.22.2.1 Required Privilege

You must have Administrator privilege to run this command.

2.22.3 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_log`

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.

2.22.4 See Also

The PBS Professional Administrator's Guide, "pbs_probe" on page 83

2.23 pbs_mom

Runs the PBS job monitoring and execution daemon

2.23.1 Synopsis

pbs_mom [-a <alarm timeout>] [-C <checkpoint directory>] [-c <config file>] [-d <MoM home directory>] [-L <logfile>] [-M <MoM port>] [-N] [-n <nice value>] [-p|-r] [-R <inter-MoM communication port>] [-S <server port>] [-s <script options>]

pbs_mom --version

2.23.2 Description

The `pbs_mom` command starts the PBS job monitoring and execution daemon, called *MoM*.

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 as a set of virtual nodes, in which case one MoM manages all of the host's vnodes. See the PBS Professional Administrator's Guide.

2.23.2.1 Logging

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 writes 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.

2.23.2.2 Required Permission

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

2.23.2.3 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 contains CPU 0.

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

The CPUSET_CPU_EXCLUSIVE flag prevents CPU 0 from being used by the MoM in the creation of job cpusets. This flag is set by default, 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.

2.23.2.3.i HPE SGI Systems Running Supported Versions of HPE MPI

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 machines that run supported versions of HPE MPI.

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

2.23.2.4 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.

2.23.3 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 "Specifying Checkpoint Path" on page 424 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 is 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 <MoM 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 <MoM port>

Specifies the port number on which MoM will listen for server requests and instructions. Overrides PBS_MOM_SERVICE_PORT setting in `pbs.conf` and environment variable.

Format: *Integer port number*.

Default: *15002*.

-n <nice value>

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.

HPE SGI systems running HPE MPI:

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 <inter-MoM communication port>

Specifies the port number on which MoM will listen for pings, resource information requests, communication from other MoMs, etc. Overrides PBS_MANAGER_SERVICE_PORT setting in pbs.conf and environment variable.

Format: *Integer port number*

Default: *15003*

-S <server port>

Specifies the port number on which pbs_mom initially contacts 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. The *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*. Example:

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

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 and pbs_mom will print a diagnostic message and exit with a nonzero status.

-s remove <scriptname>

The configuration file named *scriptname* is removed if it exists. Example:

```
pbs_mom -s remove <scriptname>
```

If the given name does not exist or if an attempt is made to remove a script with the reserved "PBS" prefix, the operation fails, a diagnostic is presented, and pbs_mom exits with a nonzero status.

-s show <scriptname>

Causes the contents of the named script to be printed to standard output. Example:

```
pbs_mom -s show <scriptname>
```

If *scriptname* does not exist, the operation fails, a diagnostic is presented, and pbs_mom exits with a non-zero status.

-s list

Causes pbs_mom to list the set of PBS-prefixed and site-defined configuration files in the order in which they are 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.

2.23.4 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.

2.23.5 Signal Handling

pbs_mom handles the following signals:

SIGHUP

The pbs_mom daemon rereads its configuration files, closes and reopens the log file, and reinitializes 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.

2.23.6 Exit Status

Zero

Upon success

Greater than zero

- If the pbs_mom daemon fails to start
- If the -s insert option is used with an existing *scriptname*
- If the administrator attempts to add a script whose name begins with "*PBS*"
- 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*"
- If the administrator attempts to use the -s show option on a nonexistent script

2.23.7 See Also

The PBS Professional Administrator's Guide

2.24 pbs_mpihp

Runs an MPI application in a PBS job with HP MPI

2.24.1 Synopsis

pbs_mpihp [-h <hostname>] [-np <number>] [<other HP mpirun options>] <program> [<args>]

pbs_mpihp [<HP mpirun options>] -f <appfile> [-- [<extra args>]]

pbs_mpihp --version

2.24.2 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`.

When `pbs_mpihp` is invoked from a PBS job, it processes the command line arguments, then calls standard HP `mpirun` to actually start the MPI ranks. The ranks created are mapped onto CPUs on the vnodes allocated to the PBS job. The environment variable MPI_REMSH is set to `$PBS_EXEC/bin/pbs_tmrsh`. This causes 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 points to standard HP `mpirun`. If it does not exist, a call to `mpirun -version` is made to determine whether it is HP `mpirun`. If so, the call is made to "`mpirun`" without an absolute path. If HP `mpirun` cannot be found, an error is output, all temp files are cleaned up and the script exits with value 127.

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

2.24.2.1 Configuration

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>
```

2.24.2.2 Usage

Usage is the same as for HP `mpirun`.

`pbs_mpihp <program>` allows one executable to be specified.

`pbs_mpihp -f <appfile>` 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 is read to determine what executables are to be run and how many processes are started for each.

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

2.24.3 Options to `pbs_mpihp`

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

-client
Not supported.

-f <appfile>
The specified *appfile* is read by `pbs_mpihp`.

-h <hostname>
Ignored by `pbs_mpihp`.

-l <username>
Ignored by `pbs_mpihp`.

-np <number>
Specifies the *number* of processes to run on the PBS vnodes.

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

2.24.4 Exit Values

127
If HP `mpirun` cannot be found

2.24.5 See Also

The PBS Professional Administrator's Guide
`mpirun(1)`

2.25 pbs_mpilam

Runs MPI programs under PBS with LAM MPI

2.25.1 Synopsis

pbs_mpilam [<mpilam options>]

pbs_mpilam --version

2.25.2 Description

The PBS command `pbs_mpilam` replaces the standard `mpirun` command in a PBS LAM MPI job.

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), PBS does not manage the spawned user processes of each program.

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

2.25.2.1 Prerequisites

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

2.25.2.2 Usage

Usage is the same as for LAM `mpirun`. All options are passed directly to `mpirun`.

2.25.3 Options to `pbs_mpilam`

<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.

2.25.4 See Also

The PBS Professional Administrator's Guide

`mpirun(1)`

2.26 pbs_mpirun

Deprecated. Runs MPI programs under PBS with MPICH

2.26.1 Synopsis

pbs_mpirun [<mpirun options>]

pbs_mpirun --version

2.26.2 Description

The PBS command `pbs_mpirun` replaces the standard `mpirun` command in a PBS MPICH job using P4.

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

2.26.2.1 Prerequisite

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

2.26.2.2 Usage

Usage is the same as for `mpirun`, except for the `-machinefile` option. All other options are passed directly to `mpirun`.

2.26.3 Options to `pbs_mpirun`

<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.

2.26.4 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.

2.26.5 See Also

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

2.27 pbs_password

Sets or updates password of a PBS user

2.27.1 Synopsis

pbs_password [-d] [-r] [-s <server name>] [<username>]

pbs_password --version

2.27.2 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.

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 login password on the current host, only the password that is cached in PBS.

The `pbs_password` command encrypts the password obtained from the user before sending it to the PBS server.

2.27.2.1 Required Privilege

An unprivileged user can use this command to set or update their own password.

Root or Admin can use this command to set or update the password of another user.

2.27.3 Options to `pbs_password`

(no options)

The user executing this command is prompted for a new password. This user's password credential on the default PBS server is updated to the prompted password. Any user jobs previously held due to an invalid password are not released.

-d

Deletes password. If *username* is not specified, deletes the password of the current user. If *username* is specified and the current user is root or Admin, deletes the password of the specified user.

-r

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

-s <server name>

The *server name* where the password is to be changed.

<username>

The password credential of user *username* is updated to the prompted password. If *username* 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 *username* 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 `hosts.equiv` file, or the current user has an entry in *username*'s `HOMEDIR\.rhosts` file

--version

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

2.27.4 Exit Status

Table 2-4: 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*

2.27.5 See Also

The PBS Professional Administrator's Guide

2.28 pbs_probe

Deprecated. Reports PBS diagnostic information and fixes permission errors

2.28.1 Synopsis

pbs_probe [-f | -v]

pbs_probe --version

2.28.2 Description

The `pbs_probe` command reports post-installation information useful for PBS diagnostics, and fixes permission errors.

2.28.2.1 Information Sources

- Information that is supplied on the command line
- The file `/etc/pbs.conf`
- The file `/etc/init.d/pbs`
- The values of any of the following environment variables; these 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

2.28.2.2 Required Privilege

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

2.28.3 Options to `pbs_probe`

(no options)

Run in "report" mode. In this mode `pbs_probe` reports any permission errors detected in PBS infrastructure files. The command categorizes the errors and writes a list of them by category. Empty categories are not written.

-f

Run in "fix" mode. In this mode `pbs_probe` examines each of the relevant infrastructure files and, where possible, fixes any permission errors that it detects, and prints a message saying what got changed. If it is unable to fix a problem, it prints a message saying what was detected.

-v

Run in "verbose" mode. In this mode `pbs_probe` writes 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.

2.28.4 Standard Error

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

2.28.5 Exit Status

Exit code does not reflect results of probe; it reflects whether or not the program ran.

Zero
: When run correctly, whether or not `pbs_probe` finds any problems or errors

Non-negative
: When run incorrectly

2.28.6 See Also

The PBS Professional Administrator's Guide

2.29 pbs_python

Python interpreter for debugging a hook script from the command line

2.29.1 Synopsis

pbs_python --hook [-e <log event mask>] [-i <event input file>] [-L <log dir>] [-l <log file>] [-o <hook execution record>] [-r <resourcedef file>] [-s <site data file>] [<Python script>]

pbs_python <standard Python options>

pbs_python --version

2.29.2 Description

The PBS Python interpreter, `pbs_python`, is a wrapper for Python.

You can use the `pbs_python` wrapper that is shipped with PBS to debug hooks. Either:

- Use the `--hook` option to `pbs_python` to run `pbs_python` as a wrapper to Python, employing the `pbs_python` options. With the `--hook` option, you cannot use the standard Python options. The rest of this section covers how to use `pbs_python` with the `--hook` option.
- Do not use the `--hook` option, so `pbs_python` runs the Python interpreter, with the standard Python options, and without access to the `pbs_python` options.

2.29.2.1 Debugging Hooks

You can get each hook to write out debugging files, and then modify the files and use them as debugging input to `pbs_python`. Alternatively, you can write the files yourself.

Debugging files can contain information about the event, about the site, and about what the hook changed. You can use these as inputs to a hook when debugging.

For a complete description of using `pbs_python` with debugging files, see "Debugging Hooks" on page 149 in the PBS Professional Hooks Guide.

2.29.3 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.

-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 section 2.23, "pbs_mom", on page 71. Available only when `--hook` option is used.

-i <event 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().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.

-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 <hook execution record>

The hook execution record contains the changes made after executing the hook script, such as the attributes and resources set in any pbs.event() jobs and reservations, whether an action was accepted or rejected, and any pbs.reject() messages.

Example hook execution record:

```
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>

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

<resource name> type=<typename> [flag=<value>]

Available only when `--hook` option is used.

-s <site data file>

The site data file can contain any relevant information about the server, queues, vnodes, and jobs at the server. This file can be written by a hook or by the administrator.

When the hook writes it, this file contains the values that populate the server, queues, vnodes, reservations, and jobs, with all attributes and resources for which there are values.

The site data file is named *hook_<event type>_<hook name>_<random integer>.data*. It can be passed to `pbs_python` using the `-s <site data file>` option.

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.

2.29.4 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>
...
```

2.30 pbs_ralter

Modifies an existing advance or standing reservation

2.30.1 Summary

Alter an existing advance or standing reservation.

2.30.2 Synopsis

pbs_ralter [-E <end time>] [-I <block time>] [-m <mail points>] [-M <mail list>] [-N <reservation name>] [-R <start time>] <reservation ID>

2.30.3 Description

You can use the `pbs_ralter` command to alter an existing advance or standing reservation. You can change the start time, end time, events that generate mail, mail recipient list, and reservation name. You can change the end time for a running reservation with running jobs.

You can use this command to change an advance reservation or the next or current instance of a standing reservation.

After the change is requested, the change is either confirmed or denied. On denial of the change, the reservation is not deleted and is left as is, and the following message appears in the server's log:

```
Unable to alter reservation <reservation ID>
```

When a reservation is confirmed, the following message appears in the server's log:

```
Reservation alter successful for <reservation ID>
```

To find out whether or not the change was allowed:

- Use the `pbs_rstat` command: see whether you altered reservation attribute(s)
- Use the interactive option: check for confirmation after the blocking time has run out
- Check the server log for confirmation or denial messages

Before the change is confirmed or denied, the change is unconfirmed, and the reservation state is *AL*.

Once a reservation change is confirmed, the reservation state is *CO* or *RN*.

If the reservation has not started and it cannot be confirmed on the same vnodes, PBS searches for another set of vnodes.

2.30.3.1 Required Privilege

You must be the reservation owner or the PBS Administrator to run this command.

2.30.4 Options to pbs_ralter

-R <start time>

Specifies reservation's new start time. This option can be used either when the reservation is not running or there are no jobs are submitted to the reservation. You cannot use this option when a reservation is not empty and has started running.

The specifications for providing the time are the same as for `pbs_rsub`:

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 alter a reservation with the specification `-R 1110` at 11:15 a.m., it is interpreted as being for 11:10 a.m. 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*

-E <end time>

Specifies reservation's new end time. This option can be used even when the reservation is running and has jobs that are submitted to and/or are running in the reservation.

Format: *Datetime*

-I <block time>

Specifies interactive mode. The `pbs_ralter` command will block, up to *block time* seconds, while waiting for the reservation's change request to be confirmed or denied.

The value for *block time* must be positive. The `pbs_ralter` command returns either the status "*CONFIRMED*" or the status "*DENIED*".

Format: *Integer*

Default: *Not interactive*

-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 2-5: Suboptions to -m Option

Character	Meaning
a	Notify if reservation is terminated for any 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: No default; if not specified, mail events are unchanged.

-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: *<username>[@<hostname>][,<username>[@<hostname>]...]*

Default: No default; if not specified, user list is unchanged.

-N <reservation name>

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: No default; if not specified, reservation name is unchanged.

2.30.5 Operands

The `pbs_ralter` command takes a reservation ID.

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> specifies a reservation at a server other than the default server.

2.31 pbs_rdel

Deletes a PBS advance or standing reservation

2.31.1 Synopsis

pbs_rdel <reservation ID>[,<reservation ID>...]

pbs_rdel --version

2.31.2 Description

The `pbs_rdel` command deletes reservations in the order specified.

This command deletes the specified reservations, whether or not they are running, all jobs in the reservations, and the reservation queues.

2.31.2.1 Required Privilege

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

2.31.3 Options

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

2.31.4 Operands

The `pbs_rdel` command accepts one or more *reservation ID* 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> specifies a reservation at a server other than the default server.

2.31.5 Exit Status

Zero
: Upon success

Greater than zero
: Upon failure to process any operand

2.31.6 See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide, "pbs_rsub" on page 96, "pbs_rstat" on page 94, "Reservation Attributes" on page 295

2.32 pbs_release_nodes

Releases vnodes assigned to a PBS job

2.32.1 Synopsis

pbs_release_nodes [-j <job ID>] <vnode> [<vnode> [<vnode>] ...]

pbs_release_nodes [-j <job ID>] -a

pbs_release_nodes --version

2.32.2 Description

You can use the `pbs_release_nodes` command to release no-longer-needed vnodes assigned to a running job, before the job would normally release them. These vnodes are then available for use by other jobs.

You can specify the names of vnodes to be released, or you can release all vnodes not on the primary execution host that are assigned to a running job via the `-a` option.

2.32.2.1 Caveats and Restrictions

- You can release only vnodes that are not on the primary execution host. You cannot release vnodes on the primary execution host.
- The job must be running (in the *R* state).
- The `pbs_release_nodes` command is not supported on vnodes tied to Cray X* series systems (vnodes whose vntype has the "cray_" prefix).
- The `pbs_release_nodes` command is not supported for vnodes managed by cpuset MoMs; partial release of vnodes may result in leftover cpusets.
- If cgroups support is enabled, and `pbs_release_nodes` is called to release some but not all the vnodes managed by a MoM, resources on those vnodes that are part of a cgroup are not released until the entire cgroup is released.

2.32.2.2 Required Privilege

This command can be run by the job owner, the PBS Manager, Operator, and Administrator, as well as root on Linux and Admin on Windows.

2.32.3 Options to pbs_release_nodes

-a
: Releases all job vnodes not on the primary execution host. Cannot be used with list of vnode names.

-j <job ID>
: Specifies the job ID for the job whose vnode(s) are to be released.

(no options)
: Without the `-j` option, `pbs_release_nodes` uses the value of the PBS_JOBID environment variable as the job ID of the job whose vnodes are to be released.

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

2.32.4 Operands for pbs_release_nodes

The `pbs_release_nodes` command can take as an operand a list of vnodes. Format:

<vnode name> [<vnode name> [<vnode name>] ...]

Cannot be used with the `-a` option.

2.32.5 Usage

This command can be run at the command line, or called inside a job script, where it can use the value of the PBS_JOBID environment variable.

You can release any vnode that appears in the job's exec_vnode attribute that is not on the primary execution host. You can release a particular set of a job's vnodes, or you can release all of a job's non-primary-execution-host vnodes.

To release specific vnodes:

pbs_release_nodes [-j <job ID>] <vnode name> [<vnode name>] ...]

To release all of a job's vnodes that are not on the primary execution host:

pbs_release_nodes [-j <job ID>] -a

2.33 pbs_rstat

Shows status of PBS advance or standing reservations

2.33.1 Synopsis

pbs_rstat [-B] [-F] [-S] [<reservation ID>...]

pbs_rstat --version

2.33.2 Description

The `pbs_rstat` command shows the status of all reservations at the PBS server. Denied reservations are not displayed.

2.33.2.1 Required Privilege

This command can be run by a user with any level of PBS privilege. For full output, users without manager or operator privilege cannot print custom resources which were created to be invisible to users.

2.33.3 Output

The `pbs_rstat` command displays output in any of brief, short, or full formats.

See section 6.8, “Reservation Attributes”, on page 295 and section 8.6, “Reservation States”, on page 357.

2.33.4 Options to `pbs_rstat`

-B
: Brief output. Displays each reservation identifier only.

-F
: Full output. 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.

-S
: Short output. Displays a table showing the name, queue, owner, state, start time, duration, and end time of each reservation.

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

(no options)
: Short output. Same behavior as `-S` option.

2.33.5 Operands

The `pbs_rstat` command accepts one or more *reservation ID* 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> specifies a reservation at a server other than the default server.

2.33.6 See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide, "pbs_rsub" on page 96, "pbs_rdel" on page 91, "Reservation Attributes" on page 295

2.34 pbs_rsub

Creates a PBS advance or standing reservation

2.34.1 Synopsis

pbs_rsub [-D <duration>] [-E <end time>] [-g <group list>] [-G <auth group list>] [-H <auth host list>] [-I <block time>] [-m <mail events>] [-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

2.34.2 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.

2.34.2.1 Requirements

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

If you want to run jobs in the reservation that will request exclusive placement, you must create the reservation with exclusive placement via `-l place=excl`.

2.34.3 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 in this list, along with an ordered set of rules, to associate a group name with the reservation. The reservation creator's primary group is automatically added to this list.

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

Refer to the Authorized_Groups reservation attribute in section 6.8, "Reservation Attributes", on page 295.

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 in section 6.8, "Reservation Attributes", on page 295.

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

Default: All hosts are authorized to submit jobs

-I <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 a 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 can contain the following elements, in any order:

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

where

arrangement

Whether this reservation chunk is willing to share this vnode or host with other chunks from this reservation. One of *free* | *pack* | *scatter* | *vscatter*

sharing

Whether this reservation chunk is willing to share this vnode or host with other reservations or jobs. One of *excl | shared | exclhost*

grouping

Whether the chunks from this reservation should be placed on vnodes that all have the same value for a resource. Can have only one instance of *group=<resource name>*

free

Place reservation on any vnode(s).

pack

All chunks are taken from one host.

scatter

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

vscatter

Only one chunk is taken from any vnode. Each chunk must fit on a vnode.

excl

Only this reservation uses the vnodes chosen.

shared

This reservation can share the vnodes chosen.

exclhost

The entire host is allocated to the reservation.

group=<resource name>

Chunks are grouped according to the specified resource. All vnodes in the group must have a common value for *resource*, which can be either the built-in resource host or a custom vnode-level resource.

Resource name must be a string or a string array.

If you want to run jobs in the reservation that will request exclusive placement, you must create the reservation with exclusive placement via `-l place=excl`.

The place statement cannot start with a colon. Colons are delimiters; use them only to separate parts of a place statement, unless they are quoted inside resource values.

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

See section 6.10, “Vnode Attributes”, on page 311.

-l <resource request>

The *resource request* specifies the resources required for the reservation. These resources are 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.

Requesting resources in chunks:

-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> ...]

Requesting job-wide resources:

-l <resource name>=<value>[,<resource name>=<value> ...]

-m <mail events>

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 one of the following:

- Any combination of "*a*", "*b*", "*c*" or "*e*"
- The single character "*n*"

The following table lists the sub-options to the -m option:

Table 2-6: Sub-options 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 events>` option.

Format: *<username>[@<hostname>][,<username>[@<hostname>]...]*

Default: Reservation owner.

-N <reservation name>

Specifies a name for the reservation.

Format: *Reservation Name*. See "Reservation Name" on page 347.

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 *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.

Format: *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>[T<HHMMSS>]

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.
- Spaces are not allowed.

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 duration 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 submit a reservation with the specification `-R 1110` at 11:15 a.m., it is interpreted 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>

Not used. Comma-separated list of user names.

Format: *<username>[@<hostname>][,<username>[@<hostname>] ...]*

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 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 the Authorized_Users and Authorized_Groups reservation attributes 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 in section 6.8, "Reservation Attributes", on page 295.

Format: *[+|-]<username>@<hostname>[,[+|-]<username>@<hostname>...]*

Default: Job owner only.

-W <attribute value list>

This allows you to define other attributes for the reservation.

Supported attributes:

qmove=<job ID> [-I -<timeout>]

Converts a normal job designated by *job ID* 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.

You can use the `-I` option to specify a timeout for the conversion. If you use the `qmove` option to convert a job to a reservation, 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, you must 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=<job ID> option.

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

--version

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

2.34.4 Output

The `pbs_rsub` command returns the reservation identifier.

For an advance reservation, this has the form

R<NNNN>.<server name>

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

For a standing reservation, this has the form

S<NNNN>.<server name>

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

2.34.5 See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide, "pbs_rstat" on page 94, "pbs_rdel" on page 91, "Reservation Attributes" on page 295

2.35 pbs_sched

Runs a PBS scheduler

2.35.1 Synopsis

pbs_sched [-a <alarm>] [-c <clientsfile>] [-d <home dir>] [-I <scheduler name>] [-L <logfile>] [-n] [-N] [-p <output file>] [-R <port number>] [-S <port number>]

pbs_sched --version

2.35.2 Description

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

2.35.2.1 Required Permission

`pbs_sched` must be executed with root permission on Linux and Admin privilege on Windows.

2.35.3 Options to `pbs_sched`

-a <alarm>

Deprecated. Overwrites 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 this 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 this scheduler. If *clientsfile* cannot be opened, this scheduler aborts. Path can be absolute or relative. If relative, it is relative to `PBS_HOME/sched_priv/`.

-d <home dir>

The directory in which this scheduler will run.

Default: `PBS_HOME/sched_priv`.

-I <scheduler name>

Name of scheduler to start. Required when starting a multisched.

-L <logfile>

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

See the -d option.

Default: This scheduler opens a file named for the current date in the `PBS_HOME/sched_log` directory.

-n

Tells this scheduler to not restart itself if it receives a sigsegv or a sigbus. A scheduler by default restarts itself if it receives either of these two signals more than five minutes after starting. A scheduler does not restart itself if it receives either one within five minutes of starting.

-N

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

-p <output file>

Any output which is written to standard out or standard error is written to *output file*. The pathname can be absolute or relative, in which case it is relative to `PBS_HOME/sched_priv`.

See the `-d` option.

Default: `PBS_HOME/sched_priv/sched_out`

-R <port number>

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 number>

The port for this scheduler to use.

Required when starting a multisched.

For the default scheduler, if this option is not specified, the default port is taken from PBS_SCHEDULER_SERVICE_PORT, in `pbs.conf`.

Default value for default scheduler: *15004*

Default value for multisched: none

--version

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

2.35.4 Signal Handling

All signals are ignored until the end of the cycle. Most signals are handled in the standard UNIX fashion.

SIGHUP

This scheduler closes and reopens its log file and rereads its configuration file if one exists.

SIGALRM, SIGBUS, etc.

Ignored until end of scheduling cycle. This scheduler quits.

SIGINT and SIGTERM

This scheduler closes its log file and shuts down.

All other signals have the default action installed.

2.35.5 Exit Status

Zero

Upon normal termination

2.35.6 See Also

The PBS Professional Administrator's Guide

2.36 pbs_server

Starts a PBS batch server

2.36.1 Synopsis

pbs_server [-A <acctfile>] [-a <active>] [-C] [-d <config path>] [-e <mask>] [-F <delay>] [-L <logfile>] [-M <MoM port>] [-N] [-p <port number>] [-R <MoM RPP port>] [-S <default scheduler port>] [-s <replacement string>] [-t <restart type>]

pbs_server --version

2.36.2 Description

The `pbs_server` command starts a batch server on the local host. Typically, this command is in a local boot file such as `/etc/rc.local`. If the batch server is already running, `pbs_server` exits with an error.

2.36.2.1 Required Permission

To ensure that the `pbs_server` command is not runnable by the general user community, the server runs only if its real and effective UID is zero. You must be root on Linux or Admin on Windows.

2.36.3 Options to `pbs_server`

-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.

-a <value>

When *True*, the server is in state "*active*" and the default scheduler is called to schedule jobs. When *False*, the server is in state "*idle*" and the default scheduler is not called to schedule jobs. Sets the server's scheduling attribute. If this option is not specified, the server uses the previously specified *value* for the scheduling attribute.

Format: *Boolean*

-C

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

-d <config path>

Specifies the absolute path to the directory containing the server configuration files, `PBS_HOME`. A host may have multiple servers. Each server must have a different configuration directory. The default configuration directory is specified in $PBS_HOME, and is typically `/var/spool/pbs`.

-e <mask>

Specifies a log event mask to be used when logging. See "log_events" in section 6.6, "Server Attributes", on page 273.

-F <delay>

Specifies the number of seconds that the secondary server should wait before taking over when it believes the primary server is down. If the number 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 the absolute path name for the log file. If not specified, the file is named for the current date in the PBS_HOME/server_logs directory. PBS_HOME is specified in the $PBS_HOME environment variable or in /etc/pbs.conf; see the -d option.

-M <MoM port>

Specifies the hostname and/or port number on which the server should connect to MoM. The option argument, *MoM port*, uses the syntax:

[<hostname>][:<port number>]

If *hostname* not specified, the local host is assumed.

If *port number* is not specified, the default port is assumed.

See the -M option in section 2.23, “pbs_mom”, on page 71.

Default: *15002*

-N

The server runs in standalone mode. On Windows, it does not register as a Windows service. On other platforms, MoM does not detach from the current session.

-p <port number>

Specifies the port number on which the server is to 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 testing with multiple batch systems on a single host.

Format: Integer port number

Default: *15001*

-R <MoM RPP port>

Specifies the port number on which the server should query the up/down status of MoM. See the -R option in section 2.23, “pbs_mom”, on page 71.

Default: *15003*

-S <default scheduler port>

Specifies the port number to which the server should connect when contacting the default scheduler. The option argument, *default scheduler port*, uses the syntax:

[<hostname>][:<port number>]

If *hostname* not specified, the local host is assumed. If *port number* is not specified, the default port is assumed.

Default: *15004*

-s <replacement string>

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

-t <restart type>

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

cold

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

create

The server discards any existing configuration files: server, nodes, queues, and jobs, and initializes configuration files to the default values. The default scheduler is idled (scheduling is set to *False*). Any multischeds are deleted.

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 is 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, normal scheduling takes place for all remaining queued jobs. All other jobs are retained in their current state.

If a job cannot be restarted immediately because of a missing resource, such as a vnode being down, the server attempts 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 default scheduler typically chooses new jobs for 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.

2.36.4 Files

`$PBS_HOME/server_priv`
Default directory for configuration files.

`$PBS_HOME/server_logs`
Directory for log files recorded by the server.

2.36.5 Signal Handling for pbs_server

When it receives the following signals, the server performs the following actions:

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 where SIGSHUTDN is defined, causes an orderly "*quick*" shutdown of the server.

SIGPIPE, SIGUSR1, SIGUSR2
These signals are ignored.

All other signals have their default behavior installed.

2.36.6 Diagnostic Messages

The server records 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.

2.36.7 Stopping the PBS Server

2.36.7.1 Stopping the Server on Linux

Use the `qterm` command (see section 2.61, “qterm”, on page 226):

```
qterm
```

or send a SIGTERM:

```
kill <server PID>
```

2.36.7.2 Stopping the Server on Windows

If you’re running “`pbs_server -N`” for a standalone mode server, use

```
<cntrl>-<break>
```

2.36.8 Exit Status

Zero
: When the server has run in the background and then exits

Greater than zero
: If the server daemon fails to begin batch operation

2.36.9 See Also

The PBS Professional Administrator’s Guide

2.37 pbs_snapshot

Linux only. Captures PBS data to be used for diagnostics

2.37.1 Synopsis

pbs_snapshot -h, --help

pbs_snapshot -o <output directory path> [--accounting-logs=<number of days>] [--additional-hosts=<hostname list>] [--daemon-logs=<number of days>] [-H <server host>] [-l <log level>] [--map=<file path>] [--obfuscate] [--with-sudo]

pbs_snapshot --version

2.37.2 Description

You use `pbs_snapshot` to capture PBS data for diagnostics. This tool is written in Python and uses PTL libraries, including `PBSSnapUtils`, to extract the data. You can optionally anonymize the PBS data. The `pbs_snapshot` command captures data from all multischeds. The command detects which daemon or daemons are running on the host where it is collecting information, and captures daemon and system data accordingly. If no PBS daemons are running, the command collects system information. The output tarball contains information about the host designated via the `-H` option, or if that is not specified, the local host. If you specify additional hosts, the command creates a tarball for each additional host and includes it as a sub-tarball in the output.

2.37.2.1 Required Privilege

The `pbs_snapshot` command allows you to use the `sudo` infrastructure provided by the PTL framework to capture root-owned information via `--with-sudo`. All other information is collected as a normal user. If you need to run `pbs_snapshot` as a non-prvileged user, and without using the PTL `--with-sudo` infrastructure, you must be root if you want root-owned information to be collected.

2.37.2.2 Restrictions

The `pbs_snapshot` command is not available on Windows.

2.37.3 Options to pbs_snapshot

--accounting-logs=<number of days>

Specifies number of days of accounting logs to be collected; this count includes the current day.

Value of *number of days* must be >=0:

- If number of days is 0, no logs are captured.
- If number of days is 1, only the logs for the current day are captured.

Default: `pbs_snapshot` collects 30 days of accounting logs

--additional-hosts=<hostname list>

Specifies that pbs_snapshot should gather data from the specified list of additional hosts. Launches the pbs_snapshot command on each specified host, creates a tarball there named *<hostname>_snapshot.tgz*, and includes it as a sub-tarball in the output for the main output. If you use the --with-sudo option, each launched copy uses that option as well.

The command does not query the server when it runs at a non-server host.

The command collects a full snapshot, including the following information:

- Daemon logs, for the number of days of logs being captured, specified via the --daemon-logs=<number of days> option
- The PBS_HOME/<daemon>_priv directory
- Accounting logs if server daemon runs on host
- System information

Format for *hostname list* is a comma-separated list of one or more hostnames:

<hostname>[, <hostname> ...]

--daemon-logs=<number of days>

Specifies number of days of daemon logs to be collected; this count includes the current day.

Value of *number of days* must be >=0:

- If number of days is 0, no logs are captured.
- If number of days is 1, only the logs for the current day are captured.

Default: pbs_snapshot collects 5 days of daemon logs

-h, --help

Prints usage and exits.

-H <hostname>

Specifies hostname for host whose retrieved data is to be at the top level in the output tarball. If not specified, pbs_snapshot puts data for the local host at the top level in the output tarball.

-l <log level>

Specifies level at which pbs_snapshot writes its log. The log file is pbs_snapshot.log, in the output directory path specified using the -o <output directory path> option.

Valid values, from most comprehensive to least: *DEBUG2, DEBUG, INFOCLI2*, *INFOCLI, INFO*, *WARNING*, *ERROR*, *FATAL*

Default: *INFOCLI2*

--map=<file path>

Specifies path for file containing obfuscation map, which is a <key>:<value> pair-mapping of obfuscated data. Path can be absolute or relative to current working directory.

Default: pbs_snapshot writes its obfuscation map in a file called "obfuscate.map" in the location specified via the -o <output directory path> option.

Can only be used with the --obfuscate option.

--obfuscate

Obfuscates (anonymizes) or deletes sensitive PBS data captured by pbs_snapshot.

- Obfuscates the following data: euser, egroup, project, Account_Name, operators, managers, group_list, Mail_Users, User_List, server_host, acl_groups, acl_users, acl_resv_groups, acl_resv_users, sched_host, acl_resv_hosts, acl_hosts, Job_Owner, exec_host, Host, Mom, resources_available.host, resources_available.vnode
- Deletes the following data: Variable_List, Error_Path, Output_Path, mail_from, Mail_Points, Job_Name, jobdir, Submit_arguments, Shell_Path_List

--version

The pbs_snapshot command prints its PBS version information and exits. Can only be used alone.

--with-sudo

Uses the PTL sudo infrastructure in order capture root-owned information via sudo. (Information not owned by root is captured using normal privilege, not root privilege.) With this option, you do not need to prefix your pbs_snapshot command with sudo, and you do not need root privilege.

2.37.4 Arguments to pbs_snapshot

-o <output directory path>

Path to directory where pbs_snapshot writes its output tarball. Required. Path can be absolute or relative to current working directory.

For example, if you specify "-o /temp", pbs_snapshot writes "/temp/snapshot_<timestamp>.tgz".

The output directory path must already exist.

2.37.5 Output

2.37.5.1 Output Location

You must use the -o <output directory path> option to specify the directory where pbs_snapshot writes its output. The path can be absolute or relative to current working directory. The output directory must already exist. As an example, if you specify "-o /temp", pbs_snapshot writes "/temp/snapshot_<timestamp>.tgz".

2.37.5.2 Output Contents

The pbs_snapshot command writes the output for the local host and each specified remote host as a tarball. Tarballs for remote hosts are included in the main tarball.

The command captures JSON output from qstat -f -F json and pbsnodes -av -F json.

The main tarball contains the following directory structure, files, and tarballs:

Table 2-7: Contents of Snapshot

Directory or File	Directory Contents	Description
server/		
	qstat_B.out	Output of qstat -B
	qstat_Bf.out	Output of qstat -Bf
	qmgr_ps.out	Output of qmgr print server
	qstat_Q.out	Output of qstat -Q
	qstat_Qf.out	Output of qstat -Qf
	qmgr_pr.out	Output of qmgr print resource
server_priv/	Copy of the PBS_HOME/server_priv directory. Core files are captured separately; see core_file_bt/.	
	accounting/	Accounting logs from PBS_HOME/server_priv/accounting/ directory for the number of days specified via --accounting-logs option
server_logs/	Server logs from the PBS_HOME/server_logs directory for the number of days specified via --daemon-logs option	
job/		
	qstat.out	Output of qstat
	qstat_f.out	Output of qstat -f
	qstat_f_F_json.out	Output of qstat -f -F json
	qstat_t.out	Output of qstat -t
	qstat_tf.out	Output of qstat -tf
	qstat_x.out	Output of qstat -x
	qstat_xf.out	Output of qstat -xf
	qstat_ns.out	Output of qstat -ns
	qstat_fx_F_dsv.out	Output of qstat -fx -F dsv
	qstat_f_F_dsv.out	Output of qstat -f -F dsv

Table 2-7: Contents of Snapshot

Directory or File	Directory Contents	Description
node/		
	pbsnodes_va.out	Output of pbsnodes -va
	pbsnodes_a.out	Output of pbsnodes -a
	pbsnodes_avSj.out	Output of pbsnodes -avSj
	pbsnodes_aSj.out	Output of pbsnodes -aSj
	pbsnodes_avS.out	Output of pbsnodes -avS
	pbsnodes_aS.out	Output of pbsnodes -aS
	pbsnodes_aFdsv.out	Output of pbsnodes -aF dsv
	pbsnodes_avFdsv.out	Output of pbsnodes -avF dsv
	pbsnodes_avFjson.out	Output of pbsnodes -avF json
	qmgr_pn_default.out	Output of qmgr print node @default
mom_priv/	Copy of the PBS_HOME/mom_priv directory. Core files are captured separately; see core_file_bt/.	
mom_logs/	MoM logs from the PBS_HOME/mom_logs directory for the number of days specified via --daemon-logs option	
comm_logs/	Comm logs from the PBS_HOME/comm_logs directory for the number of days specified via --daemon-logs option	
sched_priv/	Copy of the PBS_HOME/sched_priv directory, with all files. Core files are not captured; see core_file_bt/.	
sched_logs/	Scheduler logs from the PBS_HOME/sched_log directory for the number of days specified via --daemon-logs option	
sched_priv_<multi-sched name>/	Copy of the PBS_HOME/sched_priv_<multisched name> directory, with all files. Core files are not captured; see core_file_bt/.	
sched_logs_<multi-sched name>/	Multisched logs from the PBS_HOME/sched_log_<multisched name> directory for the number of days specified via --daemon-logs option	
reservation/		
	pbs_rstat_f.out	Output of pbs_rstat -f
	pbs_rstat.out	Output of pbs_rstat
scheduler/		
	qmgr_lsched.out	Output of qmgr list sched
hook/		
	qmgr_ph_default.out	Output of qmgr print hook @default
	qmgr_lpbshook.out	Output of qmgr list pbshook

Table 2-7: Contents of Snapshot

Directory or File	Directory Contents	Description
datastore/		
	pg_log/	Copy of the PBS_HOME/datastore/pg_log directory for the number of days specified via `--daemon-logs` option
core_file_bt/	Stack backtrace from core files	
	sched_priv/	Files containing the output of `thread apply all backtrace full` on all core files captured from PBS_HOME/sched_priv
	sched_priv_<multisched name>	Files containing the output of `thread apply all backtrace full` on all core files captured from PBS_HOME/sched_priv_<multisched name>
	server_priv/	Files containing the output of `thread apply all backtrace full` on all core files captured from PBS_HOME/server_priv
	mom_priv/	Files containing the output of `thread apply all backtrace full` on all core files captured from PBS_HOME/mom_priv
	misc/	Files containing the output of `thread apply all backtrace full` on any other core files found inside PBS_HOME
system/		
	pbs_probe_v.out	Output of `pbs_probe -v`
	pbs_hostn_v.out	Output of `pbs_hostn -v $(hostname)`
	pbs_environment	Copy of PBS_HOME/pbs_environment file
	os_info	Information about the OS
	process_info	List of processes running on the system when the snapshot was taken. Output of `ps -aux | grep [p]bs` on Linux systems, or `tasklist /v` on Windows systems
	ps_leaf.out	Output of `ps -leaf`. Linux only.
	lsof_pbs.out	Output of `lsof | grep [p]bs`. Linux only.
	etc_hosts	Copy of /etc/hosts file. Linux only.
	etc_nsswitch_conf	Copy of /etc/nsswitch.conf file. Linux only.
	vmstat.out	Output of the command `vmstat`. Linux only.
	df_h.out	Output of the command `df -h`. Linux only.
	dmesg.out	Output of the `dmesg` command. Linux only.
pbs.conf	Copy of the pbs.conf file on the server host	
ctime	Contains the time in seconds since epoch when the snapshot was taken	
pbs_snapshot.log	Log messages written by `pbs_snapshot`	
<remote host-name>.tgz	Tarball of output from running the `pbs_snapshot` command at a remote host	

2.37.6 Examples

pbs_snapshot -o /tmp

Writes a snapshot to /temp/snapshot_<timestamp>.tgz that includes 30 days of accounting logs and 5 days of daemon logs from the server host.

pbs_snapshot --daemon-logs=1 --accounting-logs=1 -o /tmp --obfuscate --map=mapfile.txt

Writes a snapshot to /temp/snapshot_<timestamp>.tgz that includes 1 day of accounting and daemon logs. Obfuscates the data and stores the data mapping in the map file named "mapfile.txt".

2.38 pbs_tclsh

TCL shell with TCL-wrapped PBS API

2.38.1 Synopsis

pbs_tclsh

pbs_tclsh --version

2.38.2 Description

The `pbs_tclsh` command starts a version of the TCL shell which includes wrapped versions of the PBS external API. The PBS TCL API is documented in "TCL/tk Interface" on page 79 in the PBS Professional Programmer's Guide.

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>
```

2.38.2.1 Required Permission

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.

2.38.3 Options

--version

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

2.38.4 Standard Error

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

2.38.5 See Also

The PBS Professional Administrator's Guide, the PBS Programmer's Guide, "pbs_wish" on page 121

2.39 pbs_tmrsh

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

2.39.1 Synopsis

pbs_tmrsh <hostname> [-l <username>] [-n] <command> [<args> ...]

pbs_tmrsh --version

2.39.2 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.

Running "`pbs_tmrsh` <hostname> `<command>`" causes a PBS task to be started on *hostname* running *command*.

2.39.2.1 Requirements for Environment Variables

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` should contain the environment variable PATH in which to search for the program executable. This applies to both Windows and Linux. It is expected that a full path will be specified for the *command* and the PATH variable will not be needed.

2.39.3 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
: 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.

2.39.4 Operands

command
: Specifies command to be run as a PBS task.

hostname
: Specifies host on which to run PBS task. The *hostname* may be specified in IP-dot-address form.

2.39.5 Output and Error

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

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

2.39.6 Exit Status

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

2.39.7 See Also

The PBS Professional Administrator's Guide, "pbs_attach" on page 55, "TM Library Routines", on page 69 of the PBS Professional Programmer's Guide

2.40 pbs_topologyinfo

Reports topological information used for licensing purposes

2.40.1 Synopsis

pbs_topologyinfo (-a | --all) [(-l | --license) | (-s | --sockets)]

pbs_topologyinfo (-l | --license) <vnode name> [<vnode name> ...]

pbs_topologyinfo (-s | --sockets) <vnode name> [<vnode name> ...]

pbs_topologyinfo -h | --help

2.40.2 Description

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

This command must be run on the server host.

2.40.2.1 Usage

`pbs_topologyinfo -al` reports number of node licenses needed for all vnodes.

`pbs_topologyinfo -l <vnode name>` reports number of node licenses needed for *vnode name*.

`pbs_topologyinfo -as` reports socket counts for all vnodes that have reported sockets.

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

2.40.2.2 Prerequisites

Before you use this command, the server and MoMs must be configured so that they can contact each other, and must have been run.

2.40.2.3 Required Privilege for `pbs_topologyinfo`

This command can be run only by root or Admin on Windows.

2.40.3 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.

-h, --help

Prints usage and exits.

-l, --license [<vnode name(s)>]

Reports number of node licenses required. If you specify *vnode name(s)*, the command reports node licenses needed for the specified vnode(s) only.

-s, --sockets [<vnode name(s)>]

Reports derived socket counts. If you specify *vnode name(s)*, the command reports socket count information for the specified vnode(s) only.

(no options)

Does not report any information.

2.40.4 Errors

If you specify an invalid vnode name, the command prints a message to standard error.

2.40.5 Operands

vnode name [<vnode name> ...]

Name(s) of vnode(s) about which to report.

2.40.6 Exit Status

0

Success

1

Any error following successful command line processing

2.40.7 Standard Error

If an invalid vnode name is specified, a message is printed to standard error.

2.40.8 See Also

The PBS Professional Administrator's Guide

2.41 pbs_wish

TK window shell with TCL-wrapped PBS API

2.41.1 Synopsis

pbs_wish

pbs_wish --version

2.41.2 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 "TCL/tk Interface" on page 79 in the PBS Professional Programmer's Guide.

2.41.3 Options

--version

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

2.41.4 Standard Error

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

2.41.5 See Also

The PBS Professional Administrator's Guide, "pbs_tclsh" on page 116

2.42 printjob

Prints job information

2.42.1 Synopsis

printjob [-a | -s] <job ID>

printjob [-a] <file path> [<file path>...]

printjob --version

2.42.2 Description

Prints job information. This command is mainly useful for troubleshooting, as during normal operation, the "qstat" command is the preferred method for displaying job-specific data and attributes. The server and MoM do not have to be running to execute this command.

2.42.2.1 Usage

For a running job, you can run this command at any host using a job ID, and you can run this command at any execution host where the job is running using a .JB file path.

For a finished job, if job history is enabled, you can run this command at the server using the job ID.

When querying the server, you must use the job ID, and the data service must be running.

Results will vary depending on whether you use the job ID or a .JB file, and on which execution host you query with a .JB file.

2.42.2.2 Permissions

In order to execute `printjob`, you must have root or Windows Administrator privilege.

2.42.3 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. Can be used at server or primary execution host. Cannot be used with `-a` option. Must be used with a job ID.

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

2.42.4 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 "Job ID, Job Identifier" on page 345. Data service must be running.

2.42.5 Standard Error

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

2.42.6 Exit Status

Zero

Upon successful processing of all operands presented

Greater than zero

If the `printjob` command fails to process any operand

2.42.7 See Also

The PBS Professional Administrator's Guide, "qstat" on page 192

2.43 qalter

Alters a PBS job

2.43.1 Synopsis

qalter [-a <date and time>] [-A <account string>] [-c <checkpoint spec>] [-e <error path>] [-h <hold list>] [-j <join>] [-k <discard>] [-l <resource list>] [-m <mail events>] [-M <user list>] [-N <name>] [-o <output path>] [-p <priority>] [-P <project>] [-r <y|n>] [-R <remove options>] [-S <path list>] [-u <user list>] [-W <additional attributes>] <job ID> [<job ID> ...]

qalter --version

2.43.2 Description

The `qalter` command is used to alter one or more PBS batch jobs. Each of certain job attributes can be modified using the `qalter` option for that attribute. You can alter a job or a job array, but not a subjob or range of subjobs.

2.43.2.1 Required Privilege

A non-privileged user can alter their own jobs, whether they are queued or running. An Operator or Manager can alter any job, whether it is queued or running.

A non-privileged user can only lower resource requests. An Operator or Manager can raise or lower resource requests.

2.43.2.2 Modifying Resources and Job Placement

A Manager or Operator may lower or raise requested resource limits, except for per-process limits such as pcput and pmem, because these are set when the process starts, and enforced by the kernel. A non-privileged user can only lower resource requests.

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.

If a job is running, the only resources that can be modified are cput, walltime, min_walltime, and max_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.

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.

Requesting resources includes setting limits on resource usage and controlling how the job is placed on vnodes.

See Chapter 5, "List of Built-in Resources", on page 255.

2.43.2.2.i Syntax for Modifying Resources and Job Placement

Resources are modified 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> ...]

2.43.2.2.ii The Place Statement

You choose how your chunks are placed using the *place statement*. The *place statement* can contain the following elements, in any order:

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

where

arrangement

Whether this chunk is willing to share this vnode or host with other chunks from the same job. One of *free* | *pack* | *scatter* | *vscatter*

sharing

Whether this this chunk is willing to share this vnode or host with other jobs. One of *excl* | *shared* | *exclhost*

grouping

Whether the chunks from this job should be placed on vnodes that all have the same value for a resource. Can have only one instance of *group=<resource name>*

free

Place job on any vnode(s).

pack

All chunks are taken from one host.

scatter

Only one chunk with any MPI processes is taken from a host. A chunk with no MPI processes may be taken from the same vnode 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 name>

Chunks are grouped according to a resource. All vnodes in the group must have a common value for *resource*, which can be either the built-in resource host or a custom vnode-level resource. The *resource name* must be a string or a string array.

The *place statement* cannot begin with a colon. Colons are delimiters; use them only to separate parts of a place statement, unless they are quoted inside resource values.

Note that vnodes can have sharing attributes that override job placement requests. See section 6.10, “Vnode Attributes”, on page 311.

For more on resources, resource requests, usage limits, and job placement, see "Using PBS Resources" on page 227 in the PBS Professional Administrator’s Guide and "Allocating Resources & Placing Jobs", on page 51 of the PBS Professional User’s Guide.

2.43.2.3 Modifying Attributes

The user alters job attributes via 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.

2.43.2.4 Caveats and Restrictions for Altering Jobs

- When you lengthen the walltime of a running job, make sure that the new walltime will not interfere with any existing reservations etc.
- 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.

2.43.3 Options to `qalter`

-a <date and 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 and 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* 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.

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.

The argument *checkpoint spec* can take 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 greater than zero. 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 job's execution queue, 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 <error path>

Replaces the path to be used for the job's standard error stream. Sets job's Error_Path attribute to *error path*. Overridden by -k option.

Format: *[<hostname>:]<path>*

The *error path* is interpreted as follows:

path

If *path* is relative, 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* is absolute, it is taken to be an absolute path on the current host where the `qalter` command is executing.

hostname:path

If *path* is relative, it is taken to be relative to the user's home directory on the host named *hostname*.

If *path* is absolute, it is the absolute path on the host named *hostname*.

If *path* does not include a filename, the default filename is *<job ID>.ER*

If the -e option is not specified, PBS writes standard error to the default filename, which has this form:

<job name>.e<sequence 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.

If you use a UNC path, the hostname is optional. If you use a non-UNC path, the hostname is required.

-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 characters. The following table shows the holds and the privilege required to set each:

Table 2-8: Hold Types

Hold Type	Meaning	Who Can Set
u	*User*	*Job owner, Operator, Manager, administrator, root*
o	*Other*	*Operator, Manager, administrator, root*
s	*System*	*Manager, administrator, root, PBS (dependency)*
n	*None*	*Job owner, Operator, Manager, administrator, root*
p	*Bad password*	*Administrator, root*

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*.

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: *n*; not merged

The *join* argument can take the following values:

Table 2-9: 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.

-k <discard>

Changes whether and which of the standard output and standard error streams is left behind on the execution host, and whether they are written to their final destinations. Sets the job's Keep_Files attribute to *discard*. Overrides default path names for these streams. Overrides -o and -e options.

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: *n*; neither is retained, and files are not written to final destinations

The *discard* argument can take the following values:

Table 2-10: discard 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 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.
d	Output and/or error are written directly to their final destination. Overrides the action of leaving files behind on execution host.
n	Neither stream is retained.

-l <resource list>

Allows the user to change requested resources and job placement. Sets job's Resource_list attribute to *resource list*. 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. For syntax, see section 2.43.2.2.i, "Syntax for Modifying Resources and Job Placement", on page 124.

If a requested modification to a resource would exceed the server's or the job queue's limits, the resource request is rejected. Which resources can be altered is system-dependent.

If the job was submitted with an explicit "`-l select=`", vnode-level resources must be qaltered using the "`-l select=`" form. In this case a vnode-level resource *resource* cannot be qaltered with the "`-l <resource name>`" form.

The place statement cannot begin with a colon.

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 <resource name>`" 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 "Allocating Resources & Placing Jobs", on page 51 of the PBS Professional User's Guide.

-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 one of the following:

- The single character "*n*"
- Any combination of "*a*", "*b*", and "*e*", with optional "*j*"

The following table lists the sub-options to the `-m` option:

Table 2-11: Sub-options to m Option

Suboption	Meaning
n	No mail is sent.
a	Mail is sent when the job is aborted by PBS.
b	Mail is sent when the job begins execution.
e	Mail is sent when the job terminates.
j	Mail is sent for subjobs. Must be combined with one or more of a, b, or e options

Can be used with job arrays but not subjobs.

Format: *String*

Syntax: *n | [j](one or more of a, b, e)*

Example: -m ja

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: *<username>[@<hostname>][,<username>[@<hostname>],...]*

Default: Job owner.

-N <name>

Renames the job. Sets job's Job_Name attribute to *name*.

Format: *Job Name*. See "Job Name, Job Array Name" on page 345.

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 <output path>

Alters path to be used for the job's standard output stream. Sets job's Output_Path attribute to *output path*. Overridden by `-k` option.

Format: *[<hostname>:]<path>*

The *output path* is interpreted as follows:

path

If *path* is relative, it is taken to be relative to the current working directory of the command, where it is executing on the current host.

If *path* is absolute, it is taken to be an absolute path on the current host where the command is executing.

<hostname>:<path>

If *path* is relative, it is taken to be relative to the user's home directory on the host named *hostname*.

If *path* is absolute, it is the absolute path on the host named *hostname*.

If *path* does not include a filename, the default filename is:

<job ID>.OU

If the -o option is not specified, PBS writes standard output to the default filename, which has this form:

<job name>.o<sequence 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.

If you use a UNC path, the hostname is optional. If you use a non-UNC path, the hostname is required.

-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: *Project Name*; see "Project Name" on page 347

Default: "*_pbs_project_default*"

-r <y|n>

Changes whether the job is rerunnable. Sets job's Rerunable attribute to the argument. Does not affect how job is handled when the job is unable to begin execution.

See "qrerun" on page 173.

Format: Single character, "*y*" or "*n*".

y
: Job is rerunnable.

n
: Job is not rerunnable.

Default: "*y*".

Interactive jobs are not rerunnable. Job arrays are always rerunnable.

-R <remove options>

Changes whether standard output and/or standard error files are automatically removed upon job completion.

Sets the job's Remove_Files attribute to *remove options*. Overrides default path names for these streams. Overrides -o and -e options.

This attribute cannot be altered once the job has begun execution.

Default: unset; neither is removed

The *remove options* argument can take the following values:

Table 2-12: discard Argument Values

Option	Meaning
e	The standard error stream is removed (deleted) upon job completion
o	The standard output stream is removed (deleted) upon job completion
eo, oe	Both standard output and standard error streams are removed (deleted) upon job completion
unset	Neither stream is removed

-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 hostname. Only one path may be specified per named host. The path selected is the one whose hostname 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>[@<hostname>][,<path>@<hostname> ...]

If the path contains spaces, it must be quoted. For example:

```
qsub -S "C:Program Files\PBS Pro\bin\pbs_python.exe" <script name>
```

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 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 run under a username from this list. Sets job's User_List attribute to *user list*.

Only one username may be specified without a hostname. Only one username may be specified per named host. The server on which the job resides will select first the username whose hostname 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: *<username>[@<hostname>][,<username>@<hostname> ...]*

Default: Job owner (username on submit host)

-W <additional attributes>

Each sub-option to the `-W` option allows you to change a specific job attribute.

Format: *-W <attribute name> = <attribute value>[,<attribute name>=<attribute value>...]*

If white space occurs within the *additional attributes* argument, or the equal sign ("=") occurs within an *attribute value* string, that argument or string must be enclosed in single or double quotes. PBS supports setting the following attributes via 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:

<job ID>[:<job ID> ...]

The types and their argument lists 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 2.43.6.1, "Warning About Exit Status with csh", on page 136.

afternotok: <arg list>

This job may be scheduled for execution only after all jobs in *arg list* have terminated with errors. See section 2.43.6.1, "Warning About Exit Status with csh", on page 136.

afterany: <arg list>

This job may be scheduled for execution after all jobs in *arg list* have terminated, with or without errors. This job will not run if a job in the *arg list* was deleted without ever having been run.

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 2.43.6.1, "Warning About Exit Status with csh", on page 136.

beforenotok: <arg list>

If this job terminates execution with errors, jobs in *arg list* may begin. See section 2.43.6.1, "Warning About Exit Status with csh", on page 136.

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 depends is performed after the job is queued. If an error is detected, the new job is deleted by the server. Mail is 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=<group list>

Alters list of group names. Job will run under a group name from this list. Sets job's group_List attribute to *group list*.

Only one group name may be specified without a hostname. Only one group name may be specified per named host. The server on which the job resides will select first the group name whose hostname is the same as the server name. Failing that, the next selection is the group name with no specified hostname. The group names on the server and execution hosts must be the same.

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

Default: Login group name of job owner.

release_nodes_on_stageout=<value>

When set to *True*, all of the job's vnodes not on the primary execution host are released when stageout begins.

Cannot be used with vnodes managed by cpuset MoMs, (whose arch is *linux_cpuset*), or with vnodes tied to Cray X* series systems.

When cgroups is enabled and this is used with some but not all vnodes from one MoM, resources on those vnodes that are part of a cgroup are not released until the entire cgroup is released.

The job's stageout attribute must be set for the release_nodes_on_stageout attribute to take effect.

Format: *Boolean*

Default: *False*

run_count=<count>

Sets the number of times the server thinks it has run the job. Sets the job's run_count attribute to *count*. Can be altered while job is running. Job is held when the value of this attribute goes over *20*.

Format: Integer greater than or equal to zero

sandbox=<sandbox spec>

Changes which directory PBS uses for the job's staging and execution. Sets job's sandbox attribute to the value of *sandbox spec*.

Format: *String*

Allowed values for *sandbox spec*:

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.

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 list*s. On completion of the job, all staged-in and staged-out files and directories are removed from the execution host(s). A *path list* has the form:

<filespec>[,<filespec>]

where *filespec* is

<execution path>@<hostname>:<storage path>

regardless of the direction of the copy. The *execution 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 *execution path* from *storage path*.

The *storage 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.

If you use a UNC path, the hostname is optional. If you use a non-UNC path, the hostname is required.

umask=<mask value>

Alters the umask with which the job is started. Controls umask of job's standard output and standard error. Sets job's umask attribute to *mask value*.

Format: one to four digits; typically two

The following example allows group and world read of the job's output and error:

```
-W umask=33
```

Default: *077*

--version

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

2.43.4 Operands

The `qalter` command accepts a *job ID* list as its operand. The *job ID* list is a space-separated list of one or more job IDs for normal jobs or array jobs.

Subjobs and ranges of subjobs are not alterable.

Job IDs have the form:

<sequence number>[.<server name>][@<server name>]

<sequence number>[][.<server name>][@<server name>]

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

2.43.5 Standard Error

The `qalter` 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 failure

2.43.6.1 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.

2.43.7 See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide, "Job Attributes" on page 318, Chapter 5, "List of Built-in Resources", on page 255

2.44 qdel

Deletes PBS jobs

2.44.1 Synopsis

qdel [-x] [-Wforce| -Wsuppress_email=<N>] <job ID> [<job ID> ...]

qdel --version

2.44.2 Description

The `qdel` command deletes jobs in the order given, whether they are at the local server or at a remote server.

2.44.2.1 Usage

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.

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 "qsub" on page 207.

If the job is in the process of provisioning, it can be deleted only by using the `-W force` option.

2.44.2.2 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 section 6.6, "Server Attributes", on page 273.

2.44.2.3 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 standard out (.o) and standard error (.e) files.
4. Files that were staged in or out are deleted.
5. The job's temp directory is removed.
6. The job is removed from the MoM(s) and the server.

2.44.2.4 Required Privilege

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.

2.44.3 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.

If the server can contact the MoM, this option is ignored; the server allows the job to be deleted normally. If the server cannot contact the MoM or the job is in the *E* state, the server deletes its information about the job.

-Wsuppress_email=<N>

Sets limit on number of emails sent when deleting multiple jobs or subjobs.

- If *N* >= *1* and *N* or more *job IDs* 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.

2.44.4 Operands

The `qdel` command accepts one or more space-separated *job ID* operands. These operands can be job identifiers, job array identifiers, subjob identifiers, or subjob range identifiers.

Job IDs have the form:

<sequence number>[.<server name>][@<server name>]

Job arrays have the form:

<sequence number>[][.<server name>][@<server name>]

Subjobs have the form:

<sequence number>[<index>][.<server name>][@<server name>]

Ranges of subjobs have the form:

<sequence number>[<first>-<last>][.<server name>][@<server name>]

Job array identifiers must be enclosed in double quotes for some shells.

2.44.5 Standard Error

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

2.44.6 Exit Status

Zero
: Upon successful processing of input

Greater than zero
: Upon error

2.44.7 See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide

2.45 qdisable

Prevents a queue from accepting jobs

2.45.1 Synopsis

qdisable <destination> [<destination> ...]

qdisable --version

2.45.2 Description

The `qdisable` command prevents a queue from accepting batch jobs. Sets the value of the queue's enabled attribute to *False*. If the command is accepted, the queue no longer accepts Queue Job requests. Jobs already in the queue continue to be processed. You can use this to drain a queue of jobs.

2.45.2.1 Required Permission

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

2.45.3 Options

--version

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

2.45.4 Operands

The `qdisable` command accepts one or more space-separated *destination* operands. The operands take any of the following forms:

<queue name>

Prevents specified queue at default server from accepting jobs.

@<server name>

Prevents all queues at specified server from accepting jobs.

<queue name>@<server name>

Prevents specified queue at specified server from accepting jobs.

To prevent all queues at the default server from accepting jobs, use the `qmgr` command:

```
Qmgr: set queue @default enabled=false
```

2.45.5 Standard Error

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

2.45.6 Exit Status

Zero
: Upon successful processing of all the operands

Greater than zero
: If the `qdisable` command fails to process any operand

2.45.7 See Also

The PBS Professional Administrator's Guide, "qmgr" on page 146, "qenable" on page 142

2.46 qenable

Allows a queue to accept jobs

2.46.1 Synopsis

qenable <destination> [<destination> ...]

qenable --version

2.46.2 Description

The `qenable` command allows a queue to accept batch jobs. Sets the value of the queue's enabled attribute to *True*. If the command is accepted, the *destination* accepts Queue Job requests.

2.46.2.1 Required Privilege

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

2.46.3 Options

--version

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

2.46.4 Operands

The `qenable` command accepts one or more space-separated *destination* operands. The operands take any of the following forms:

<queue name>

Allows specified queue at default server to accept jobs.

@<server name>

Allows all queues at specified server to accept jobs.

<queue name>@<server name>

Allows specified queue at specified server to accept jobs.

To allow all queues at the default server to accept jobs, use the `qmgr` command:

```
Qmgr: set queue @default enabled=true
```

2.46.5 Standard Error

The `qenable` command writes 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 `qenable` command fails to process any operand

2.46.7 See Also

The PBS Professional Administrator's Guide, "qmgr" on page 146, "qdisable" on page 140

2.47 qhold

Holds PBS batch jobs

2.47.1 Synopsis

qhold [-h <hold list>] <job ID> [<job ID> ...]

qhold --version

2.47.2 Description

Places one or more holds on a job. A job that has a hold is not eligible for execution. Can be used on jobs and job arrays, but not on subjobs or ranges of subjobs.

If a job identified by *job ID* is in the queued, held, or waiting states, all that occurs is that the hold type is added to the job. The job is then put into the held state if it resides in an execution queue.

If the job is running, 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. If the administrator sets a *system* hold on a job with a dependency, when the dependency is satisfied, the job becomes eligible for execution.

If the job is in the process of provisioning, it cannot be held.

A hold on a job can be released by the PBS Administrator, root, a Manager, an Operator, or the job owner, when the job reaches the time set in its Execution_Time attribute, or when a dependency clears. See "qrls" on page 175.

2.47.2.1 Effect of Privilege on Behavior

The following table shows the holds and the privilege required to set each:

Table 2-13: Hold Types

Hold Type	Meaning	Who Can Set
u	*User*	*Job owner, Operator, Manager, PBS Administrator, root*
o	*Other*	*Operator, Manager, PBS Administrator, root*
s	*System*	*Manager, PBS Administrator, root, PBS (dependency)*
n	*No hold*	*Job owner, Operator, Manager, PBS Administrator, root*
p	*Bad password*	*PBS Administrator, root*

2.47.3 Options to qhold

(no options)

Same as -h u. Applies the *user* hold to the specified job(s).

-h <hold list>

Types of holds to be placed on the job(s).

The *hold list* argument is a string consisting of one or more of the letters "*u*", "*o*", or "*s*" in any combination, or one of the letters "*n*" or "*p*".

--version

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

2.47.4 Operands

The qhold command can be used on jobs and job arrays, but not on subjobs or ranges of subjobs. The qhold command accepts one or more *job IDs* in the form:

<sequence number>[.<server name>][@<server name>]

<sequence number>[][.<server name>][@<server name>]

Note that some shells require that you enclose a job array identifier in double quotes.

2.47.5 Standard Error

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

2.47.6 Exit Status

Zero

Upon successful processing of all operands

Greater than zero

If the qhold command fails to process any operand

2.47.7 See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide, "qrls" on page 175

2.48 qmgr

Administrator's command interface for managing PBS

2.48.1 Synopsis

At shell command line:

qmgr -c '<directive> [-a] [-e] [-n] [-z]'

qmgr -c 'help [<help option>]'

qmgr <return>

qmgr --version

In qmgr session:

<directive> [-a] [-e] [-n] [-z]

help <help option>

2.48.2 Description

The PBS manager command, `qmgr`, provides a command-line interface to parts of PBS. The `qmgr` command is used to create or delete queues, vnodes, resources, and hooks, to set or change vnode, queue, hook, server, or scheduler attributes and resources, and to view information about hooks, queues, vnodes, resource definitions, the server, and schedulers.

For a list of quick summaries of information about syntax, commands, attributes, operators, names, and values, type "`help`" or "`?`" at the `qmgr` prompt. See section 2.48.11, "Printing Usage Information", on page 165.

2.48.2.1 Modes of Operation

- When you type `qmgr -c '<directive>'`, `qmgr` performs its task and then exits.
- When you type `qmgr <return>`, `qmgr` starts a session and presents you with its command line prompt. The `qmgr` command then reads directives etc. from standard input; see section 2.48.4.1, "Directive Syntax", on page 148. You can edit the command line; see section 2.48.2.4, "Reusing and Editing the qmgr Command Line", on page 147.

  For a `qmgr` prompt, type:

  ```
  qmgr <return>
  ```

  You will see the `qmgr` prompt:

  ```
  Qmgr:
  ```

2.48.2.2 Required Privilege

The `qmgr` command requires different levels of privilege depending on the operation to be performed.

All users can list or print attributes except for hook attributes.

PBS Operator or Manager privilege is required in order to set or change vnode, queue, server, or scheduler attributes. PBS Manager privilege is required in order to create or delete queues, vnodes, and resources.

Under Linux, root privilege is required in order to create hooks, or 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 or resource definitions which were created to be invisible to users.

2.48.2.3 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.

2.48.2.4 Reusing and Editing the qmgr Command Line

You can reuse or edit qmgr command lines. The qmgr command maintains a history of commands entered, up to a maximum of 500. You can use the 'history' command to see a numbered list of commands, and the !<n> command to execute the line whose number is *n*. You must not put any spaces between the bang ("!") and the number. For example, to execute the 123rd command, type the following:

```
!123
```

You can see the last *m* commands by typing 'history m'. For example, to see the last 6 commands, type the following:

```
history 6
```

You can use the up and down arrows to navigate through the command history list, and the left and right arrows to navigate within a command line. Within a command line, you can use emacs commands to move forward and backward, and delete characters.

You can edit the qmgr command line using the backspace and delete keys, and you can insert characters anywhere in a command line.

History is maintained across qmgr sessions, so that if you start qmgr, then exit, then restart it, you can reuse your commands from the previous session. If you exit qmgr and then restart it, the command lines are renumbered.

If you enter the same command line more than once in a row, only one occurrence is recorded in the history. If you enter the same command line multiple times, but intersperse other command lines after each line, each occurrence is recorded.

Each user's history is unique to that user on that host.

In the case where an account runs concurrent sessions, the most recent logout of a session overwrites history from previous logouts. For example, if two people are both logged in as root and using qmgr, the second person to log out overwrites the history file.

2.48.2.4.i The qmgr History File

The qmgr command stores and retrieves its history. First, it tries to write its history in the ${HOME}/.pbs_qmgr_history file. If this file or directory location is not writable, the command stores its history in $PBS_HOME/spool/.pbs_qmgr_history_<user name>. If this file is also not writable, the following happens:

- The qmgr command prints error messages once at qmgr startup
- The qmgr command cannot provide history across qmgr sessions

2.48.3 Options to qmgr

The following table lists the options to qmgr:

Table 2-14: qmgr Options

Option	Action
<return>	Starts a qmgr session and presents user with qmgr prompt
-a	Aborts qmgr on any syntax errors or any requests rejected by a server.
-c '<directive>'	Executes a single command (*directive*) and exit qmgr. The *directive* must be enclosed in single or double quote marks, for example: qmgr -c "print server"
-c 'help [<help option>]'	Prints out usage information. See "Printing Usage Information" on page 165
-e	Echoes all commands to standard output
-n	No commands are executed; syntax checking only is performed
-z	No errors are written to standard error
--version	The qmgr command returns its PBS version information and exits. This option can only be used alone

2.48.4 Directives

A qmgr *directive* is a *command* together with the *object(s)* to be operated on, the *attribute(s)* belonging to the object that is to be changed, the *operator*, and the *value(s)* the *attribute(s)* will take. In the case of resources, you can set the *type* and/or *flag*(s).

2.48.4.1 Directive Syntax

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 ("\").

Comments begin with the "#" character and continue to the end of the line. Comments and blank lines are ignored by qmgr.

2.48.4.1.i Server, Scheduler, Queue, Vnode Directives

Syntax for operating on servers, schedulers, queues, and vnodes:

<command> <object type> [<object name(s)>] [<attribute> <operator> <value>[,<attribute> <operator> <value>,...]]

For information about attributes, see Chapter 6, "Attributes", on page 269.

2.48.4.1.ii Resource Directives

Syntax for operating on resources:

<command> <resource name> [<resource name> ...] [type = <type>][,flag = <flag(s)>]

For information about resources, see "Using PBS Resources" on page 227 in the PBS Professional Administrator's Guide and Chapter 5, "List of Built-in Resources", on page 255.

2.48.4.1.iii Hook-only Directives

The directives here apply only to hooks. Other directives apply to all objects such as queues, resources, hooks, etc.

Syntax for importing and exporting site-defined hooks:

"*import hook <hook name> application/x-python <content-encoding> (<input file> | -)*"

"*export hook <hook name> <content-type> <content-encoding>*" > *[<output file>]*

Syntax for importing site-defined hook configuration file:

"*import hook <hook name> application/x-config <content-encoding> (<input file> | -)*"

Syntax for importing built-in hook configuration file:

"*import pbshook <hook name> application/x-config <content-encoding> (<input file> | -)*"

2.48.4.2 Using Directives

You can use a *directive* from the shell command line or from within the `qmgr` session.

- To use a directive from the command line, enclose the command and its arguments in single or double quotes.

  qmgr -c '<command> <command arguments>'

  For example, to have `qmgr` print server information and exit:

  ```
  qmgr -c "print server"
  ```

- To use a directive from within the `qmgr` session, first start `qmgr`:

  ```
  qmgr <return>
  ```

  The `qmgr` session presents a `qmgr` prompt:

  ```
  Qmgr:
  ```

  At the `qmgr` prompt, enter the directive (a command and its arguments). For example, to enter the same "print server" directive:

  ```
  Qmgr: print server
  ```

2.48.4.3 Commands Used in Directives

Commands can be abbreviated to their minimum unambiguous form. Commands apply to all target objects unless explicitly limited. The following table lists the commands, briefly tells what they do, and gives a link to a full description:

Table 2-15: `qmgr` Commands Used in Directives

Command	Abbr.	Effect	Description
`active`	`a`	Specifies active objects	See section 2.48.6.1, "Making Objects Active", on page 153
`create`	`c`	Creates object	See section 2.48.6.2, "Creating Objects (Server, Scheduler, Vnode, Queue, Hook)", on page 154
`delete`	`d`	Deletes object	See section 2.48.6.3, "Deleting Objects", on page 154
`exit`		Exits (quits) the `qmgr` session	
`export`	`e`	Exports hook or hook configuration file	See section 2.48.10.6, "Exporting Hooks", on page 164 and section 2.48.10.5.ii, "Exporting Configuration Files", on page 163

Table 2-15: `qmgr` Commands Used in Directives

Command	Abbr.	Effect	Description
`help or ?`	`h, ?`	Prints usage to `stdout`	See section 2.48.11, "Printing Usage Information", on page 165
`import`	`i`	Imports hook or configuration file	See section 2.48.10.4, "Importing Hooks", on page 162 or section 2.48.10.5.i, "Importing Configuration Files", on page 163
`list`	`l`	Lists object attributes and their values	See section 2.48.8.1, "Listing Objects and Their Attributes", on page 159
`print`	`p`	Prints creation and configuration commands	See section 2.48.8.3, "Printing Creation and Configuration Commands", on page 161
`quit`	`q`	Quits (exits) the `qmgr` session	
`set`	`s`	Sets value of attribute	See section 2.48.7.1, "Setting Attribute and Resource Values", on page 155
`unset`	`u`	Unsets value of attribute	See section 2.48.7.2, "Unsetting Attribute and Resource Values", on page 156

2.48.5 Arguments to Directive Commands

2.48.5.1 Object Arguments to Directive Commands

The `qmgr` command can operate on objects (servers, schedulers, queues, vnodes, resources, hooks, and built-in hooks). Each of these can be abbreviated inside a directive. The following table lists the objects and their abbreviations:

Table 2-16: `qmgr` Objects

Object Name	Abbr.	Object	Can Be Created/Deleted By:	Can Be Modified By:
server	*s*	server	No one (created at installation)	Administrator, Operator, Manager
sched	*sc*	default scheduler	No one (created at installation)	Administrator, Operator, Manager
		multisched	Administrator, Manager	Administrator, Operator, Manager
queue	*q*	queue	Administrator, Operator, Manager	Administrator, Operator, Manager
node	*n*	vnode	Administrator, Operator, Manager	Administrator, Operator, Manager
resource	*r*	resource	Administrator, Manager	Administrator, Manager
hook	*h*	hook	Linux: root Windows: installation account	Linux: root Windows: installation account
pbshook	*p*	built-in hook	No one (created at installation)	Linux: root Windows: installation account

2.48.5.1.i Specifying Active 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. Server names have the following format:

<hostname>[:<port number>]

where *hostname* is the fully-qualified domain name of the host on which the server is running and *port number* is the port number to which to connect. If *port number* is not specified, the default port number, *15001*, is used.

- To specify the default server:

  @default
- To specify a named server:

  @<server name>
- To specify all active servers:

  @active

2.48.5.1.ii Using Lists of Object Names

In a qmgr directive, *object name(s)* is a list of one or more names of specific objects. The administrator specifies the name of an object when creating the object. The name list is in the form:

<object name>[@<server>][,<object name>[@<server>] ...]

where *server* is replaced in the directive with "*default*", "*active*", or the name of the server. The name list must conform to the following:

- There must be no space between the object name and the @ sign.
- Name lists must not contain white space between entries.
- All objects in a list must be of the same type.
- Node attributes cannot be used as vnode names.

2.48.5.1.iii Specifying Object Type and Name

You can specify objects in the following ways:

- To act on the active objects of the named type, at the active server:

  <object type>

  For example, to list all active vnodes, along with their attributes, at the active server:

  Qmgr: list node
- To act on the active objects of the named type, at a specified server:

  <object type> @<server name> (note space before @ sign)

  For example, to list all active vnodes at the default server, along with their attributes:

  Qmgr: list node @default

  For example, to print out all queues at the default server, along with their attributes:

  qmgr -c "print queue @default"
- To act on a specific named object:

  <object type> <object name>

  For example, to list Node1 and its attributes:

  Qmgr: list node Node1

To list queues workq, slowq, and fastq at the active server:

```
Qmgr: list queue workq,slowq,fastq
```

- To act on the named object at the specified server:

  <object type> <object name>@<server name>

  For example, to list Node1 at the default server, along with the attributes of Node1:

  ```
  Qmgr: list node Node1@default
  ```

  To 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
  ```

2.48.5.2 Operators in Directive Commands

In a `qmgr` directive, *operator* is the operation to be performed with the attribute and its value. Operators are listed here:

Table 2-17: Operators in Directive Commands

Operator	Effect
=	Sets the value of the attribute or resource. If the attribute or resource has an existing value, the current value is replaced with the new value.
+=	Increases the current value of the attribute or resource 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 or resource by the specified amount. When used for a string array, removes the first matching string.

Example 2-4: Set routing destination for queue Queue1 to be Dest1:

```
Qmgr: set queue route_destinations = Dest1
```

Example 2-5: Add new routing destination for queue Queue1:

```
Qmgr: set queue route_destinations += Dest2
```

Example 2-6: Remove new routing destination for queue Queue1:

```
Qmgr: set queue route_destinations -= Dest2
```

When setting numerical resource values, you can use only the equal sign (“=”).

2.48.5.3 Windows Requirements For Directive Arguments

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 Set-
   tings\pbsuser1\hook1.py"'
```

2.48.6 Operating on Objects (Server, Scheduler, Vnode, Queue, Hook)

2.48.6.1 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. 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. You can make any type of object active except for resources or hooks.

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 at a given server, that list of objects replaces any active objects of the same kind at that server. For example, if you have four queues at a particular server, 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 different 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.

2.48.6.1.i Using the `active` Command

- To make the named object(s) of the specified type active:

  active <object type> [<object name>[,<object name> ...]]

  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
  ```

- To make all object(s) of the specified type at the specified server active:

  active <object type> @<server name> (note space before @ sign)

  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
  ```

- To report which objects of the specified type are active:

  active <object type>

  The `qmgr` command prints a list of names of active objects of the specified type to `stdout`.

2.48.6.2 Creating Objects (Server, Scheduler, Vnode, Queue, Hook)

- To create one new object of the specified type for each name, and give it the specified name:

  create <object type> <object name>[,<object name> ...] [[<attribute> = <value>] [,<attribute> = <value>] ...]

  Can be used only with multischeds, queues, vnodes, resources, and hooks. Cannot be used with built-in hooks.

  For example, to create a multisched named multisched_1 at the active server:

  Qmgr: create sched multisched_1

  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

  To create a host-level consumable string resource named “foo”:

  Qmgr: qmgr -c "create resource foo type=string,flag=nh"

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.

To create multiple resources of the same type and flag, separate each resource name with a comma:

qmgr -c "create resource <resource>[,<resource> ...] type=<type>,flag=<flag(s)>"

2.48.6.2.i Examples of Creating Objects

Example 2-7: Create queue:

create queue fast priority=10,queue_type=e,enabled = true,max_running=0

Example 2-8: Create queue, set resources:

create queue little
set queue little resources_max.mem=8mw,resources_max.cput=10

2.48.6.3 Deleting Objects

- To delete the named object(s):

  delete <object type> <object name>[,<object name> ...]

  When you delete more than one object, do not put a space after a comma.

  Can be used only with queues, vnodes, resources, and hooks. Cannot be used with built-in 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
```

For example, to delete resource "foo" at the active server:

```
Qmgr: delete resource foo
```

- To delete the active objects of the specified type:

  delete <object type>

  For example, to delete the active queues:

  ```
  Qmgr: delete queue
  ```

- To delete the active objects of the specified type at the specified server:

  delete <object type> @<server name>

  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. To delete multiple resources, separate the resource names with commas.

For example:

```
Qmgr: delete resource r1,r2
```

You cannot delete a resource that is requested by a job or reservation, or that is set on a server, queue, or vnode.

2.48.7 Operating on Attributes and Resources

You can specify attributes and resources for named objects or for all objects of a type.

2.48.7.1 Setting Attribute and Resource Values

- To set the value of the specified attribute(s) for the named object(s):

  set <object type> <object name>[,<object name> ...] <attribute> = <value> [,<attribute> = <value> ...]

  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.

- To set the attribute value for all active objects when there are active objects of the type specified:

  set <object type> <attribute> = <value>

- To set the attribute value for all active objects at the specified server when there are active objects of the type specified:

  set <object type> @<server name> <attribute> = <value>

  For example, to set the amount of memory on a vnode:

  ```
  Qmgr: set node Vnode1 resources_available.mem = 2mb
  ```

If the attribute is one which describes a set of resources such as resources_available, resources_default, resources_max, resources_used, etc., the attribute is specified in the form:

<attribute name>.<resource name>

You can have spaces between *attribute=value* pairs.

2.48.7.1.i Examples of Setting Attribute Values

Example 2-9: Increase limit on queue:

```
set queue fast max_running +=2
```

Example 2-10: Set software resource on mynode:

```
set node mynode resources_available.software = "myapp=/tmp/foo"
```

Example 2-11: Set limit on queue:

```
set queue max_running = 10
```

Example 2-12: Set vnode offline:

```
set node state = "offline"
```

2.48.7.2 Unsetting Attribute and Resource Values

You can use the qmgr command to unset attributes of any object, except for the type attribute of a built-in hook.

- To unset the value of the specified attributes of the named object(s):
  unset <object type> <object name>[,<object name> ...] <attribute>[,<attribute>...]
- To unset the value of specified attributes of active objects:
  unset <object type> <attribute>[,<attribute>...]
- To unset the value of specified attributes of the named object:
  unset <object type> <object name> <attribute>[,<attribute>...]
- To unset the value of specified attributes of the named object:
  unset <object type> @<server name> <attribute>[,<attribute>...]

2.48.7.2.i Example of Unsetting Attribute Value

Example 2-13: Unset limit on queue

```
unset queue fast max_running
```

2.48.7.3 Caveats and Restrictions for Setting Attribute and Resource Values

- 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. For example:
  ```
  Qmgr: set node Vnode1 comment="Node will be taken offline Friday at 1:00 for memory
     upgrade."
  ```
- You can set or unset attribute values for only one type of object in each command.
- You can use the qmgr command to set attributes of any object, except for the type attribute of a built-in hook.
- You can have spaces between attribute names.
- Attribute and resource values must conform to the format for the attribute or resource type. Each attribute's type is listed in Chapter 6, "Attributes", on page 269. Each format is described in Chapter 7, "Formats", on page 343.
- 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 "Choosing Configuration Method" on page 44 in the PBS Professional Administrator's Guide.

2.48.7.4 Setting Resource Type and Flag(s)

You can use the qmgr command to set or unset the type and flag(s) for resources.

Resource types can be the following; see Chapter 7, "Formats", on page 343:

string

boolean

string_array

long

size

float

- To set a resource type:

  set resource <resource name> type = <type>

  Sets the type of the named resource to the specified *type*. For example:

  ```
  Qmgr: qmgr -c "set resource foo type=string_array"
  ```

2.48.7.4.i Resource Accumulation Flags

The resource accumulation flag for a resource can be one of the following:

Table 2-18: Resource Accumulation Flags

Flag	Meaning
(no flags)	Indicates a queue-level or server-level resource that is not consumable.
fh	The amount is consumable at the host level for only the first vnode allocated to the job (vnode with first task.) Must be consumable or time-based. Cannot be used with Boolean or string resources. This flag specifies that the resource is accumulated at the first vnode, meaning that the value of resources_assigned.<resource> is incremented only at the first vnode when a job is allocated this resource or when a reservation requesting this resource on this vnode starts.

Table 2-18: Resource Accumulation Flags

Flag	Meaning
h	Indicates a host-level resource. Used alone, means that the resource is not consumable. Required for any resource that will be used inside a select statement. This flag selects hardware. This flag indicates that the resource must be requested inside of a select statement. Example: for a Boolean resource named "green": `Qmgr: create resource green type=boolean,flag=h`
nh	The amount is consumable at the host level, for all vnodes assigned to the job. Must be consumable or time-based. Cannot be used with Boolean or string resources. This flag specifies that the resource is accumulated at the vnode level, meaning that the value of resources_assigned.<resource> is incremented at relevant vnodes when a job is allocated this resource or when a reservation requesting this resource on this vnode starts. This flag is not used with dynamic consumable resources. A scheduler will not oversubscribe dynamic consumable resources.
q	The amount is consumable at the queue and server level. When a job is assigned one unit of a resource with this flag, the resources_assigned.<resource> attribute at the server and any queue is incremented by one. Must be consumable or time-based. This flag specifies that the resource is accumulated at the queue and server level, meaning that the value of resources_assigned.<resource> is incremented at each queue and at the server when a job is allocated this resource. When a reservation starts, allocated resources are added to the server's resources_assigned attribute. This flag is not used with dynamic consumable resources. A scheduler will not oversubscribe dynamic consumable resources.

See "Resource Accumulation Flags" on page 255 in the PBS Professional Administrator's Guide.

2.48.7.4.ii Resource Permission Flags

The permission flag for a resource can be one of the following:

Table 2-19: Resource Permission Flags

Flag	Meaning
(no flag)	Users can view and request the resource, and `qalter` a resource request for this resource.
i	"Invisible". Users cannot view or request the resource. Users cannot `qalter` a resource request for this resource.
r	"Read only". Users can view the resource, but cannot request it or `qalter` a resource request for this resource.

See "Resource Permission Flags" on page 257 in the PBS Professional Administrator's Guide.

To set resource flags, concatenate the flags you want without spaces or commas.

- To set the flag(s) of the named resource to the specified *flag(s)*:
  set resource <resource name> flag=<flag(s)>

For example:

```
qmgr -c "set resource foo flag=nhi"
```

- To set both type and flag(s):

  set resource <resource name> type=<type>,flag=<flag(s)>

  Sets the type and flag(s) of the named resource to the specified *type* and *flag(s)*. For example:

  ```
  qmgr -c "set resource foo type=long,flag=nhi"
  ```

You can set multiple resources by separating the names with commas. For example:

```
qmgr -c "set resource r1,r2 type=long"
```

You cannot set the type for a resource that is requested by a job or reservation, or set on a server, queue, or vnode.

You cannot set the flag(s) to *h*, *nh*, *fh*, or *q* for a resource that is requested by a job or reservation.

2.48.7.5 Unsetting Resource Flag(s)

You can use the `qmgr` command to unset the flag(s) for resources.

- To unset the flag(s) of the named resource:

  unset resource <resource name> flag

  For example:

  ```
  qmgr -c "unset resource foo flag"
  ```

You can unset the flag(s) of multiple resources by separating the resource names with commas. For example:

```
qmgr -c "unset resource r1,r2 flag"
```

You cannot unset the type for a resource.

You cannot unset the flag(s) for a resource that is requested by a job or reservation, or set on any server, queue, or vnode.

2.48.8 Viewing Object, Attribute, and Resource Information

2.48.8.1 Listing Objects and Their Attributes

You can use the `qmgr` command to list attributes of any object, including attributes at their default values.

- To list the attributes, with associated values, of the named object(s):

  list <object type> <object name>[,<object name> ...]
- To list values of the specified attributes of the named object:

  list <object type> <object name> <attribute name>[, <attribute name>]...
- To list attributes, with associated values, of active objects of the specified type at the active server:

  list <object type>
- To list all objects of the specified type at the specified server, with their attributes and the values associated with the attributes:

  list <object type> @<server name>
- To list attributes of the active server:

  list server

  If no server other than the default server has been made active, lists attributes of the default server (it is the active server).
- To list attributes of the specified server:

list server <server name>

- To list attributes of all schedulers:

  list sched

- To list attributes of the specified scheduler:

  list sched <scheduler name>

- To list all hooks, along with their attributes:

  list hook

- To list attributes of the specified hook:

  list hook <hook name>

2.48.8.1.i Examples of Listing Objects and Their Attributes

Example 2-14: List serverA's schedulers' attributes:

```
list sched @serverA
```

Example 2-15: List attributes for default server's scheduler(s):

```
l sched @default
```

Example 2-16: List PBS version for default server's scheduler(s):

```
l sched @default pbs_version
```

Example 2-17: List queues at a specified server:

```
list queue @server1
```

2.48.8.2 Listing Resource Definitions

You can use the `qmgr list` and `print` commands to list resource definitions showing resource name, type, and flag(s).

- To list the name, type, and flag(s) of the named resource(s):

  list resource <resource name>[,<resource name> ...]

  or

  print resource <resource name>[,<resource name> ...]

- To list name, type, and flag(s) of custom resources only:

  list resource

  or

  print resource

  or

  print server (note that this also prints information for the active server)

- To list all custom resources at the specified server, with their names, types, and flags:

  list resource @<server name>

  or

  print resource @<server name>

When used by a non-privileged user, `qmgr` prints only resource definitions for resources that are visible to non-privileged users (those that do not have the *i* flag set).

2.48.8.3 Printing Creation and Configuration Commands

For printing the creation commands for any object except for a built-in hook.

- To print out the commands to create the named object(s) and set their attributes to their current values:

  print <object type> <object name>[,<object name> ...]

  where *object name* follows the name rules in section 2.48.5.1.ii, “Using Lists of Object Names”, on page 151.
- To print out the commands to create the named object and set its attributes to their current values:

  print <object type> <object name> [<attribute name>[, <attribute name>]...]

  where *object name* follows the name rules in section 2.48.5.1.ii, “Using Lists of Object Names”, on page 151.
- To print out the commands to create and configure the active objects of the named type:

  print <object type>
- To print out the commands to create and configure all of the objects of the specified type at the specified server:

  print <object type> @<server name>
- To print out the commands to create each queue, set the attributes of each queue to their current values, and set the attributes of the server to their current values:

  print server

  This is used for the server and queues, but not hooks.

  Prints information for the active server. If there is no active server, prints information for the default server.
- To print out the creation commands for all schedulers:

  print sched
- To print out the creation commands for the specified scheduler:

  print sched <scheduler name>

2.48.8.4 Caveats for Viewing Information

Some attributes whose values are unset do not appear in the output of the `qmgr` command.

Definitions for built-in resources do not appear in the output of the `qmgr` command.

When a non-privileged user prints resource definitions, `qmgr` prints only resource definitions for resources that are visible to non-privileged users (those that do not have the *i* flag set).

2.48.9 Saving and Re-creating Server and Queue Information

To save and recreate server and queue configuration, print the configuration information to a file, then read it back in later. For example, to save your configuration:

```
# qmgr -c "print server" > savedsettings
```

or

```
Qmgr: print server > savedsettings
```

When re-creating queue and server configuration, read the commands back into `qmgr`. For example:

```
qmgr < savedsettings
```

2.48.10 Operating on Hooks

2.48.10.1 Creating Hooks

- To create a hook:

  Qmgr: create hook <hook name>

  For example:

  Qmgr: create hook my_hook

2.48.10.2 Deleting Hooks

- To delete a hook:

  Qmgr: delete hook <hook name>

  For example:

  Qmgr: delete hook my_hook

2.48.10.3 Setting and Unsetting Hook Attributes

- To set a hook attribute:

  Qmgr: set hook <hook name> <attribute> = <value>
- To unset a hook attribute:

  Qmgr: unset hook <hook name> <attribute>

Example 2-18: Unset hook1's alarm attribute, causing hook1's alarm to revert to its default value of 30 seconds:

Qmgr: unset hook hook1 alarm

2.48.10.4 Importing Hooks

For importing the contents of a site-defined hook. Cannot be used with built-in 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 site-defined hook:

import hook <hook name> application/x-python <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 of *content type* and must be encoded with *content encoding*.
- The allowed values for *content encoding* are "*default*" (7bit) and "*base64*".
- If the source of input is `stdin` (-) and *content encoding* is "*default*", `qmgr` expects the input data to be terminated by *EOF*.
- If the source of input is `stdin` (-) and *content encoding* is "base64", `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, that is overwritten by this import call.
- If the name in *input file* contains spaces as are used in Windows filenames, *input file* must be quoted.

There is no restriction on the size of the hook script.

2.48.10.4.i Examples of Importing Hooks

Example 2-19: Given a Python script in ASCII text file "hello.py", use its contents as 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 2-20: 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'
```

Example 2-21: 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
```

2.48.10.5 Importing and Exporting Hook Configuration Files

2.48.10.5.i Importing Configuration Files

For importing the contents of a site-defined or built-in hook configuration file. To import a hook configuration file, you import the contents of a file to a hook. You must specify a filename that is locally accessible to qmgr and the PBS server.

Format for importing a site-defined hook configuration file:

import hook <hook name> application/x-config <content encoding> {<config file>|-}

Format for importing a built-in hook configuration file:

import pbshook <hook name> application/x-config <content encoding> {<config file>|-}

This uses the contents of *config file* or stdin (-) as the contents of the configuration file for hook *hook name*.

- The *config file* or stdin (-) data must have a format of *content-type* and must be encoded with *content encoding*.
- The allowed values for *content encoding* are "*default*" (7bit) and "*base64*".
- If the source of input is stdin (-) and *content encoding* is "*default*", qmgr expects the input data to be terminated by *EOF*.
- If the source of input is stdin (-) and *content encoding* is "base64", qmgr expects input data to be terminated by a blank line.
- *config file* must be locally accessible to both qmgr and the requested batch server.
- A relative path *config file* is relative to the directory where qmgr was executed.
- If a hook already has a configuration file, that file is overwritten by this import call.
- If the name in *config file* contains spaces as are used in Windows filenames, *input file* must be quoted.

There is no restriction on the size of the hook configuration file.

2.48.10.5.ii Exporting Configuration Files

Format for exporting a site-defined hook configuration file:

qmgr -c "export hook <hook name> application/x-config default" > {<config file>|-}

Format for exporting a built-in hook configuration file:

qmgr -c "export pbshook <hook name> application/x-config default" > {<config file>|-}

2.48.10.5.iii Hook Configuration File Format

PBS supports several file formats for configuration files. The format of the file is specified in its suffix. Formats can be any of the following:

- .ini
- .json
- .py (Python)
- .txt (generic, no special format)
- .xml
- No suffix: treat the input file as if it is a .txt file
- The dash (-) symbol: configuration file content is taken from STDIN. The content is treated as if it is a .txt file.

Example 2-22: To import a configuration file in .json format:

```
# qmgr -c "import hook my_hook application/x-config default my_input_file.json"
```

2.48.10.6 Exporting Hooks

For exporting the contents of a site-defined hook. Cannot be used with built-in hooks.

Format for exporting a hook:

qmgr -c "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, an error message is displayed in stderr.
- If the *output file* name contains spaces like the ones used in Windows file names, *output file* must be enclosed in quotes.

2.48.10.6.i Examples of Exporting Hooks

Example 2-23: Dump 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 2-24: To< dump the script contents of a hook 'hook1' into a file in "\My Hooks\hook1.py":

```
qmgr -c "export hook hook1 application/x-python default" > "\My Hooks\hook1.py"
```

2.48.10.7 Printing Hook Information

- To print out the commands to create and configure all hooks, including their configuration files:

print hook

- To print out the commands to create and configure the specified hook, including its configuration file:

  print hook <hook name>

2.48.10.8 Saving and Re-creating Hook Information

You can save creation and configuration information for all hooks. For example:

```
# qmgr -c "print hook" > hook.qmgr
```

You can re-create all hooks and their configuration files. For example:

```
# qmgr < hook.qmgr
```

2.48.10.9 Restrictions on Built-in Hooks

You cannot do the following with built-in hooks:

- Import a built-in hook
- Export a built-in hook
- Print creation commands for a built-in hook
- Create a built-in hook
- Delete a built-in hook
- Set the type attribute for a built-in hook

2.48.11 Printing Usage Information

You use the `help` command or a question mark ("?") to invoke the `qmgr` built-in help function. You can request usage information for any of the `qmgr` commands, and for topics including attributes, operators, names, and values.

- To print out usage information for the specified command or topic:

  Qmgr: help [<command or topic>]

  or

  Qmgr: ? [<command or topic>]

  For example, to print usage information for the `set` command:

```
qmgr
Qmgr: help set
Syntax: set object [name][,name...] attribute[.resource] OP value
```

2.48.12 Standard Input

When you start a `qmgr` session, the `qmgr` command reads standard input for directives until it reaches end-of-file, or it reads the *exit* or *quit* command.

2.48.13 Standard Output

When you start a `qmgr` session, and standard output is connected to a terminal, `qmgr` writes a command prompt to standard output.

If you specify the `-e` option, `qmgr` echoes the directives it reads from standard input to standard output.

2.48.14 Standard Error

If you do not specify the `-z` option, the `qmgr` command writes a diagnostic message to standard error for each error occurrence.

2.48.15 Exit Status

0
: Success

1
: Error in parsing

2
: Error in execution

3
: Error connecting to server

4
: Error making object active

5
: Memory allocation error

2.48.16 See Also

The PBS Professional Administrator's Guide, Chapter 6, "Attributes", on page 269, Chapter 5, "List of Built-in Resources", on page 255

2.49 qmove

Moves a PBS job from one queue to another

2.49.1 Synopsis

qmove <destination> <job ID> [<job ID> ...]

qmove --version

2.49.2 Description

Moves a job from one queue to another.

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.

2.49.2.1 Restrictions

The qmove command can be used on job arrays, but not on subjobs or ranges of subjobs.

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.

2.49.2.2 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 following restrictions apply only to unprivileged users:

- 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
- The queue cannot have its from_route_only attribute set to *True* (accepting jobs only from routing queues)

2.49.3 Options

--version

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

2.49.4 Operands

destination

Where job(s) are to end up. First operand. Syntax:

<queue name>

Moves the job(s) into the specified queue at the job's current server.

@<server name>

Moves the job(s) into the default queue at specified server.

<queue name>@<server name>

Moves the job(s) into the specified queue at the specified server.

See Chapter 7, "Formats", on page 343 for destination identifier formats.

job ID

Job(s) and/or job array(s) to be moved to the new destination . The `qmove` command accepts one or more *job ID* operands of the form:

<sequence number>[.<server name>][@<server name>]

<sequence number>[][.<server name>][@<server name>]

Note that some shells require that you enclose a job array identifier in double quotes.

2.49.5 Standard Error

The `qmove` command writes a diagnostic messages to standard error for each error occurrence.

2.49.6 Exit Status

Zero

Upon successful processing of all the operands presented to the `qmove` command.

Greater than zero

If the `qmove` command fails to process any operand.

2.49.7 See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide

2.50 qmsg

Writes message string into one or more job output files

2.50.1 Synopsis

qmsg [-E] [-O] <message string> <job ID> [<job ID> ...]

qmsg --version

2.50.2 Description

Writes 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. Also called “sending a message to a 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.

2.50.3 Options

-E
: The message is written to the standard error of each job.

-O
: 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.

(no options)
: The message is written to the standard error of each job.

2.50.4 Operands

message string
: The message to be written. String. First operand. If the string contains blanks, the string must be quoted. If the final character of the string is not a newline, a newline character is added when written to the job’s file.

job ID
: The job(s) to receive the message string. This operand follows the *message string* operand. Cannot be a job array, subjob, or range of subjobs. The `qmsg` command accepts one or more *job ID* operands of the form:

  <sequence number>[.<server name>][@<server name>]

2.50.5 Standard Error

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

2.50.6 Exit Status

Zero

Upon successful processing of all the operands presented to the `qmsg` command.

Greater than zero

If the `qmsg` command fails to process any operand.

2.50.7 See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide

2.51 qorder

Swaps queue positions of two PBS jobs

2.51.1 Synopsis

qorder <job ID> <job ID>

qorder --version

2.51.2 Description

Exchanges positions in queue(s) of two jobs, whether in the same or different queue(s).

No attribute of either 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.

2.51.2.1 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.

2.51.2.2 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.

2.51.3 Options

--version

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

2.51.4 Operands

Both operands are job IDs which specify the jobs to be exchanged. The `qorder` command accepts two *job ID* operands of the form:

<sequence number>[.<server name>][@<server name>]

<sequence number>[][.<server name>][@<server name>]

If you specify the server for both jobs, they must be at the same server.

Note that some shells require that you enclose a job array identifier in double quotes.

2.51.5 Standard Error

The `qorder` command writes diagnostic messages to standard error for each error occurrence.

2.51.6 Exit Status

Zero

Upon successful processing of all the operands presented to the `qorder` command

Greater than zero

If the `qorder` command fails to process any operand

2.51.7 See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide

2.52 qrerun

Requeues a PBS job

2.52.1 Synopsis

qrerun [-W force] <job ID> [<job ID> ...]

qrerun --version

2.52.2 Description

If possible, kills the specified job(s), then requeues each job in the execution queue from which it was run.

The `qrerun` command can be used on jobs, job arrays, subjobs, and ranges of subjobs. If you give a job array identifier as an argument, the job array is returned to its initial state at submission time, or to its altered state if it has been qaltered. All of that job array's subjobs are requeued, which includes those that are currently running, and those that are completed and deleted. If a you give a subjob or range as an argument, those subjobs are requeued.

2.52.2.1 Restrictions

If a job is marked as not rerunnable, `qrerun` neither kills nor requeues the job. See the `-r` option for the `qsub` and `qalter` commands, and the Rerunable job attribute.

The `qrerun` command cannot requeue a job or subjob which is not running, is held, or is suspended.

2.52.2.2 Required Privilege

PBS Manager or Operator privilege is required to use this command.

2.52.3 Options

-W force

The job is to be requeued even if the vnode 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.

2.52.4 Operands

The `qrerun` command accepts one or more *job ID* operands of the form:

<sequence number>[.<server name>][@<server name>]

<sequence number>[][.<server name>][@<server name>]

<sequence number>[<index>][.<server name>][@<server name>]

<sequence number>[<index start>-<index end>][.<server name>][@<server name>]

Note that some shells require that you enclose a job array identifier in double quotes.

2.52.5 Standard Error

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

2.52.6 Exit Status

Zero
: Upon successful processing of all operands

Greater than zero
: Upon failure to process any operand

2.52.7 See Also

PBS Professional Administrator's Guide, PBS Professional User’s Guide

2.53 qrls

Releases holds on PBS jobs

2.53.1 Synopsis

qrls [-h <hold list>] <job ID> [<job ID> ...]

qrls --version

2.53.2 Description

The `qrls` command releases or removes holds on batch jobs or job arrays, but not on subjobs or ranges of subjobs.

A job may have one or more types of holds which make the job ineligible for execution.

When you `qrls` a job whose Execution_Time attribute is not set to a time in the future, the job changes to the *queued* state. If Execution_Time is in the future, the job changes to the *waiting* state.

Holds can be set by the owner, an Operator, or Manager, when a job has a dependency, or when a job has its Execution_Time attribute set to a time in the future. See "qhold" on page 144.

2.53.2.1 Effect of Privilege on Behavior

The following table shows the holds and the privilege required to release each:

Table 2-20: Hold Types

Hold Type	Meaning	Privilege Required to Release
u	*User*	*Job owner, Operator, Manager, PBS Administrator, root*
o	*Other*	*Operator, Manager, administrator, root*
s	*System*	*Manager, administrator, root, PBS (dependency)*
n	*No hold*	*Job owner, Operator, Manager, administrator, root*
p	*Bad password*	*Administrator, root*

If you try to release a hold for which the you do not have privilege, the entire request is rejected, and no holds are released.

2.53.3 Options

(no options)

Defaults to `-h u`, removing *user* hold.

-h <hold list>

Types of hold to be released for 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 of the letters *n* or *p*.

--version

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

2.53.4 Operands

The `qrls` command can be used on jobs and job arrays, but not on subjobs or ranges of subjobs. The `qrls` command accepts one or more *job ID* operands of the form:

<sequence number>[.<server name>][@<server name>]

<sequence number>[][.<server name>][@<server name>]

Note that some shells require that you enclose a job array identifier in double quotes.

2.53.5 Standard Error

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

2.53.6 Exit Status

Zero
: Upon successful processing of all the operands presented to the `qrls` command

Greater than zero
: If the `qrls` command fails to process any operand

2.53.7 See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide, "qhold" on page 144

2.54 qrun

Runs a PBS job immediately

2.54.1 Synopsis

qrun [-a] [-H <vnode specification>] <job ID> [<job ID> ...]

qrun [-a] [-H -] <job ID> [<job ID> ...]

qrun --version

2.54.2 Description

Forces a job to run, regardless of scheduling position or resource requirements.

The `qrun` command can be used on jobs, subjobs, or ranges of subjobs, but not on job arrays. When it is used on a range of subjobs, the non-running subjobs in that range are run.

When preemption is enabled, a scheduler preempts other jobs in order to run this job. Running a job via `qrun` gives the job higher preemption priority than any of the priorities defined in the preempt_prio scheduler parameter. See "Using Preemption" on page 182 in the PBS Professional Administrator's Guide.

2.54.2.1 Required Privilege

In order to execute `qrun`, you must have PBS Operator or Manager privilege.

2.54.2.2 Caveats for qrun

- The job is run without respect for limits, primetime, or dedicated time.
- 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.
- If you don't use the -H option, the job must be in the *Queued* state and reside in an execution queue.
- If you do use the -H option, the job must be in the *Queued* or *Suspended* state and reside in an execution queue.
- The `qrun` command cannot be used on a job that is in the process of provisioning.
- If you use the `-H` option, all schedulers are bypassed, and partition boundaries are ignored.

2.54.3 Options to `qrun`

-a

The `qrun` command exits before the job actually starts execution.

(no -H option)

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 by itself, with no `-H` option, 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 primetime slot even when it's not prime-time, or vice versa. Primetime boundaries are not honored.
- Dedicated time: you can run a job in a dedicated time slot, even if it's not in a dedicated time queue, and vice versa. However, dedicated time boundaries are still honored.

The `qrun` command by itself, with no `-H` option, does not override the following:

- Server and queue resource usage limits

(with -H option)

Do **NOT** use this option unless you know exactly what you are doing.

With the `-H` option, all scheduling policies are bypassed and the job is run directly. The job is run immediately on the named or previously-assigned vnodes, regardless of current usage on those vnodes or which scheduler manages those vnodes, with the exception of vnode state. The job is not run and the `qrun` request is rejected if any named vnode is down, already allocated exclusively, or would need to be allocated exclusively and another job is already running on the vnode. The job is run if the vnode is *offline*.

The `-H` option runs jobs that are queued or suspended.

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 name>[+<vnode name> ...]

Example:

```
-H (VnodeA+VnodeB)+(VnodeC)
```

PBS applies 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 name>:<vnode resources>[+<vnode name>:<vnode resources> ...]

and where *vnode resources* has the format

<resource name>=<value>[:<resource name>=<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 with resources*, using it instead of the original specification from the user. Any single resource specification results 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.

-H -

Runs the job on the set of resources to which it is already assigned. You can run a job on the set of resources already assigned to the job, without having to list the resources, by using the **-** (dash) argument to the `-H` option.

--version

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

2.54.4 Operands

Job ID

The `qrun` command accepts a list of job IDs, of the form:

<sequence number>[.<server name>][@<server name>]

<sequence number>[<index>][.<server name>][@<server name>]

<sequence number>[<index start>-<index end>][.<server name>][@<server name>]

Note that some shells require that you enclose a job array identifier in double quotes.

vnode specification

The *vnode specification without resources* has this format:

(<vchunk>)[+(<vchunk>) ...]

where vchunk has the format

<vnode name>[+<vnode name> ...]

Example:

```
-H (VnodeA+VnodeB)+(VnodeC)
```

The *vnode specification with resources* has this format:

(<vchunk>)[+(<vchunk>) ...]

where *vchunk* has the format

<vnode name>:<vnode resources>[+<vnode name>:<vnode resources> ...]

and where *vnode resources* has the format

<resource name>=<value>[:<resource name>=<value> ...]

Example:

```
-H (VnodeA:mem=100kb:ncpus=1) +(VnodeB:mem=100kb:ncpus=2+VnodeC:mem=100kb)
```

A *vnode name* is the name of the vnode, not the name of the host.

2.54.5 Standard Error

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

2.54.6 Exit Status

Zero

On success

Greater than zero

If the `qrun` command fails to process any operand

2.54.7 See Also

The PBS Professional Administrator's Guide

2.55 qselect

Selects specified PBS jobs

2.55.1 Synopsis

qselect [-a [<op>] <date and 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 <time option> [<comparison>] <specified time>] [-T] [-u <user list>] [-x]

qselect --version

2.55.2 Description

The `qselect` command lists those jobs that meet the specified selection criteria. You can compare certain job attribute values to specified values using a comparison operator shown as *op* in the option description.

You can select jobs, job arrays, or subjobs. You can select jobs from one server per call to the command.

Each option acts as a filter restricting which jobs are listed.

You can select jobs according to the values of some of the resources in the Resource_List job attribute. You can also select jobs according the selection directive (although because this is a string, you can only check for equality or inequality.)

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.

2.55.2.1 Comparison Operations

You can select jobs by comparing the values of certain job attributes to values you specify. The following table lists the comparison operations you can use:

Table 2-21: Comparison Operations

Operation	Type of Comparison
.eq.	The value of the job attribute is equal to the value of the option argument.
.ne.	The value of the job attribute is not equal to the value of the option argument.
.ge.	The value of the job attribute is greater than or equal to the value of the option argument.
.gt.	The value of the job attribute is greater than the value of the option argument.
.le.	The value of the job attribute is less than or equal to the value of the option argument.
.lt.	The value of the job attribute is less than the value of the option argument.

For example, to select jobs whose Priority attribute has a value greater than 5:

```
qselect -p.gt.5
```

Where an optional comparison is not specified, the comparison operation defaults to *.eq*, meaning PBS checks whether the value of the attribute is equal to the option argument.

2.55.2.2 Required Permissions

When selecting jobs according to resource values, users without operator or manager privilege cannot specify custom resources which were created to be invisible to unprivileged users.

2.55.3 Options to `qselect`

(no options)

Lists all jobs at the server which the user is authorized to list (query status of).

-a [<op>] <date and time>

Deprecated. Restricts selection to those jobs whose Execution_Time attribute qualifies when compared to the *date and time* argument. You can select a range of execution times by using this option twice, to compare to a minimum time and a maximum time.

The *date and time* argument has the format:

[[CC]YY]MMDDhhmm[.SS]

where *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.

-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 meets the comparison criteria.

The *interval* argument can take one of the following values:

c

c=<minutes>

n

s

w

w=<minutes>

We give the range of interval values for the Checkpoint attribute the following ordered relationship:

n > s > c=<minutes> > c > u

(Information about w and w=<minutes> is not available.)

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. The holds in the Hold_Types job attribute must be the same as those in the *hold list* argument, but can be in a different order.

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 2-22: Hold Types

Letter	Hold Type
n	None
u	User
o	Other
p	Bad password
s	System

-H

Restricts selection to finished and moved jobs.

-J

Limits selection to job arrays only.

-l <resource list>

Restricts selection of jobs to those with specified resource amounts. Resource must be job-wide, or be mem, ncpus, or vmem.

The *resource list* is in the following format:

<resource name> <op> <value>[,<resource name> <op> <value> ...]

You must specify *op*, and you can use any of the comparison operators.

Because resource specifications for chunks using the select statement, and placement using the place statement, are stored as strings, the only useful operators for these are *.eq.* and *.ne.*

Unprivileged users cannot specify custom resources which were created to be invisible to unprivileged users.

-N <name>

Restricts selection of jobs to those with the specified value for the Job_Name attribute.

-p [<op>]<priority>

Restricts selection of jobs to those with the specified Priority value(s).

-P <project>

Restricts selection of jobs to those matching the specified value for the project attribute.

Format: *Project Name*; see "Project Name" on page 347

-q <destination>

Restricts selection to those jobs at the specified *destination*.

The *destination* may take of one of the following forms:

<queue name>

Restricts selection to the specified queue at the default server.

@<server name>

Restricts selection to the specified server.

<queue name>@<server name>

Restricts selection to the specified queue at the specified server.

If the -q option is not specified, jobs are selected from the default server.

-r <rerun>

Restricts selection of jobs to those with the specified value for the Rerunable attribute . The option argument *rerun* must be a single character, either *y* or *n* .

-s <states>

Restricts job selection to those whose job_state attribute has the specified value(s).

The *states* argument is a character string consisting of any combination of these characters: *B, E, F, H, M, Q, R, S, T, U, W*, and *X*. (A repeated character is accepted, but no additional meaning is assigned to it.)

Table 2-23: 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 in any of the specified states are selected.

Job arrays are never in states *R, S, T*, or *U*. Subjobs may be in those states.

-t <time option> [<op>] <specified time>

Jobs are selected according to one of their time-based attributes. The *time option* specifies which time-based attribute is tested. You give the *specified time* in *datetime* format. See Chapter 7, "Formats", on page 343.

The *time option* is one of the following:

Table 2-24: Sub-options to the `-t` Option

Time Option	Time Attribute	Option Format(s)	Attribute Description
a	Execution_Time	*Timestamp* Use *datetime* format to specify.	Time at which the job is eligible for execution.
c	ctime	*Timestamp* Printed by `qstat` in human-readable *Date* format. Output in hooks as seconds since epoch.	Time at which the job was created.
e	etime	*Timestamp* Printed by `qstat` in human-readable *Date* format. Output in hooks as seconds since epoch.	Time 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, or is held then released. Not affected by qaltering.
g	eligible_time	Use *duration* format to specify.	Amount of eligible time job accrued waiting to run.
m	mtime	*Timestamp* Printed by `qstat` in human-readable *Date* format. Output in hooks as seconds since epoch.	Time that the job was last modified, changed state, or changed locations.
q	qtime	*Timestamp* Printed by `qstat` in human-readable *Date* format. Output in hooks as seconds since epoch.	Time that the job entered the current queue.
s	stime	*Timestamp* Printed by `qstat` in human-readable *Date* format. Output in hooks as seconds since epoch	Time the job started. Updated when job is restarted. .
t	estimated.start_time	Use *datetime* format to specify. Printed by `qstat` in human-readable *Date* format. Output in hooks as seconds since epoch.	Job's estimated start time.

To bracket a time period, use 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 jobs and subjobs.

-u <user list>

Restricts selection to jobs owned by the specified usernames.

Syntax of *user list*:

<username>[@<hostname>][,<username>[@<hostname>],...]

Selects jobs which are owned by the listed users at the corresponding hosts. Hostnames may be wildcarded on the left end, e.g. "*.nasa.gov". A username without a "@<hostname>" is equivalent to "<username>@*", meaning that it is valid at any host.

-x

Selects finished and moved jobs in addition to queued and running jobs.

--version

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

2.55.4 Standard Output

PBS writes a list of the selected job IDs to standard output. Each job ID is separated by white space. A job ID can represent a job, a job array, or a subjob. Each job ID has one of the forms:

<sequence number>.<server name>[@<server name>]

<sequence number>[].<server name>[@<server name>]

<sequence number>[<index>].<server name>[@<server name>]

@<server name> identifies the server which currently owns the job.

2.55.5 Standard Error

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

2.55.6 Exit Status

Zero

Upon successful processing of all options presented to the `qselect` command

Greater than zero

If the `qselect` command fails to process any option

2.55.7 See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide, section 6.11, "Job Attributes", on page 318, Chapter 5, "List of Built-in Resources", on page 255

2.56 qsig

Send signal to PBS job

2.56.1 Synopsis

qsig [-s <signal>] <job ID> [<job ID> ...]

qsig --version

2.56.2 Description

The `qsig` command sends a signal to all the processes of the specified job(s). The `qsig` command sends a Signal Job batch request to the server which owns the job.

The `qsig` command can be used for jobs, job arrays, subjobs, and ranges of subjobs. If it is used on a range of subjobs, the running subjobs in the range are signaled.

Not all signal names are recognized by `qsig`; if using a signal name does not work, try issuing the signal number instead.

2.56.2.1 Using admin-suspend and admin-resume

If you have a vnode requiring maintenance while remaining powered up, where you don't want jobs running during the maintenance, you can use the special signals *admin-suspend* and *admin-resume* to suspend and resume the jobs on the vnode. When you use *admin-suspend* on a vnode's job(s), the vnode goes into the *maintenance* state, and its scheduler does not schedule jobs on it. You must separately *admin-suspend* each job on the vnode. When its last *admin-suspend*ed job is *admin-resume*d, a vnode leaves the *maintenance* state.

2.56.2.2 Restrictions

The request to signal a job is rejected if:

- The user is not authorized to signal the job
- The job is not in the *running* or *suspended* state
- The requested signal is not supported by the system upon which the job is executing
- The job is in the process of provisioning
- You attempt to use *admin-resume* on a job that was *suspend*ed
- You attempt to use *resume* on a job that was *admin-suspend*ed

2.56.2.3 Required Privilege

Manager or Operator privilege is required to use the *admin-suspend*, *admin-resume*, *suspend*, or *resume* signals. Unprivileged users can use other signals.

2.56.3 Options to qsig

(no options)

PBS sends SIGTERM to the job.

-s <signal>

PBS sends signal *signal* to the job.

--version

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

2.56.3.1 Signals

You can send standard signals to a job, or the special signals described below. The *signal* argument can be in any of the following formats:

- A signal name, e.g. *SIGKILL*
- A signal name without the SIG prefix, e.g. *KILL*
- An unsigned signal number, e.g. *9*

The signal name *SIGNULL* is allowed; in this case the server sends the signal 0 to the job, which has no effect.

2.56.3.1.i Special Signals

The following special signals are all lower-case, and have no associated signal number:

admin-suspend

Suspends a job and puts its vnodes into the *maintenance* state. The job is put into the *S* state and its processes are suspended. When suspended, a job is not executing and is not charged for walltime.

Syntax: *qsig -s admin-suspend <job ID>*

admin-resume

Resumes a job that was suspended using the *admin-suspend* signal, without waiting for its scheduler. Cannot be used on jobs that were suspended with the *suspend* signal.

Syntax: *qsig -s admin-resume <job ID>*

suspend

Suspends specified job(s). Job goes into *suspended (S)* state. When suspended, a job is not executing and is not charged for walltime.

resume

Marks specified job(s) for resumption by its scheduler when there are sufficient resources. If you use `qsig -s resume` on a job that was suspended using `qsig -s suspend`, the job is resumed when there are sufficient resources. Cannot be used on jobs that were suspended with the *admin_suspend* signal.

2.56.4 Operands

The `qsig` command accepts one or more *job ID* operands. For a job, this has the form:

<sequence number>[.<server name>][@<server name>]

For a job array, *job ID* takes this form:

<sequence number>[][.<server name>][@<server name>]

Note that some shells require that you enclose a job array identifier in double quotes.

2.56.5 Standard Error

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

2.56.6 Exit Status

Zero
: Upon successful processing of all the operands presented to the `qsig` command

Greater than zero
: If the `qsig` command fails to process any operand

2.56.7 See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide

2.57 qstart

Turns on scheduling or routing for the jobs in a PBS queue

2.57.1 Synopsis

qstart <destination> [<destination> ...]

qstart --version

2.57.2 Description

If *destination* is an execution queue, the `qstart` command allows a PBS scheduler to schedule jobs residing in the specified queue. If *destination* is a routing queue, the server can begin routing jobs from that queue. Sets the value of the queue's started attribute to *True*.

2.57.2.1 Required Privilege

In order to execute `qstart`, you must have PBS Operator or Manager privilege.

2.57.3 Options

--version

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

2.57.4 Operands

The `qstart` command accepts one or more space-separated *destination* operands. The operands take one of three forms:

<queue name>

Starts scheduling or routing from the specified queue.

@<server name>

Starts scheduling or routing from all queues at the specified server.

<queue name>@<server name>

Starts scheduling or routing from the specified queue at the specified server.

To start scheduling at all queues at the default server, use the `qmgr` command:

```
Qmgr: set queue @default started=true
```

2.57.5 Standard Error

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

2.57.6 Exit Status

Zero
: Upon successful processing of all the operands presented to the `qstart` command

Greater than zero
: If the `qstart` command fails to process any operand

2.57.7 See Also

The PBS Professional Administrator's Guide, "qmgr" on page 146, "qstop" on page 205

2.58 qstat

Displays status of PBS jobs, queues, or servers

2.58.1 Synopsis

2.58.1.1 Displaying Job Status

Default format:

qstat [-E] [-J] [-p] [-t] [-x] [[<job ID> | <destination>] ...]

Long format:

qstat -f [-F json|dsv [-D <delimiter>]] [-E] [-J] [-p] [-t] [-w] [-x] [[<job ID> | <destination>] ...]

Alternate format:

qstat [-a [-w]| -H | -i | -r] [-E] [-G | -M] [-J] [-n [-1] [-w]] [-s [-1] [-w]] [-t] [-T [-w]] [-u <user list>] [[<job ID> | <destination>] ...]

2.58.1.2 Displaying Queue Status

Default format:

qstat -Q [<destination> ...]

Long format:

qstat -Q -f [-F json|dsv [-D <delimiter>]] [-w] [<destination> ...]

Alternate format:

qstat -q [-G | -M] [<destination> ...]

2.58.1.3 Displaying Server Status

Default format:

qstat -B [<server name> ...]

Long format:

qstat -B -f [-F json|dsv [-D <delimiter>]] [-w] [<server name> ...]

2.58.1.4 Displaying Version Information

qstat --version

2.58.2 Description

The `qstat` command displays the status of jobs, queues, or servers, writing the status information to standard output.

When displaying job status information, the `qstat` command displays status information about all specified jobs, job arrays, and subjobs. You can specify jobs by ID, or by destination, for example all jobs at a specified queue or server.

2.58.2.1 Display Formats

You can use particular options to display status information in a default format, an alternate format, or a long format. Default and alternate formats display all status information for a job, queue, or server with one line per object, in columns. Long formats display status information showing all attributes, one attribute to a line.

2.58.2.2 Displaying Information for Finished and Moved Jobs

You can display status information for finished and moved jobs by using the -x and -H options.

If your job has been moved to another server through peer scheduling, give the job ID as an argument to `qstat`. If you do not specify the job ID, your job will not appear to exist. For example, your job 123.ServerA is moved to ServerB. In this case, you can use:

```
qstat 123
```

or

```
qstat 123.ServerA
```

Specifying the full job name, including the server, avoids the possibility that `qstat` will report on a job named 123.ServerB that was moved to ServerA.

To list all jobs at ServerB, you can use:

```
qstat @ServerB
```

2.58.2.3 Required Privilege

Users without Manager or Operator privilege cannot view resources or attributes that are invisible to unprivileged users.

2.58.3 Displaying Job Status

2.58.3.1 Job Status in Default Format

Triggers: any of the -J, -p, -t, or -x options.

The `qstat` command displays job status in default format when you specify any of the -J, -p, -t, or -x options. Jobs are displayed one to a line, with these column headers:

```
Job id   Name       User      Time Use S Queue
-------- ---------- --------- -------- - -----
```

Description of columns:

Table 2-25: Description of Default Job Status Columns

Column	Description
Job id	The job ID assigned by PBS
Name	Job name specified by submitter
User	Username of job owner

Table 2-25: Description of Default Job Status Columns

Column	Description	
Time Use	The CPU time used by the job. 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, this field changes to "00:00:00". After that, every time the MoM polls for resource usage, the field is updated. The MoM on each execution host polls for the usage of all processes on her host belonging to the job. Usage is summed. The polling interval is short when a job first starts running and lengthens to a maximum of 2 minutes. See "Configuring MoM Polling Cycle" on page 48 in the PBS Professional Administrator's Guide.	
S	The job's state. See section 8.1, "Job States", on page 351	
	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	

2.58.3.2 Job Status in Long Format

Trigger: the `-f` option.

If you specify the `-f` (full) option, 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 and exec_vnode strings. The full output can be very large.

The exec_host string has this format:

<host1>/<T1><P1>[+<host2>/<T2>*<P2>+...]*

where

T1 is the task slot number (the index) of the job on host1.

P1 is the number of processors allocated to the job from host1. The number of processors allocated does not appear if it is 1.

The exec_vnode string has the format:

(<vnode1>:ncpus=<N1>:mem=<M1>)[+(<vnode2>:ncpus=<N2>:mem=<M2>)+...]

where

N1 is the number of CPUs allocated to that job on *vnode1*.

M1 is the amount of memory allocated to that job on *vnode1*.

2.58.3.3 Job Status in Alternate Format

Triggers: any of the `-a, -i, -G, -H, -M, -n, -r, -s,` or `-u <user list>` options.

The `qstat` command displays job status in alternate format if you specify any of the `-a, -i, -G, -H, -M, -n, -r, -s,` or `-u <user list>` options. 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.

2.58.3.3.i Job Status Alternate Format Output Columns

Alternate format job status output contains the following columns:

```
                                                  Req'd  Req'd   Elap
Job ID Username Queue Jobname SessID NDS TSK Memory Time  S Time
------ -------- ----- ------- ------ --- --- ------ ----- - ----
```

Description of columns:

Table 2-26: Description of Alternate Format Job Status Columns

Column	Description
Job ID	The job ID assigned by PBS
Username	Username of job owner
Queue	Queue in which the job resides
Jobname	Job name specified by submitter
SessID	Session ID. Appears only if the job is running
NDS	Number of chunks or vnodes 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 "States" on page 351 for states
Elap Time	If CPU time is requested, this shows CPU time. Otherwise, shows walltime

2.58.3.4 Grouping Jobs and Sorting by ID

Trigger: the `-E` option.

You can use the -E option to sort and group jobs in the output of qstat. The -E option groups jobs by server and displays each group by ascending ID. This option also improves qstat performance. The following table shows how the -E option affects the behavior of qstat:

Table 2-27: How -E Option Affects qstat Output

How qstat is Used	Result Without -E	Result With -E
qstat (no job ID specified)	Queries the default server and displays result	No change in behavior; same as without -E option
qstat <list of job IDs from single server>	Displays results in the order specified	Displays results in ascending ID order
qstat <job IDs at multiple servers>	Displays results in the order they are specified	Groups jobs by server. Displays each group in ascending order

2.58.4 Displaying Queue Status

2.58.4.1 Queue Status in Default Format

Trigger: the -Q option by itself.

The qstat command displays queue status in default format if the only option is -Q. 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 2-28: Description of Default Queue Status Columns

Column	Description
Queue	Queue name
Max	Maximum number of jobs allowed to run concurrently in this 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

2.58.4.2 Queue Status in Long Format

Trigger: the -q and -f options together.

If you specify the -f (full) option with the -q option, 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.

2.58.4.2.i Queue Status: Alternate Format

Triggers: any of the -q, -G, or -M options.

The qstat command displays queue status in the alternate format if you specify any of the -q, -G, or -M options. Queue status is displayed one queue to a line, and the lowest line contains totals for some columns.

These are the alternate format queue status column headers:

```
Queue   Memory CPU Time Walltime Node Run Que Lm State
------- ------ -------- -------- ---- --- --- -- -----
```

Description of columns:

Table 2-29: Description of Queue Alternate Status Columns

Column	Description
Queue	Queue name
Memory	Maximum amount of memory that can be requested by a job in this queue
CPU Time	Maximum amount of CPU time that can be requested by a job in this queue
Walltime	Maximum amount of walltime that can be requested by a job in this queue
Node	Maximum number of vnodes that can be requested by a job in this queue
Run	Number of running and suspended jobs. Lowest row is total number of running and suspended jobs in all the queues shown
Que	Number of queued, waiting, and held jobs. Lowest row is total number of queued, waiting, and held jobs in all the queues shown
Lm	Maximum number of jobs allowed to run concurrently in this queue
State	State of this queue: *E* (enabled) or *D* (disabled), and *R* (running) or *S* (stopped)

2.58.5 Displaying Server Status

2.58.5.1 Server Status in Default Format:

Trigger: the -B option.

The qstat command displays server status if the only option given is -B.

Column headers for default server status output:

```
Server   Max   Tot   Que   Run   Hld   Wat   Trn   Ext   Status
-------- ----- ----- ----- ----- ----- ----- ----- ----- ------
```

Description of columns:

Table 2-30: Description of Server Status Default Display Columns

Column	Description
Server	Name of server
Max	Maximum number of jobs allowed to be running concurrently 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

2.58.5.2 Server Status in Long Format

Trigger: the `-f` option.

If you specify the `-f` (full) option, displays full server status information starting with the server name, followed by each server attribute, one to a line, as <*name*> = <*value*> pairs. Includes PBS version information.

2.58.6 Options to `qstat`

2.58.6.1 Generic Job Status Options

-E

Groups jobs by server and displays jobs sorted by ascending ID. When `qstat` is presented with a list of jobs, jobs are grouped by server and each group is displayed by ascending ID. This option also improves `qstat` performance. See section 2.58.3.4, “Grouping Jobs and Sorting by ID”, on page 195.

2.58.6.2 Default Job Status Options

The following options cause job status information to be displayed in default format:

-J

Displays status information for job arrays (not subjobs).

-t

Displays status information for jobs, job arrays, and subjobs. When used with `-J` option, displays status information for subjobs only.

-p

The *Time Use* column is replaced with the percentage completed for the job. For a job array this is the percentage of subjobs completed. For a normal job, it is the percentage of allocated CPU time used.

-x

Displays status information for finished and moved jobs in addition to queued and running jobs.

2.58.6.3 Alternate Job Status Options

The following options cause job status information to be displayed in alternate format:

-a

All queued and running jobs are displayed. If a *destination* is specified, information for all jobs at that *destination* is displayed. If a *job ID* is specified, 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 ID* is given, displays information for that job regardless of its state. If a *destination* is specified, displays information for finished or moved jobs, or specified job(s), at *destination*.

-i

If a *destination* is given, information for queued, held or waiting jobs at that *destination* is displayed. If a *job ID* is given, information about that job is displayed regardless of its state.

-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 does not take effect.

-r

If a *destination* is given, information for running or suspended jobs at that *destination* is displayed. If a *job ID* is given, information about that job is displayed regardless of its state.

-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 does not take effect.

-T

Displays estimated start time for queued jobs, replacing the *Elap Time* field with the *Est Start Time* 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 the unset value shown as a double dash ("--"). Queued jobs with estimated start times in the past are treated as if their estimated start times are unset.

If a job's estimated start time cannot be calculated, the start time is shown as a question mark ("?").

Time displayed is local to the `qstat` command. Current week begins on Sunday.

The following table shows the format for the *Est Start Time* field when the -w option is not used:

Table 2-31: Format for Estimated Start Time Field without -w Option

Format	Job 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	`2018`
>5yrs	More than 5 years from today	`>5yrs`

The following table shows the format for the *Est Start Time* field when the -w option is used:

Table 2-32: Format for Estimated Start Time Field with -w Option

Format	Job Estimated Start Time	Example
Today <HH>:<MM>	Today	`Today 13:34`
<Day> <HH>:<MM>	This week, but after today	`Wed 15:34`
<Day> <Month> <Daynum> <HH>:<MM>	This year, but after this week	`Wed Feb 10 15:34`
<Day> <Month> <Daynum> <YYYY> <HH>:<MM>	After this year	`Wed Feb 10 2011 15:34`

When used with the -f option, prints the full timezone-qualified start time.

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 ID* is given, status information for that job is displayed regardless of the job's ownership.

Format: *<username>[@<hostname>][, <username>[@<hostname>], ...]* in comma-separated list.

Hostnames may be wildcarded, but not domain names. When no hostname is specified, *username* is for any host.

-w

Allows display of wider fields up to 120 characters. The *Job ID* column can be up to 30 characters wide. *Username*, *Queue,* and *Jobname* can be up to 15 characters wide. *SessID* can be up to 8 characters wide and *NDS* can be up to 4 characters wide. *TSK* can be up to 5 characters wide. *Req d Memory* can be 6 characters, *Elap Time* can be 5 characters, and *S* can be only 1 character wide. Can be used only in conjunction with the `-a`, `-n`, `-s`, or `-T` options. This option is different from the `-w` option used with `-f`.

-1

Reformats `qstat` output to a single line. Can be used only in conjunction with the `-n` and/or `-s` options.

2.58.6.4 Queue Status Options

-Q

Displays queue status in default format. Operands must be *destinations*.

-q

Displays queue status in alternate format. Operands must be *destinations*.

2.58.6.5 Server Status Options

-B

Display server status. Operands must be names of servers.

2.58.6.6 Job, Queue, and Server Status Options

-f [-w]

Full display. Job, subjob, queue, or server attributes displayed one to a line.

JSON output:

PBS reports resources_used values for resources that are created or set in a hook as JSON strings in the output of `qstat -f`.

If MoM returns a JSON object (a Python dictionary), PBS reports the value as a string in single quotes:

```
resources_used.<resource_name> = '{ <MoM JSON item value>, <MoM JSON item value>, <MoM JSON item value>, ..}'
```

Example: MoM returns { "a":1, "b":2, "c":1,"d": 4} for resources_used.foo_str. We get:

```
resources_used.foo_str='{"a": 1, "b": 2, "c":1,"d": 4}'
```

If MoM returns a value that is not a valid JSON object, the value is reported verbatim.

Example: MoM returns "hello" for resources_used.foo_str. We get:

```
resources_used.foo_str="hello"
```

Optional `-w` prints each attribute on one unbroken line. Feed characters are converted:

- Newline is converted to backslash concatenated with “n”, resulting in “\n”
- Form feed is converted to backslash concatenated with “f”, resulting in “\f”

This `-w` is independent of the `-w` job alternate format output option.

-F dsv [-D <delimiter>]

Prints output in delimiter-separated value format. The default *delimiter* is a pipe ("|"). You can specify a character or a string *delimiter* using the -D argument to the -F dsv option. For example, to use a comma as the delimiter:

```
qstat -f -F dsv -D,
```

If the delimiter itself appears in a value, it is escaped:

- On Linux, the delimiter is escaped with a backslash ("\").
- On Windows, the delimiter is escaped with a caret ("^").

Feed characters are converted:

- Newline is converted to backslash concatenated with "n", resulting in "\n"
- Form feed is converted to backslash concatenated with "f", resulting in "\f"

A newline separates each job from the next. Using newline as the delimiter leads to undefined behavior.

Example of getting output in delimiter-separated value format:

```
qstat -f -Fdsv
Job Id: 1.vbox|Job_Name = STDIN|Job_Owner = root@vbox|job_state = Q|queue = workq|server =
    vbox|Checkpoint = u|ctime = Fri Nov 11 17:57:05 2016|Error_Path = ...
```

-F json

Prints output in JSON format (http://www.json.org/).

Attribute output is preceded by timestamp, PBS version, and PBS server hostname.

Example:

```
qstat -f -F json
{
    "timestamp":1479277336,
    "pbs_version":"14.1",
    "pbs_server":"vbox",
    "Jobs":{
        "1.vbox":{
            "Job_Name":"STDIN",
            "Job_Owner":"root@vbox",
            "job_state":"Q",
...
```

-G

Shows size in gigabytes. Triggers alternate format.

-M

Shows size in megawords. A word is considered to be 8 bytes. Triggers alternate format.

2.58.6.7 Version Information

--version

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

2.58.7 Operands

2.58.7.1 Job Identifier Operands

The *job ID* is assigned by PBS at submission. Job IDs are used only with job status requests. Status information for specified job(s) is displayed. Formats:

Job ID:

<sequence number>[.<server name>][@<server name>]

Job array ID:

<sequence number>[][.<server name>][@<server name>]

Subjob ID:

<sequence number>[<index>][.<server name>][@<server name>]

Range of subjobs:

<sequence number>[<index start>-<index end>][.<server name>][@<server name>]

Note that some shells require that you enclose a job array identifier in double quotes.

2.58.7.2 Destination Operands

Name of queue, name of server, or name of queue at a specific server. Formats:

queue name

Specifies name of queue for job or queue display.

- When displaying job status, PBS displays status for all jobs in the specified queue at the default server.
- When displaying queue status, PBS displays status for the specified queue at the default server.

queue name@server name

Specifies name of queue at server for job or queue display.

- When displaying job status, PBS displays status for all jobs in the specified queue at the specified server.
- When displaying queue status, PBS displays status for the specified queue at the specified server.

@server name

Specifies server name for job or queue display.

- When displaying job status, PBS displays status for all jobs at all queues at the specified server.
- When displaying queue status, PBS displays status for all queues at the specified server.

server name

Specifies server name for server display.

- When displaying server status (with the `-B` option) PBS displays status for the specified server.

2.58.8 Standard Error

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

2.58.9 Exit Status

Zero

Upon successful processing of all operands

Greater than zero

If any operands could not be processed

2.58.10 See Also

The PBS Professional User's Guide, the PBS Professional Administrator's Guide, "Attributes" on page 269

2.59 qstop

Prevents PBS jobs in the specified queue from being scheduled or routed

2.59.1 Synopsis

qstop <destination> [<destination> ...]

qstop --version

2.59.2 Description

If *destination* is an execution queue, the `qstop` command stops a scheduler from scheduling jobs residing in *destination*. If *destination* is a routing queue, the server stops routing jobs from that queue. Sets the value of the queue's started attribute to *False*.

2.59.2.1 Required Privilege

You must have PBS Operator or Manager privilege to run this command.

2.59.3 Options

--version

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

2.59.4 Operands

The `qstop` command accepts one or more space-separated *destination* operands. The operands take one of three forms:

<queue name>

Stops scheduling or routing from the specified queue.

@<server name>

Stops scheduling or routing from all queues at the specified server.

<queue name>@<server name>

Stops scheduling or routing from the specified queue at the specified server.

To stop scheduling at all queues at the default server, use the `qmgr` command:

```
Qmgr: set queue @default started=false
```

2.59.5 Standard Error

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

2.59.6 Exit Status

Zero

Upon successful processing of all operands presented to the `qstop` command

Greater than zero

If the `qstop` command fails to process any operand

2.59.7 See Also

The PBS Professional Administrator's Guide, "qmgr" on page 146, "qstart" on page 190

2.60 qsub

Submits a job to PBS

2.60.1 Synopsis

qsub [-a <date and time>] [-A <account string>] [-c <checkpoint spec>] [-C <directive prefix>] [-e <path>] [-f] [-h] [-I [-G [-- <GUI application/script>]] | [-X]] [-j <join>] [-J <range>] [-k <discard>] [-l <resource list>] [-m <mail events>] [-M <user list>] [-N <name>] [-o <path>] [-p <priority>] [-P <project>] [-q <destination>] [-r <y | n>] [-R <remove options>] [-S <path list>] [-u <user list>] [-v <variable list>] [-V] [-W <additional attributes>] [-z] [- | <script> | -- <executable> [<arguments to executable>]]

qsub --version

2.60.2 Description

You use the `qsub` command 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>.<server name>

For an array job, this is of the form:

<sequence number>[].<server name>

During execution, jobs can be interactive or non-interactive. Interactive jobs are not rerunnable, and if they are blocking, you cannot use their exit status.

Jobs are run as the user and group who submitted the job.

2.60.2.1 Background Process

By default, on the first invocation, `qsub` spawns a background process to manage communication with the PBS server. Later invocations of `qsub` attempt to communicate with this background process. Under certain circumstances, calls to `qsub` when it uses the background process can result in communication problems. You can prevent `qsub` from spawning a background process by using the `-f` option, although this can degrade performance.

2.60.2.2 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. However, you can specify the output paths using the `-o` and `-e` options. You can also specify which and whether these files should be kept on the execution host via the `-k` option, or deleted, using the `-R` option.

See the `-k`, `-o`, `-e`, and `-R` options, and "Managing Output and Error Files", on page 41 of the PBS Professional User's Guide.

2.60.2.3 Submitting Jobs By Using Job Scripts

To submit a PBS job by using a script, you specify a job script on the command line:

qsub [<options>] <script name>

For example:

```
qsub myscript.sh
```

Job scripts are run as the user and group who submitted the job. Job scripts can be written in Python, Linux shells such as `csh` and `sh`, the Windows command batch language, Perl, etc.

A PBS job script consists of the following:

- Optional shell specification
- Any PBS directives
- The user's tasks: programs, commands, or applications
- Optional comments

  Under Windows, comments can contain only ASCII characters. See the PBS Professional User's Guide.

2.60.2.3.i Using Shells and Interpreters

By default, PBS uses your login shell to run your script. You can optionally specify a different shell or interpreter to run your script:

- Via the `-S` option to `qsub`:

  qsub -S <path to shell> <script name>

  For example:

  ```
  qsub -S /bin/bash myscript.sh
  ```

- In the first line of your script. For example:

  ```
  cat myscript.sh
  #!/bin/sh
  #PBS -N MyHelloJob
  print "Hello"
  ```

2.60.2.3.ii Python Job Scripts

You can use the same Python script under Linux or under Windows, if the script is written to be portable. PBS includes a Python package, allowing Python job scripts to run; you do not need to install Python. You can include PBS directives in a Python job script as you would in a Linux shell script. Python job scripts can access Win32 APIs, including the following modules:

Win32api

Win32con

Pywintypes

Example 2-25: We have a Python job script that includes PBS directives:

```
cat myjob.py
#!/usr/bin/python
#PBS -l select=1:ncpus=3:mem=1gb
#PBS -N HelloJob
print "Hello"
```

To run a Python job script under Linux, use the Python path on the execution host:

qsub -S <Python path on execution host> <script name>

For example,

```
qsub -S $PBS_EXEC/bin/pbs_python <script name>
```

To run a Python job script under Windows, use the Python path on the execution host:

qsub -S <Python path on execution host> <script name>

For example:

qsub -S %PBS_EXEC%\bin\pbs_python.exe <script name>

If the script pathname contains spaces, it must be quoted, for example:

qsub -S "C:\Program Files\PBS\bin\pbs_python.exe" <script name>

2.60.2.3.iii Linux Shell Job Scripts

Example 2-26: We have a Linux job script named "weatherscript" for a job named "Weather1" which runs the executable "weathersim" on 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>

2.60.2.3.iv Windows Command Job Scripts

Example 2-27: We have 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>

In Windows, if you use `notepad` to create a job script, the last line does not automatically get newline-terminated. Be sure to put one explicitly, otherwise, PBS job will get the following error message:

```
More?
```

when the Windows command interpreter tries to execute that last line.

2.60.2.4 Submitting Jobs From Standard Input

To submit a PBS job by typing job specifications at the command line, you type:

qsub [<options>] [-] <return>

then type any directives, then any tasks, followed by:

- Linux: `CTRL-D` on a line by itself
- Windows: `CTRL-Z <return>`

to terminate the input.

The `qsub` command behaves the same both with and without the dash operand.

For example, on Linux:

```
qsub <return>
#PBS -N StdInJob
sleep 100
<CTRL-D>
```

2.60.2.5 Submitting Job Directly by Specifying Executable on Command Line

To submit a job directly, you specify the executable on the command line:

qsub [<options>] -- <executable> [<arguments to executable>] <return>

When you run `qsub` this way, it runs the *executable* directly. It does not start a shell, so no shell initialization scripts are run, and execution paths and other environment variables are not set. There is not an easy way to run your command in a different directory. You should make sure that environment variables are set correctly, and you will usually have to specify the full path to the command.

Example 2-28: To run `myprog` with the arguments *a* and *b*:

```
qsub -- myprog a b <return>
```

Example 2-29: To run `myprog` with the arguments *a* and *b*, naming the job "JobA":

```
qsub -N JobA -- myprog a b <return>
```

2.60.2.6 Requesting Resources and Placing Jobs

Requesting resources includes setting limits on resource usage and controlling how the job is placed on vnodes.

Resources are requested by using the `-l` option, either in job-wide requests using *<resource name>=<value>* pairs, or in chunks inside of *selection statements*. See Chapter 5, "List of Built-in Resources", on page 255.

Job-wide *<resource name>=<value>* requests are of the form:

-l <resource name>=<value>[,<resource name>=<value> ...]

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> ...]

You choose how your chunks are placed using the *place statement*. The *place statement* can contain the following elements, in any order:

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

where

arrangement

Whether this chunk is willing to share this vnode or host with other chunks from the same job. One of *free* | *pack* | *scatter* | *vscatter*

sharing

Whether this this chunk is willing to share this vnode or host with other jobs. One of *excl* | *shared* | *exclhost*

grouping

Whether the chunks from this job should be placed on vnodes that all have the same value for a resource. Can have only one instance of *group=<resource name>*

free

Place job on any vnode(s).

pack

All chunks are taken from one host.

scatter

Only one chunk with any MPI processes is taken from a host. A chunk with no MPI processes may be taken from the same vnode 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 name>

Chunks are grouped according to a resource. All vnodes in the group must have a common value for *resource*, which can be either the built-in resource host or a custom vnode-level resource.

resource name must be a string or a string array.

The place statement cannot begin with a colon. Colons are delimiters; use them only to separate parts of a place statement, unless they are quoted inside resource values.

Note that vnodes can have sharing attributes that override job placement requests. See section 6.10, “Vnode Attributes”, on page 311.

For more on resources, resource requests, usage limits, and job placement, see "Using PBS Resources" on page 227 in the PBS Professional Administrator’s Guide and "Allocating Resources & Placing Jobs", on page 51 of the PBS Professional User’s Guide.

2.60.2.6.i Caveats for Requesting Resources

Do not mix old-style resource or vnode 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 unprivileged 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.

2.60.2.7 Setting Attributes

The job submitter sets job attributes by giving options to the `qsub` command or by using PBS directives. Most `qsub` options set a job attribute, and have 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 318.

2.60.2.8 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 overridden by command-line arguments and script directives. See section 6.6, “Server Attributes”, on page 273.

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.

2.60.3 Options to `qsub`

-a <date and 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 and time*.

Format: *datetime,* expressed as *[[[[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 to *checkpoint spec*. An $action script is required to checkpoint the job.

See "Using Checkpointing", on page 115 of the PBS Professional User's Guide.

The argument *checkpoint spec* can take one of the following values:

c

Checkpoint at intervals, measured in CPU time, set on job's execution queue. If there is no interval set at the queue, the 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 greater than zero. If the interval specified is less than that set on the job's execution queue, the queue's interval is used.

Format: *Integer*

w

Checkpoint at intervals, measured in walltime, set on job's execution queue. If there is no interval set at the queue, the 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 than that set on the job's execution queue, 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. The string "PBS_DPREFIX" cannot be used as a PBS directive. Length limit: 4096 characters.

-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>

The *path* is interpreted as follows:

path

If *path* is relative, 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* is absolute, it is taken to be an absolute path on the current host where the `qsub` command is executing.

hostname:path

If *path* is relative, it is taken to be relative to the user's home directory on the host named *hostname*.

If *path* is absolute, it is an absolute path on the host named *hostname*.

If *path* does not include a filename, the default filename has the form *<job ID>.ER*

If the `-e` option is not specified, PBS copies the standard error to the current working directory where the `qsub` command was executed, and writes standard error to the default filename, which has this form:

<job name>.e<sequence number>

If you use a UNC path for output or error files, the *hostname* is optional. If you use a non-UNC path, the *hostname* is required.

This option is overridden by the `-k` option.

-f

Prevents `qsub` from spawning a background process. By default, `qsub` spawns a background process to manage communication with the PBS server. When this option is specified, the `qsub` process connects directly to the server and no background process is created.

NOTE: Use of this option degrades performance of `qsub` when calls to `qsub` are made in rapid succession.

-G [<path to GUI application or script>]

Starts a GUI session. When no application or script is provided, starts a GUI-enabled interactive shell. When an application or script is provided, starts the GUI application or script. Use full path to application or script unless the path is part of the user's PATH environment variable on the execution host. When submission and execution hosts are different, this uses a remote viewer.

Session is terminated when remote viewer, GUI application, or interactive shell is terminated, or when job is deleted.

Can be used only with interactive jobs (the `-I` option).

Available only under Windows.

-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.

-j <join>

Specifies whether and how to join the job's standard error and standard output streams. Sets job's Join_Path attribute to *join*.

Default: *n*; not merged

The *join* argument can take the following values:

Table 2-33: Sub-options to -j 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.

-J <range>

Makes this job an array job. Sets job's array attribute to *True*. Use the *range* argument to specify the indices of 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*. Indices must be greater than or equal to zero.

Job arrays are always rerunnable.

-k <discard>

Specifies whether and which of the standard output and standard error streams is left behind on the execution host, or written to their final destination. Sets the job's Keep_Files attribute to *discard*. Overrides default path names for these streams. Overrides -o and -e options.

Default: *n*; neither is retained, and files are not written directly to final destinations.

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 *discard* argument can take the following values:

Table 2-34: Sub-options to discard 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.
d	Output and/or error are written directly to their final destination. Overrides action of leaving files on execution host.
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.

For how to request resources and place jobs, see section 2.60.2.6, "Requesting Resources and Placing Jobs", on page 210.

-m <mail events>

Specifies 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 one of the following:

- The single character "*n*"
- Any combination of "*a*", "*b*", and "*e*", with optional "*j*"

The following table lists the sub-options to the -m option:

Table 2-35: Sub-options to m Option

Suboption	Meaning
n	No mail is sent.
a	Mail is sent when the job is aborted by PBS.
b	Mail is sent when the job begins execution.
e	Mail is sent when the job terminates.
j	Mail is sent for subjobs. Must be combined with one or more of a, b, or e options

Format: *String*

Syntax: *n | [j](one or more of a, b, e)*

Example: -m ja

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 has the form:

<username>[@<hostname>][,<username>[@<hostname>],...]

Default: Job owner

-N <name>

Sets job's Job_Name attribute and name to *name*.

Format: *Job Name*; see "Job Name, Job Array Name" on page 345

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 has the form:

[<hostname>:]<path>

The *path* is interpreted as follows:

path

If *path* is relative, it is taken to be relative to the current working directory of the command, where it is executing on the current host.

If *path* is absolute, it is taken to be an absolute path on the current host where the command is executing.

hostname:path

If *path* is relative, it is taken to be relative to the user's home directory on the host named *hostname*.

If *path* is absolute, it is an absolute path on the host named *hostname*.

If *path* does not include a filename, the default filename has the form *<job ID>.OU*

If the -o option is not specified, PBS copies the standard output to the current working directory where the qsub command was executed, and writes standard output to the default filename, which has this form:

<job name>.o<sequence number>

If you use a UNC path, the hostname is optional. If you use a non-UNC path, the hostname is required.

This option is overridden by the -k option.

-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 *project*.

Format: *Project Name*; see "Project Name" on page 347

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 name>

Job is submitted to the specified queue at the default server.

@<server name>

Job is submitted to the default queue at the specified server.

<queue name>@<server name>

Job is submitted to the specified queue at the specified server.

Default: Default queue at default server

-r <y|n>

Declares whether the job is rerunnable. Sets job's Rerunable attribute to the argument value. Does not affect how the job is handled in the case where the job was unable to begin execution.

Format: Single character, "*y*" or "*n*"

Table 2-36: Sub-options to r Option

Suboption	Meaning
y	Job is rerunnable.
n	Job is not rerunnable.

Default: "*y*"

Interactive jobs are not rerunnable. Job arrays are always rerunnable. See "qrerun" on page 173.

-R <remove options>

Specifies whether standard output and/or standard error files are automatically removed (deleted) upon job completion.

Sets the job's Remove_Files attribute to *remove options*. Overrides default path names for these streams. Overrides -o and -e options.

This attribute cannot be altered once the job has begun execution.

Default: *Unset*; neither is removed

The *remove options* argument can take the following values:

Table 2-37: discard Argument Values

Option	Meaning
e	The standard error stream is removed (deleted) upon job completion
o	The standard output stream is removed (deleted) upon job completion
eo, oe	Both standard output and standard error streams are removed (deleted) upon job completion
unset	Neither stream is removed.

-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 hostname. Only one path may be specified per named host. The path selected is the one whose hostname is that of the server on which the job resides.

Format: *<path>[@<hostname>][,<path>@<hostname> ...]*

Default: User's login shell on execution host

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 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 hostname. Only one username may be specified per named host. The server on which the job resides will select first the username whose hostname is the same as the server name. Failing that, the next selection is 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.

Format of *user list: <username>[@<hostname>][,<username>@<hostname> ...]*

Default: Job owner (username on submission host)

-v <variable list>

Specifies environment variables and shell functions 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 variables in *variable list* and their values. See section 2.60.7, "Environment Variables", on page 223.

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

All environment variables and shell functions in the user's login environment where `qsub` is run are exported to the job. The job's Variable_List attribute is appended with all of these environment variables and their values.

-W <additional attributes>

The `-W` option allows specification of some job attributes. 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 a *value* string, it must be enclosed with single quotes or double quotes.

The following attributes can be set using the `-W` option only:

block=true

The `qsub` command 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 2.60.8, "Exit Status", on page 224.

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, and has the form:

<job ID>[:<job ID> ...]

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 2.60.8.1, "Warning About Exit Status with csh", on page 225.

afternotok: <arg list>

This job may be scheduled for execution only after all jobs in *arg list* have terminated with errors. See section 2.60.8.1, "Warning About Exit Status with csh", on page 225.

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 deleted without ever having been run.

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 2.60.8.1, "Warning About Exit Status with csh", on page 225.

beforenotok: <arg list>

If this job terminates execution with errors, jobs in *arg list* may begin. See section 2.60.8.1, "Warning About Exit Status with csh", on page 225.

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 performed after the job is queued. If an error is detected, the new job is deleted by the server. Mail is 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=<group list>

List of group names. Job is run under a group name from this list. Sets job's group_List attribute to *group list*.

Only one group name may be specified without a hostname. Only one group name may be specified per named host. The server on which the job resides will select first the group name whose hostname is the same as the server name. Failing that, the next selection is the group name with no specified hostname. The group names on the server and execution hosts must be the same. The job submitter's primary group is automatically added to the list.

Under Windows, the primary group is the first group found for the user by PBS when it queries the accounts database.

Format of *group list*: *<group name>[@<hostname>][,<group name>@<hostname> ...]*

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.

release_nodes_on_stageout=<value>

When set to *True*, all of the job's vnodes not on the primary execution host are released when stageout begins.

Cannot be used with vnodes managed by cpuset MoMs, (whose arch is linux_cpuset), or with vnodes tied to Cray X* series systems.

When cgroups is enabled and this is used with some but not all vnodes from one MoM, resources on those vnodes that are part of a cgroup are not released until the entire cgroup is released.

The job's stageout attribute must be set for the release_nodes_on_stageout attribute to take effect.

Format: *Boolean*

Default: *False*

run_count=<value>

Sets the number of times the server thinks it has run the job. Sets the value of the job's run_count attribute to *value*.

Format: Integer greater than or equal to zero

sandbox=<sandbox spec>

Determines which directory PBS uses for the job's staging and execution. Sets job's sandbox attribute to the value of *sandbox spec*.

Allowed values for *sandbox spec*:

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>

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 list*s. 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:

<file spec>[,<file spec>]

where *<file spec>* is

<execution path>@<hostname>:<storage path>

regardless of the direction of the copy. The name *execution 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 *execution path* from *storage path*.

The name *storage 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 *file spec*, i.e. it contains commas, it must be enclosed in double quotes.

If you use a UNC path, the *hostname* is optional. If you use a non-UNC path, the *hostname* is required.

umask=<mask value>

The umask with which the job is started. Sets job's umask attribute to *mask value*. Controls umask of job's standard output and standard error.

The following example allows group and world read of the job's output and error:

```
-W umask=33
```

Format: one to four digits; typically two

Default: *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 only with interactive jobs: must be used with one of the following:

```
-I
```

`-W interactive=true` (deprecated)

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.

2.60.4 Operands

The `qsub` command accepts as operands one of the following:

(no operands)

Same as with a dash. Any PBS directives and user tasks are read from the command line.

<script>

Path to script. Can be absolute or relative to current directory where qsub is run.

-

When you use a dash, any PBS directives and user tasks are read from the command line.

-- <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.

When you run qsub this way, it runs the executable directly. It does not start a shell, so no shell initialization scripts are run, and execution paths and other environment variables are not set. You should make sure that environment variables are set correctly.

2.60.5 Standard Output

Job ID for submitted job

If the job is successfully created

(No output)

If the -z option is set

2.60.6 Standard Error

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

2.60.7 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 when qsub is run.

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 when qsub is run.

PBS_JOBNAME

Job name specified by submitter. Created when qsub is run.

PBS_NODEFILE
: Name of file containing the list of vnodes assigned to the job. Created when `qsub` is run.

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 is taken from job submission, otherwise default queue.

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
: Timezone. 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 when `qsub` is run.

PBS_TMPDIR
: Pathname of scratch directory for PBS components. Set when PBS assigns it.

2.60.8 Exit Status

For non-blocking jobs:

Zero
: Upon successful processing of input

Greater than zero
: Upon failure of `qsub`

For blocking jobs:

Exit value of job
: When job runs successfully

3
: If the job is deleted without being run

2.60.8.1 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.

2.60.9 See Also

"Submitting a PBS Job", on page 11 of the PBS Professional User's Guide, "Job Attributes" on page 318, "Resources Built Into PBS" on page 258, and "Requesting Resources", on page 52 of the PBS Professional User's Guide.

2.61 qterm

Terminates one or both PBS servers, and optionally terminates scheduler(s) and/or MoMs

2.61.1 Synopsis

qterm [-f | -F | -i] [-m] [-s] [-t <type>] [<server name>[<server name> ...]]

qterm --version

2.61.2 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.

2.61.2.1 Required Privilege

In order to run the `qterm` command, you must have PBS Operator or Manager privilege.

2.61.3 Options to qterm

(no options)

The `qterm` command defaults to `qterm -t quick`.

-f

If the complex is configured for failover, shuts down both the primary and secondary servers.

Without the `-f` option, `qterm` shuts down the primary server and makes the secondary server active.

The `-f` option cannot be used with the `-i` or `-F` options.

-F

If the complex is configured for failover, shuts down only the secondary server, leaving the primary server active.

The `-F` option cannot be used with the `-f` or `-i` options.

-i

If the complex is configured for failover, leaves the secondary server idle when the primary server is shut down.

The `-i` option cannot be used with the `-f` or `-F` options.

-m

Shuts down the primary server and all MoMs (`pbs_mom`). This option does not cause jobs or subjobs to be killed. Jobs are left running subject to other options to the `qterm` command.

-s

Shuts down the primary server and the scheduler (`pbs_sched`).

-t <type>

immediate
: Shuts down the primary server. Immediately stops all running jobs. 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
: Shuts down the primary server. 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
: Shuts down the primary server. Running jobs and subjobs are left running.

  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.

2.61.4 Operands

You optionally specify the list of servers to shut down using *[<server name>[<server name> ...]]*.

If you do not specify any servers, the `qterm` command shuts down the default server.

2.61.4.1 Standard Error

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

2.61.4.2 Exit Status

Zero
: Upon successful processing of all operands presented to the `qterm` command

Greater than zero
: If the `qterm` command fails to process any operand

2.61.4.3 See Also

The PBS Professional Administrator's Guide, "pbs_server" on page 105, "pbs_sched" on page 103, "pbs_mom" on page 71

2.62 tracejob

Extracts and prints log messages for a PBS job

2.62.1 Synopsis

tracejob [-a] [-c <count>] [-f <filter>] [-l] [-m] [-n <days>] [-p <path>] [-s] [-v] [-w <cols>] [-z] <job ID>

tracejob --version

2.62.2 Description

The `tracejob` command extracts log messages for a given *job ID* and prints them in chronological order.

The `tracejob` command extracts information from server, default scheduler, accounting, and MoM logs. 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. If the job ran on multiple hosts, you must run `tracejob` on each of those hosts.

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.

2.62.3 Using tracejob on Job Arrays

If `tracejob` is run on a job array, the information returned is 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 are available as for a job. Certain log messages that occur for a regular job will not occur for a subjob.

2.62.4 Required Privilege

All users have access to server, scheduler, and MoM information. Only Administrator or root can access accounting information.

2.62.5 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. The following table shows each filter with its hex value and category:

Table 2-38: tracejob 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 number of days: *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 *cols* is not specified, tracejob queries OS to get terminal width. If OS doesn't return anything, defaults to *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.

2.62.6 Operands

The `tracejob` command accepts one *job ID* operand.

For a job, this has the form:

<sequence number>[.<server name>][@<server name>]

For a job array, the form is:

<sequence number>[][.<server name>][@<server name>]

For a subjob, the form is:

<sequence number>[<index>][.<server name>][@<server name>]

Note that some shells require that you enclose a job array identifier in double quotes.

2.62.7 Exit Status

Zero
: Upon successful processing of all options

Greater than zero
: If `tracejob` is unable to process any options

2.62.8 See Also

The PBS Professional Administrator's Guide

2.63 win_postinstall.py

For Windows. Configures PBS services

2.63.1 Synopsis

<PBS_EXEC>\etc\python win_postinstall.py -u <PBS service account> -p <PBS service account password> [-t server]

<PBS_EXEC>\etc\python win_postinstall.py -u <PBS service account> -p <PBS service account password> -t <non-server installation type> -s <server name>

2.63.2 Description

The `win_postinstall.py` command configures PBS services. It performs post-installation steps such as validating the PBS service account username and password, installing the Visual C++ redistributable binary, creating `PBS_HOME` and the default queue, initializing the database, and registering and starting the PBS_SERVER, PBS_SCHED, PBS_MOM, PBS_COMM and PBS_RSHD PBS services.

When you use this command during installation of PBS, the command automatically un-registers any old PBS services.

Available on Windows only.

2.63.2.1 Required Privilege

You must have Administrator privilege to run this command.

2.63.3 Options to win_postinstall.py

-p, --passwd <PBS service account password>
: Specifies password for PBS service account.

-s, --server <server name>
: Specifies the hostname on which the PBS server will run; required when the installation type is one of "execution", "client", or "comm".

-t, --type <installation type>
: Specifies type of installation. Type can be one of "server", "execution", "client", or "comm". Default is server installation.

-u, --user <PBS service account>
: Specifies PBS service account. When you specify the PBS service account, whether or not you are on a domain machine, include only the username, not the domain. For example, if the full username on a domain machine is *<domain>\<username>*, pass only *<username>* as an argument.

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 for MoM's partition. 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 hash-mark ("#").

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\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

3.2.1 Replacing Actions

$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.

3.2.2 MoM Parameters

$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_jitter <maximum jitter>

Cray only. PBS sends requests to ALPS to release a finished job at intervals specified by the sum of $alps_release_wait_time and a randomly generated value between zero and *maximum jitter*, in seconds.

Format: *Float*

Default: 0.12 seconds

$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 suspectbegin.

Format: *Seconds, specified as positive integer*

Default: *600* (10 minutes)

$alps_release_wait_time <wait time>

Cray only. PBS sends requests to ALPS to release a finished job at intervals specified by the sum of *wait time* and a randomly generated value between zero and the maximum specified in $alps_release_jitter, in seconds.

Format: *Float*

Default: 0.4 seconds

$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. See "Specifying Checkpoint Path" on page 424 in the PBS Professional Administrator's Guide.

$clienthost <hostname>

hostname is added to the list of hosts which are allowed to connect to MoM as long as they are using a privileged port. For example, this allows 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`, "`localhost`" and the name returned to MoM by the system call `gethostname()`. These hostnames do not need to be added to the MoM 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_create_flags <flags>

Lists the flags for when MoM does a `cpusetCreate(3)` for each job. *flags* is an or-ed list of flags. Flags for an HPE SGI machine with supported versions of HPE MPI:

CPUSET_CPU_EXCLUSIVE | 0

Default: *0*

cpuset_destroy_delay <delay>

MoM waits up to *delay* seconds before destroying a cpuset of a just-completed job, but not longer than necessary. This gives the operating system more time to clean up leftover processes after they have been killed. Example:

```
cpuset_destroy_delay 10
```

Format: *Integer*

Default for HPE SGI systems: *0*

$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
```

$enforce <limit>

MoM will enforce the given *limit*. Some limits have associated values. Syntax:

$enforce <variable name> <value>

$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.

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
```

$job_launch_delay

When the primary MoM gets a job whose tolerate_node_failures attribute is set to *all* or *job_start*, the primary MoM can wait to start the job (running the job script or executable) for up to a configured number of seconds. During this time, execjob_prologue hooks can finish and the primary MoM can check for communication problems with sister MoMs. You configure the number of seconds for the primary MoM to wait for hooks via the job_launch_delay configuration parameter in MoM's config file:

```
$job_launch_delay <number of seconds to wait>
```

Default: the sum of the values of the alarm attributes of any enabled execjob_prologue hooks. If there are no enabled execjob_prologue hooks, the default value is 30 seconds. For example, if there are two enabled execjob_prologue hooks, one with alarm = 30 and one with alarm = 60, the default value of MoM's job_launch_delay is 90 seconds.

After all the execjob_prologue hooks have finished, or MoM has waited for the value of the job_launch_delay parameter, she starts the job.

$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.

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*. See "Log Levels" on page 540 in the PBS Professional Administrator's Guide.

Default: 975

$max_check_poll <seconds>

Maximum time between polling cycles, in seconds. See "Configuring MoM Polling Cycle" on page 48 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 vnode 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 *load* to a value that is slightly higher than the number of CPUs, for example *.25 + ncpus*.

Example:

```
$max_load 3.5
```

Format: *Float*

Default: number of CPUs on machine

$max_poll_downtime <downtime>

When mother superior detects that a sister mom has lost connectivity (e.g. MoM went down or the network is having problems) it waits *downtime* seconds for the sister to reconnect before it gives up and kills the job.

Format: *Integer*

Default: *five minutes*

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*

$min_check_poll <seconds>

Minimum time between polling cycles, in seconds. Must be greater than zero and less than $max_check_poll. See "Configuring MoM Polling Cycle" on page 48 in the PBS Professional Administrator's Guide. Minimum recommended value: *10 seconds*.

Format: *Integer*

Minimum value: *1 second*

Default: *10 seconds*

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. Cray only. Requires CLE 5.2.

Default: "*true*"; enabled

$prologalarm <timeout>

Defines the maximum number of seconds the prologue and epilogue may run before timing out.

Example:

```
$prologalarm 30
```

Format: *Integer*

Default: *30 seconds*

$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. However, MoM will run previously-acquired hooks that run as root.

Format: *Boolean*

Default: *False*

$restart_background <True|False>

Controls how MoM runs a restart script after checkpointing a job. When this option is set to *True*, MoM forks a child which runs the restart script. The child returns when all restarts for all the local tasks of the job are done. MoM does not block on the restart. When this option is set to *False*, MoM runs the restart script and waits for the result.

Format: *Boolean*

Default: *False*

$restart_transmogrify <True | False>

Controls how MoM runs a restart script after checkpointing a job.

When this option is set to *True*, MoM runs the restart script, replacing the session ID of the original task's top process with the session ID of the script.

When this option is set to *False*, MoM runs the restart script and waits for the result. The restart script must restore the original session ID for all the processes of each task so that MoM can continue to track the job.

When this option is set to *False* and the restart uses an external command, the configuration parameter restart_background is ignored and treated as if it were set to *True*, preventing MoM from blocking on the restart.

Format: *Boolean*

Default: *False*

$restrict_user <True | False>

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 *True* 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 are 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
```

$sister_join_job_alarm

When the primary MoM gets a job whose tolerate_node_failures attribute is set to *all* or *job_start*, the primary MoM can wait to start the job for up to a configured number of seconds if the sister MoMs do not immediately acknowledge joining the job. This gives the sister MoMs more time to join the job. You configure the number of seconds for the primary MoM to wait for sister MoMs via the sister_join_job_alarm configuration parameter in MoM's config file:

```
$sister_join_job_alarm <number of seconds to wait>
```

Default: the sum of the values of the alarm attributes of any enabled execjob_begin hooks. If there are no enabled execjob_begin hooks, the default value is 30 seconds. For example, if there are two enabled execjob_begin hooks, one with alarm = 30 and one with alarm = 20, the default value of MoM's sister_join_job_alarm is 50 seconds.

After all the sister MoMs have joined the job, or MoM has waited for the value of the $sister_join_job_alarm parameter, she starts the job.

$suspendsig <suspend signal> [resume signal]

Alternate signal *suspend signal* is used to suspend jobs instead of SIGSTOP. Optional *resume signal* is used to resume jobs instead of SIGCONT.

$tmpdir <directory>

Location where each job's scratch directory will be created.

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 Linux, writable by *Everyone* on Windows.

Example:

```
$tmpdir /memfs
```

Default on Linux: `/var/tmp`

Default on Windows: value of the TMP environment variable

$usecp <hostname:source directory> <destination directory>

MoM uses `/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 directory* can be replaced with the *destination directory* on *hostname*. Both *source directory* and *destination directory* are absolute pathnames of directories, not files.

Overrides PBS_RCP and PBS_SCP.

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*.

PBS automatically sets the value of the $vnodedef_additive MoM configuration option to *False* on any MoM on a login node.

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 *factor*. For example:

```
$wallmult 1.5
```

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
```

4
Scheduler Parameters

This chapter lists scheduler configuration parameters. These parameters are found in each scheduler's configuration file, *PBS_HOME/sched_priv/sched_config*.

4.1 Format of Scheduler 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 primetime 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

Deprecated. Use the backfill_depth queue/server attribute instead. Toggle that controls whether PBS uses backfilling. If this is set to *True*, this 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. This 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* and help_starving_jobs is *True*, this scheduler backfills around starving jobs.

Can be used with strict_ordering and help_starving_jobs

Format: *Boolean*

Default: *True all*

backfill_prime

This 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. 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 "Examining Jobs Queue by Queue" on page 109 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 are considered for execution only when the system is in dedicated time as specified in the configuration file `PBS_HOME/sched_priv/dedicated_time`.

See "Dedicated Time" on page 127 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). Not a prime option.

See "Using Fairshare" on page 139 in the PBS Professional Administrator's Guide.

Format: *Boolean*

Default: *False all*

fairshare_decay_factor

Decay multiplier for fairshare usage reduction. Each decay period, the usage is multiplied by this value. Valid values: between 0 and 1, not inclusive. Not a prime option.

Format: *Float*

Default: 0.5

fairshare_decay_time

Time between fairshare usage decay operations. Not a prime option.

Format: *Duration*

Default: *24:00:00*

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*". Not a prime option.

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 parameter to be enabled. Not a prime option.

Format: *Boolean*

Default: *False*

fairshare_usage_res

Specifies the mathematical formula to use in fairshare calculations. Is composed of PBS resources as well as mathematical operators that are standard Python operators and/or those in the Python math module. When using a PBS resource, if resources_used.<resource name> exists, that value is used. Otherwise, the value is taken from Resource_List.<resource name>. Not a prime option.

See "Tracking Resource Usage" on page 142 in the PBS Professional Administrator's Guide.

Format: *String*

Default: *cput*

half_life

Deprecated (as of 13.0).

The half-life for fairshare usage; after the amount of time specified, the fairshare usage is halved. Requires that fair_share parameter be enabled. Not a prime option.

See "Using Fairshare" on page 139 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, no lower-priority jobs will run until the starving job can be run, unless backfilling is also used. To use this option, the max_starve configuration parameter needs to be set as well. See also max_starve, and the server's backfill_depth and eligible_time_enable attributes.

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

Specifies 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: "<resource name> HIGH|LOW"

job_sort_key: "fairshare_perc HIGH|LOW"

job_sort_key: "job_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 *resource name*:

Table 4-1: Special Sorting in job_sort_key

Special Sort	Description
fairshare_perc HIGH	Sort based on how much fairshare percentage the entity deserves, 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. For calculation, see "Computing Target Usage for Each Vertex (fairshare_perc)" on page 144 in the PBS Professional Administrator's Guide. This should only be used if entity share (strict priority) sorting is needed. See "Sorting Jobs by Entity Shares (Was Strict Priority)" on page 132 in the PBS Professional Administrator's Guide Incompatible with fair_share scheduling parameter being *True*.
job_priority HIGH|LOW	Sort jobs by the job priority attribute regardless of job owner.
sort_priority HIGH|LOW	**Deprecated**. See job_priority above.

The following example illustrates 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*, this 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 "Using Load Balancing" on page 158 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 "Using Load Balancing" on page 158 in the PBS Professional Administrator's Guide.

log_filter

Defines which event types to keep out of this 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 "Specifying Scheduler Log Events" on page 542 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.<resource name> where *resource name* is a site-defined resource. Each MoM is queried with the resource name and the return value is used to replace resources_available.<resource name> on that vnode. On a multi-vnoded machine with a natural vnode, all vnodes 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 name> | sort_priority <HIGH | LOW>

node_sort_key: <resource name> <HIGH | LOW> <total | assigned | unused>

where

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.

When sorting on a resource, the default third field is "*total*".

See "Sorting Vnodes on a Key" on page 224 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 are treated as non-primetime queues. Jobs in these queues run only during non-primetime. Primetime and non-primetime are defined in the `holidays` file.

See "Using Primetime and Holidays" on page 193 in the PBS Professional Administrator's Guide.

Format: *String*

Default: *np_*

peer_queue

Defines the mapping of a pulling queue to a furnishing queue for peer scheduling. Maximum number is 50 peer queues per scheduler.

See "Peer Scheduling" on page 167 in the PBS Professional Administrator's Guide.

Format: *String*

Default: Unset

preemptive_sched

Enables job preemption.

See preempt_order and "Using Preemption" on page 182 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 this scheduler uses 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("").

For example, PBS should first attempt to use suspension to preempt a job, and if that is unsuccessful, requeue the job:

```
preempt_order: "SR"
```

For example, if the job has between 100% and 81% of requested time remaining, first try to suspend the job, then try checkpoint, then requeue. If the job has between 80% and 51% of requested time remaining, attempt suspend, then checkpoint. Between 50% and 0% time remaining, just attempt to suspend the job:

```
preempt_order: "SCR 80 SC 50 S"
```

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. Any level not specified in the preempt_prio list is ignored.

Table 4-3: Preemption Levels

Level	Description
express_queue	Jobs in express queues preempt other jobs. See preempt_queue_prio. Does not require by_queue to be *True*.
starving_jobs	When a job becomes starving it can preempt other jobs.
fairshare	When the entity owning a job exceeds its fairshare limit.
queue_softlimits	Jobs which are over their queue soft limits
server_softlimits	Jobs which are over their server soft limits
normal_jobs	The preemption level into which a job falls if it does not fit into any other specified level.

For example, starving jobs have the highest priority, then normal jobs, and jobs whose entities are over their fairshare limit are third highest:

```
preempt_prio: "starving_jobs, normal_jobs, fairshare"
```

For example, starving jobs whose entities are also over their fairshare limit are lower priority than normal jobs:

```
preempt_prio: "normal_jobs, starving_jobs+fairshare"
```

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 to be *True*.

Format: *Integer*

Default: *150*

preempt_requeue

Deprecated. Add an "*R*" to preempt_order parameter.

preempt_sort

Specifies whether jobs most eligible for preemption are 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, meaning that this parameter is commented out, preempted job will be that with longest running time.

Must be commented out in order to be unset.

Allowable values: "*min_time_since_start*", or no preempt_sort setting.

See "Sorting Within Preemption Level" on page 189 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 in these queues run only during primetime. Primetime and non-primetime are defined in the `holidays` file.

See "Using Primetime and Holidays" on page 193 in the PBS Professional Administrator's Guide.

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 are not 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 primetime and non-primetime. This option is only meaningful if backfill_prime is *True*.

See also backfill_prime, prime_exempt_anytime_queues.

For example, non-primetime jobs can spill into primetime by 1 hour:

```
prime_spill: 1:00:00 prime
```

For example, jobs in either prime/non-prime can spill into the other by 1 hour:

```
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 PBS cannot satisfy the job's request using vnodes that already have the requested AOE instantiated, 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, or if Boolean are to be honored, 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.<resource name> on vnodes, queues, and the server. A scheduler considers numeric (integer or float) items as consumable resources and ensures that no more are assigned than are available (e.g. ncpus or mem). Any string resources are compared using string comparisons. If "host" is not added to the resources line, when the user submits a job requesting a specific vnode in the following syntax:

```
qsub -l select=host=vnodeName
```

the job will run on any host.

Format: *String*

Default: *ncpus, mem, arch, host, vnode*, *aoe*

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*, this scheduler considers one job from the first queue, then one job from the second queue, and so on in a circular fashion. 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 there are groups of queues with the same priority, and this parameter is set to *True*, this scheduler round-robins through each group of queues before moving to the next group.

If round_robin is set to *False*, this scheduler considers 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 this scheduler to replace the server's resources_available values with new values returned by a site-specific external program.

See "Dynamic Server-level Resources" on page 263 in the PBS Professional Administrator's Guide for details of usage.

Format: *String*

Default: Unset

smp_cluster_dist

Deprecated (12.2). 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 "SMP Cluster Distribution" on page 216 in the PBS Professional Administrator's Guide and "Using Load Balancing" on page 158 in the PBS Professional Administrator's Guide.

Format: *String*

Default: *pack all*

sort_by

Deprecated. Use job_sort_key.

sort_queues

Deprecated (12.2). 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.

The sorted order of queues is not taken into consideration unless by_queue is set to *True*.

See "Sorting Queues into Priority Order" on page 221 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 "FIFO with Strict Ordering" on page 150 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 "Using Fairshare" on page 139 in the PBS Professional Administrator's Guide.

Format: *Integer*

Default: The unknown group gets 0 shares

5 List of Built-in Resources

This chapter lists all of the built-in PBS resources. For information on setting, viewing, and using resources, see "Using PBS Resources" on page 227 in the PBS Professional Administrator's Guide.

5.1 Resource Data Types

Data types for resources are described in section 7.1, "List of Formats", on page 343.

5.2 Viewing Resource Information

See "Viewing Resource Information" on page 308 in the PBS Professional Administrator's Guide.

5.3 Resource Flags

Resource flags are described and listed in "Resource Accumulation Flags" on page 255 in the PBS Professional Administrator's Guide.

5.4 Attributes where Resources Are Tracked

Resources are tracked in the following attributes:

Table 5-1: Attributes Where Resources Are Tracked

Resource Being Tracked	Attribute Name			
	Server and Queue	Vnode	Job	Reservation
Amount of each resource available for use at the object (server, queue, vnode)	resources_available.<resource name>	resources_available.<resource name>		
Amount of each resource allocated to jobs running and exiting at the object (server, queue, vnode)	resources_assigned.<resource name>	resources_assigned.<resource name>		
Amount of each resource used by the job			resources_used.<resource name>	
Amount of each job-wide resource that is assigned to any job that does not explicitly request the resource	resources_default.<resource name>			
Amount of each host-level resource that is assigned to each chunk of any job where that does not explicitly request the resource	default_chunk.<resource name>			
List of resources requested by the object (job or reservation)			Resource_List.<resource name>	Resource_List.<resource name>
List of chunks for the job. Each chunk shows the name of the vnode from which it is taken along with the host-level, consumable resources allocated from that vnode.			exec_vnode	
List of vnodes and resources allocated to them to satisfy the chunks requested for this reservation or occurrence				resv_nodes

5.5 Resource Table Format

In the following tables, the columns contain the following information:

Name

The name of the resource

Description

A description of the resource's function

Format

The resource's format

Scope

Some resources are either:

- Job-wide and can be requested only outside of a select statement
- Host-level and can be requested only inside of a select statement

Consumable

A resource is consumable if use of this resource by a job reduces the amount available to other jobs

Val/Opt

If the resource can take only specific values or options, each is listed here

Value/Option Description

If the resource can take only specific values or options, the behavior of each value or option is described here

Default Value

The resource's default value, if any

Python Type

The resource's Python type

Platform

Platform where available

5.6 Resources Built Into PBS

Resources								
Name Description	Format	Scope	Consumable	Val / Opt	Value/Option Description	Default Value	Python Type	Platform
accelerator Indicates whether this vnode is associated with an accelerator. Used for requesting accelerators. On Cray, this resource exists only when there is at least one associated accelerator.	*Boolean*	Host-level	No	*True*	On Cray, this is set to *True* when there is at least one associated accelerator whose state is *UP*.	*False*	bool	
				False	On Cray, set to *False* when all associated accelerators are in state *DOWN*.			
accelerator_memory Indicates amount of memory for accelerator(s) associated with this vnode. 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 PBScray-seg=*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 A scheduler rounds all resources of type size up to the nearest kb.	*Size*	Host-level	Yes			No default	pbs.size	
accelerator_model Indicates model of the accelerator(s) associated with this vnode. On Cray, PBS sets this resource only on vnodes with at least one accelerator with state = *UP*.	*String*	Host-level	No			No default	str	

Resources								
Name Description	Format	Scope	Consumable	Val / Opt	Value/Option Description	Default Value	Python Type	Platform
aoe List of AOEs (Application Operating Environments) that can be instantiated on this vnode. Case-sensitive. An AOE is the environment that results from provisioning a vnode. Each job can request at most one AOE. Cannot be set on server's host.	*string array*	Host-level	No	Allowable values are site-dependent.		No default	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. The resources_available.arch resource is the value reported by MoM unless explicitly set by the administrator.	*String*	Host-level	No	*linux*	Linux	No default	str	Linux
				linux_cpuset	Linux with cpusets			Linux with cpusets
				XT	CLE			CLE
cput Amount of CPU time used by the job for all processes on all vnodes. Establishes a job-wide resource limit.	*Duration*	Job-wide	No			No default	pbs.duration	
energy The energy used by a job. Set by PBS.	*Float.* Units: *kWh*		Yes			No default		
eoe Stands for "Energy Operational Environment". When set on a vnode in resources_available.eoe, contains the list of available power profiles. When set for a job in Resource_List.eoe, can contain at most one power profile. (A job can request only one power profile.) Automatically added to `resources:` line in `sched_config`.	*string array*		No			For resources_available.eoe: *unset* For Resource_List.eoe: no default	str	

Resources								
Name Description	Format	Scope	Consumable	Val / Opt	Value/Option Description	Default Value	Python Type	Platform
exec_vnode The vnodes that PBS estimates this job will use. This is not the job's exec_vnode attribute. This appears only in job's estimated attribute. Cannot be requested for a job; used for reporting only. Read-only.	*String*					No default	str	
file Size of any single file that may be created by the job. A scheduler rounds all resources of type size up to the nearest kb.	*Size*	Job-wide				No default	pbs.size	
hbmem High-bandwidth memory. Available only on some architectures such as Xeon Phi KNL.	*Size*	Host-level	Yes		Values must be greater than or equal to zero.	No default	pbs.size	Xeon Phi KNL
host Name of execution host. Cannot be changed. Site-dependent.	*String*	Host-level				Automatically set to the short form of the hostname in the Mom attribute. On Cray compute node, set to *<mpp_host>_<nid>*.	str	
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 max_walltime is specified, min_walltime must also be specified. Cannot be used for resources_min or resources_max. Cannot be set on job arrays or reservations.	*Duration*	Job-wide	No	Must be greater than or equal to min_walltime.		*5 years*	pbs.duration	

Resources								
Name Description	Format	Scope	Consumable	Val / Opt	Value/Option Description	Default Value	Python Type	Platform
mem Amount of physical memory i.e. workingset allocated to the job, either job-wide or host-level. A scheduler rounds all resources of type size up to the nearest kb.	*Size*	Either job-wide or host-level. Can be requested only inside of a select statement.	Yes			No default	pbs.size	
min_walltime Minimum walltime allowed for a shrink-to-fit job. When min_walltime 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. Cannot be used for resources_min or resources_max. Cannot be set on job arrays or reservations.	*Duration*	Job-wide	No	Must be less than or equal to max_walltime.		No default	pbs.duration	
mpiprocs Number of MPI processes for this chunk. Cannot use sum from chunks as job-wide limit. 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.	*Integer*	Host-level				If ncpus >0: *1* Otherwise: *0*	int	

Resources								
Name Description	Format	Scope	Consumable	Val / Opt	Value/Option Description	Default Value	Python Type	Platform
naccelerators Number of accelerators on the host. PBS sets this resource to the number of accelerators with state = *UP*. On Cray, PBS sets this resource only on vnodes whose hosts have 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.naccelerators=1 On vnodeA_2_1: resources_available.naccelerators=@vnodeA_2_0	*Integer*	Host-level	Yes			No default	int	
nchunk 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. This resource can be used to specify the default number of chunks at the server or queue. Example: `set queue myqueue default_chunk.nchunk=2` This resource cannot be used in server and queue resources_min and resources_max.	*Integer*		No			*1*	int	

Resources								
Name Description	Format	Scope	Consumable	Val / Opt	Value/Option Description	Default Value	Python Type	Platform
ncpus Number of processors.	*Integer*	Host-level	Yes			No default	int	
nice Nice value with which the job is to be run. Host-dependent.	*Integer*	Job-wide				No default	int	
nodect **Deprecated**. Number of chunks in resource request from selection directive, or number of hosts requested from node specification. Read-only.	*Integer*	Job-wide				*1*	int	
nodes **Deprecated**. Number of hosts requested.	*Integer*					No default		
ompthreads Number of OpenMP threads for this chunk. Cannot use sum from chunks as job-wide limit. 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.	*Integer*	Host-level	No			Value of ncpus	int	
PBScrayhost Used to differentiate a Cray system, containing ALPS, login nodes running PBS MoMs, and compute nodes, from a separate Cray system with a separate ALPS.	*String*		No			Value of mpp_host for this system	str	Cray only

Resources								
Name Description	Format	Scope	Consumable	Val / Opt	Value/Option Description	Default Value	Python Type	Platform
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. Name format: *PBScraylabel_<label name>* For example, if the label name is *Blue*, the name of this resource is *PBScraylabel_Blue*.	*Boolean*					PBS sets the value of the resource to *True* on all vnodes representing the compute node.	bool	Cray only
PBScraynid Tracks the node ID of the associated compute node. All vnodes representing a particular compute node share a value for PBScraynid.	*String*		No			The value of PBScraynid is set to the value of node_id for this compute node.	str	Cray only
PBScrayorder Tracks 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. Do not use this resource in a resource request.	*Integer*		No			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.	int	Cray only
PBScrayseg Not used.	*String*					No default	str	Cray only
pcput Amount of CPU time allocated to any single process in the job. Establishes a per-process resource limit.	*Duration*	Job-wide	No			No default	pbs.duration	

Resources								
Name Description	Format	Scope	Consumable	Val / Opt	Value/Option Description	Default Value	Python Type	Platform
pmem Amount of physical memory (workingset) for use by any single process of the job. Establishes a per-process resource limit. A scheduler rounds all resources of type size up to the nearest kb.	*Size*	Job-wide	No			No default	pbs.size	
preempt_targets List of resources and/or queues. Jobs requesting those resources or in those queues are preemption targets.	*string array* Syntax: *preempt_targets="Queue=<queue name>[,Queue=<queue name>],Resource_List.<resource>=<value>[,Resource_List.<resource>= <value>]"* or *preempt_targets=None* Keywords “queue” and “none” are case-insensitive. You can list multiple comma-separated targets.	Job-wide	No			No default	str	
pvmem Amount of virtual memory for use by any single process in the job. Establishes a per-process resource limit. A scheduler rounds all resources of type size up to the nearest kb.	*Size*	Job-wide	No			No default	pbs.size	
site Arbitrary string resource.	*String*	Job-wide	No			No default	str	
software Site-specific software specification.	*String*	Job-wide		Allowable values and effect on job placement are site-dependent.		No default	pbs.software	

Resources								
Name Description	Format	Scope	Consumable	Val / Opt	Value/Option Description	Default Value	Python Type	Platform
soft_walltime Soft limit on walltime. Similar to walltime, but cannot be requested by unprivileged users, and job is not killed if it exceeds its soft_walltime. A job's soft_walltime cannot exceed its walltime. Can be set by Manager only.	*Duration*					No default	pbs.duration	
start_time The estimated start time for this job. Cannot be requested for a job; used for reporting only. Appears only in job's estimated attribute. Read-only.	*Integer*					No default	int	
vmem Amount of virtual memory for use by all concurrent processes in the job. Establishes a per-chunk resource limit. A scheduler rounds all resources of type size up to the nearest kb.	*Size*	Host-level	Yes			No default	pbs.size	
vnode Name of virtual node (vnode) on which to execute. Site-dependent. See "Vnode Attributes" on page 311 of the PBS Professional Reference Guide.	*String*	Host-level				No default	str	
vntype 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. On CLE, automatically added to `resources:` line in `sched_config`. Must be manually added to resources: line when your server/scheduler runs on non-CLE host.	*string array*	Host-level	No	*cray_compute* *cray_login*	This vnode represents part of a compute node. This vnode represents a login node.	Automatically set for Cray; none for non-Cray	str	all

Resources								
Name Description	Format	Scope	Consumable	Val / Opt	Value/Option Description	Default Value	Python Type	Platform
walltime Amount of wall-clock time. Establishes a job-wide resource limit. Actual elapsed time may differ from walltime during Daylight Savings transitions.	*Duration*	Job-wide	No			5 years	pbs.duration	

6 Attributes

This chapter lists all of the supported PBS attributes. Attributes are listed by the PBS object they modify. For example, all supported attributes of jobs are listed in section 6.11, “Job Attributes”, on page 318. Attributes are case-sensitive.

6.1 Attribute Behavior

- 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.
- When an attribute is unset, it behaves as if it is at its default value.

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 "Choosing Configuration Method" on page 44 in the PBS Professional Administrator’s Guide. The following are the instructions for setting all other attributes.

To set the value of a non-string_array 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 set or change the value of a string_array attribute, use the `qmgr` command, either from the command line or within `qmgr`:

```
qmgr -c "set <object> <attribute> = <value>"
qmgr -c 'set <object> <attribute> = "<value,value>"'
qmgr -c 'set <object> <attribute> += <value>'
qmgr -c 'set <object> <attribute> -= <value>'
Qmgr: set <object> <attribute> = <value>
Qmgr: set <object> <attribute> = '<value,value>'
Qmgr: 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 "qmgr" on page 146.

6.3 Viewing Attribute Values

If you want to view attribute values, the following commands are helpful:

`qstat`; see section 2.58, "qstat", on page 192

`qmgr`; see section 2.48, "qmgr", on page 146

`pbs_rstat`; see section 2.33, "pbs_rstat", on page 94

- 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:
  ```
  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 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

Table 6-1: User, Operator, Manager Actions

Symbol	Explanation
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								
Name Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
acl_host_enable 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_host_moms_enable Specifies whether all MoMs are automatically allowed to contact the server with the same privilege as hosts listed in the acl_hosts server attribute.	*Boolean*	*True*	All MoMs are automatically allowed to contact the server with the same privilege as hosts listed in the acl_hosts server attribute.	*False*	bool	r	r	r, w
		False	MoMs are not automatically allowed to contact the server with the same privilege as hosts listed in the acl_hosts server attribute.					
acl_hosts List of hosts from which services can be requested of this server. Requests from the server host are 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.* Syntax: *"[+|-]<host-name>.<domain>[, ...]"*			No default	pbs.acl	r	r	r, w
acl_resv_group_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
acl_resv_groups 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.* Syntax: *"[+|-]<group name>[, ...]"*				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

Server Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
acl_resv_hosts 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.* Syntax: *"[+|-]<host-name>.<domain>[, ...]"*			No default	pbs.acl	r	r	r, w
acl_resv_user_enable Specifies whether the server limits which users are allowed 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
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.* Syntax:*"[+|-]<user-name>[@<host-name>][, ...]"*			No default	pbs.acl	r	r	r, w
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 Administrator, 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.* Syntax:*"[+|-]<user-name>[@<host-name>][, ...]"*			No default; no root jobs allowed	pbs.acl	r	r	r
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.* Syntax:*"[+|-]<user-name>[@<host-name>][, ...]"*			No default	pbs.acl	r	r	r, w
backfill_depth Specifies backfilling behavior. Sets the number of jobs that are to be backfilled around. Overridden by backfill_depth queue attribute. Recommendation: set this to less than 100.	*Integer.* Must be >=0	*>=0*	PBS backfills around the specified number of jobs.	Unset. When unset, backfill depth is *1*	int	r	r, w	r, w
		Unset	Backfill depth is set to *1.*					

Server Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
comment Informational text. Can be set by a scheduler or other privileged client.	*String* of any form			No default	str	r	r, w	r, w
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*. Syntax: *default_chunk.<resource name>=<value>,default_chunk.<resource name>=<value>, ...*			No default	pbs.pbs_resource Syntax: *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.* Syntax: *"-Wsuppress_mail=<N>"*			No default	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*. Syntax: *"<option> <value> <option> <value>", e.g. "-r y -N MyJob"* To add to existing: `Qmgr: s s default_qsub_arguments +="<option> <value>"`			No default	pbs.args	r	r, w	r, w
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*			*workq*	pbs.queue	r	r, w	r, w

Server Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
eligible_time_enable Controls starving behavior. Toggles between using the value of the job's eligible_time attribute and the value of now() - etime to evaluate whether job is starving.	*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 **Obsolete**. No longer used.								
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 name>`. 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*; authorization 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.					
FLicenses The number of floating licenses currently available for allocation to unlicensed CPUs. One license is required for each virtual CPU.	*Integer*			No default	int	r	r	r

Server Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
job_history_duration The length of time PBS will keep each job's history.	*Duration*			*Two weeks*	pbs.duration	r	r	r, w
job_history_enable Enables job history management. Setting this attribute to *True* enables job history management.	*Boolean*			*False*	bool	r	r	r, w
job_requeue_timeout The amount of time that can be taken while requeueing a job. Minimum allowed value: 1 second. Maximum allowed value: 3 hours.	Duration			*45 seconds*	pbs.duration	r	r, w	r, w
job_sort_formula Formula for computing job priorities. Described in the *PBS Professional Administrator's Guide*. If the attribute job_sort_formula is set, all schedulers use the formula in it to compute job priorities. When this 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.	*String*. Syntax: mathematical formula; can be made up of expressions, where expressions contain terms which are added, subtracted, multiplied, or divided, and which can contain parentheses, exponents, unary plus and minus, the ternary operator, and Python math module functions.			Unset	pbs.job_sort_formula	r	r	r, w
jobscript_max_size Limit on the size of any job script.	*size* Units default to bytes			*100MB*	pbs.size	r	r	r, w

Server Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
license_count The license_count attribute contains the following elements with their values: Avail_Global, Avail_Local, Used, High_Use, Avail_Sockets, Unused_Sockets, Avail_Nodes, Unused_Nodes When this attribute is displayed, it shows only the elements that are relevant to the type of licensing being used. For example, if you are using node licenses, it shows only Avail_Nodes and Unused_Nodes. When no licensing is set, displays node license info.	*String.* Syntax: *Avail_Global:<value> Avail_Local:<value> Used:<value> High_Use:<value> Avail_Sockets:<value> Unused_Sockets:<value> Avail_Nodes:<value> Unused_Nodes:<value>*	Avail_Global	The number of PBS CPU licenses still kept by the Altair license server (checked in.)	*Avail_Nodes:0 Unused_Nodes:0*	pbs.license_count	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_Sockets	The total number of socket licenses in the socket license file.					
		Unused_Sockets	The number of unused socket licenses.					
		Avail_Nodes	The total number of node licenses in the node license file.					
		Unused_Nodes	The number of unused node licenses.					
log_events The types of events the server logs.	*Integer* representation of bit string			*511*	int	r	r, w	r, w
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

Server Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
managers List of PBS Managers.	*String*. Syntax: *"<user-name>@<host-name>.<subdomain>.<domain>[,<user name>@<host-name>.<subdo-main>.<domain> ...]"*. The host, sub-domain, or domain name may be wild-carded with an aster-isk (*).			*Root on the server host*	pbs.acl	r	r	r, w
max_array_size The maximum number of subjobs allowed in any array job.	*Integer*			*10000*	int	r	r, w	r, w
max_concurrent_provision The max_concurrent_provision attribute is the number of vnodes allowed to be in the process of being provisioned. Cannot be set to zero. When unset, default value is used.	*Integer*	>0		5	int	r	r	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 this PBS complex.	*String*. Syntax: *max_group_res.<resource name>=<value>*	Any PBS resource, e.g. "ncpus", "mem", "pmem"		No default	pbs.pbs_resource Syntax: *max_group_res["<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_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*. Syntax: *max_group_res_soft.<resource name>=<value>*	Any PBS resource, e.g. “ncpus”, “mem”, “pmem”, etc.		None	pbs.pbs_resource Syntax: *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 default	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 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 eligible to be preempted by jobs from groups who are not over their soft limit.	*Integer*			No default	int	r	r, w	r, w
max_job_sequence_id Maximum value of sequence number in a job ID, job array ID, or reservation ID. Minimum allowed is *9999999*. Maximum allowed is *999999999999*. After specified maximum for sequence number has been reached, job IDs start again at *0*.	*Integer*			*9999999*	int	r	r	r, w
max_queued Limit attribute. The maximum number of jobs allowed to be queued or 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 343.			No default	str	r	r, w	r, w

Server Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
max_queued_res Limit attribute. The maximum amount of the specified resource allowed to be allocated to jobs queued or 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 343. Syntax: *max_queued_res.<resource name> = <.limit>*			No default	pbs.pbs_resource Syntax: *max_queued_res["<resource name>"]=<value>* where *resource name* is any built-in or custom resource	r	r, w	r, w
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 343.			No default	str	r	r, w	r, w
max_run_res 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 343 Syntax: *max_run_res.<resource name> = <.limit>*			No default	pbs.pbs_resource Syntax: *max_run_res["<resource name>"]=<value>* where *resource name* is any built-in or custom resource	r	r, w	r, w
max_run_res_soft 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 343 *max_run_res_soft.<resource name> = <.limit>*			No default	pbs.pbs_resource Syntax: *max_run_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_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 343.			No default	str	r	r, w	r, w
max_running Old limit attribute. Incompatible with new limit attributes. The maximum number of jobs in this complex allowed to be running at any given time.	*Integer*			No default	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.* Syntax: *max_user_res.<resource name>=<value>*	Any PBS resource, e.g. “ncpus”, “mem”, “pmem”, etc.		No default	pbs.pbs_resource Syntax: *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 this 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.* Syntax: *max_user_res_soft.<resource name>=<value>*	Any valid PBS resource, e.g. “ncpus”, “mem”, “pmem”, etc		No default	pbs.pbs_resource Syntax: *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 allowed to be running within this complex at one time.	*Integer*			No default	int	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_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*			No default	int	r	r, w	r, w
node_fail_requeue Controls whether running jobs are automatically requeued or are deleted when the primary execution host 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.* Units: *Seconds.*	*<0*	Behaves as if set to *1*.	*310*	int	r	r, w	r, w
		0	Jobs are not requeued; they are left in the *Running* state until the execution host is recovered.					
		>0	When the host has been down for the specified number of seconds, jobs are requeued if they are marked as rerunnable, or are deleted.					
		Unset	Behaves as if set to default value of *310*.					
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, separate them with commas and enclose the value in double quotes.			Unset	pbs.node_group_key	r	r, w	r, w

Server Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
operators List of PBS Operators.	*String.* Syntax: *<user-name>@<host-name>.<subdomain>.<domain name>[,<user-name>@<host-name>.<subdomain>.<domain name> ...]*. The host, subdomain, or domain name may be wildcarded with an asterisk (*).			No default	pbs.acl	r	r	r, w
pbs_license_file_location **Deprecated**. Do not use.	-	-	-	-	-	-	-	-
pbs_license_info Location of license information. Can be port number and hostname of license server, or local pathname to the actual license file associated with a license server.	*String*. Port number and hostname 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. Delimiter between items is colon (":") for Linux and semicolon (";") for Windows.			No default	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.* Units: seconds.			*31536000 seconds (1 year).*	pbs.duration	r	r	r, w

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. If set to zero or unset, PBS automatically sets the value to *1*.	*Integer*			*1*	int	r	r	r, w
pbs_version The version of PBS for this server.	*String*			No default	pbs.version	r	r	r
power_provisioning Reflects use of power profiles via PBS. Set by PBS to *True* when PBS_power hook is enabled.	*Boolean*	*True* *False*	Power provisioning is enabled. Power provisioning is disabled.	*False*	bool	r	r	r, w
python_restart_max_hooks The maximum number of hooks to be serviced before the Python interpreter is restarted. If this number is exceeded, and the time limit set in python_restart_min_interval has elapsed, the Python interpreter is restarted.	*Integer*			*100*	int	r	r	r, w
python_restart_max_objects The maximum number of objects to be created before the Python interpreter is restarted. If this number is exceeded, and the time limit set in python_restart_min_interval has elapsed, the Python interpreter is restarted.	*Integer*			*1000*	int	r	r	r, w
python_restart_min_interval The minimum time interval before the Python interpreter is restarted. If this interval has elapsed, and either the maximum number of hooks to be serviced (set in python_restart_max_hooks) has been exceeded or the maximum number of objects to be created (set in python_restart_max_objects) has been exceeded, the Python interpreter is restarted.	*Integer.* Units: *Seconds* or *[[HH:]MM:]SS* (duration)			*30*	pbs.duration	r	r	r, w

Server Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
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.	On installation: *True* After being unset: *False*	bool	r	r	r, w
queued_jobs_threshold Limit attribute. The maximum number of jobs allowed to be queued 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 343.			No default	str	r	r, w	r, w
queued_jobs_threshold_res Limit attribute. The maximum amount of the specified resource allowed to be allocated to jobs queued 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 343. *queued_jobs_threshold_res.<resource name> = <.limit>*			No default	pbs.pbs_resource Syntax: *queued_jobs_threshold_res["<resource name>"]=<value>* where *resource name* is any built-in or custom resource	r	r, w	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.* Units: *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.* Units: *Seconds*			*7200 (2 hours)*	int	-	-	r, w

Server Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
resources_assigned The total of each type of resource allocated to jobs running and exiting in this complex, plus the total of each type of resource allocated to any reservation. Reservation resources are added when the reservation starts.	*String*. Syntax: *resources_assigned.<resource name>=<value>[,resources_assigned.<resource name>=<value>,...]*			No default	pbs.pbs_resource Syntax: *resources_assigned["<resource name>"]=<value>* where *resource name* is any built-in or custom resource	r	r	r
resources_available The list of available resources and their values defined on the server. Each resource is listed on a separate line.	*String.* Syntax: *resources_available.<resource name>=<value>*			No default	pbs.pbs_resource Syntax: *resources_available["<resource name>"]=<value>* where *resource name* is any built-in or custom resource	r	r, w	r, w
resources_cost No longer used.						-	-	-
resources_default 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 name> server attribute.	*String* Syntax: *resources_default.<resource name>=<value>[,...]*			No limit	pbs.pbs_resource Syntax: *resources_default["<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
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* Syntax: *resources_max.<resource name>=<value>[, ...]*			No limit	pbs.pbs_resource Syntax: *resources_max["<resource name>"]=<value>* where *resource name* is any built-in or custom resource	r	r, w	r, w
restrict_res_to_release_on_suspend Comma-separated list of consumable resources to be released when jobs are suspended. If unset, all consumable resources are released on suspension.	*String_array* Syntax: comma-separated list			unset	Python list	r	r	r, w
resv_enable Specifies whether or not advance and standing reservations can be created in this complex.	*Boolean*		When set to *True*, new reservations can be created. When changed from *True* to *False*, new reservations cannot be created, but existing reservations are honored.	*True*	bool	r	r	r, w
resv_post_processing_time The amount of time allowed for reservations to clean up after running 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_highwater The maximum number of messages.	*Integer*	Greater than or equal to one		*1024*	int	r	r	r, w
rpp_max_pkt_check Maximum number of TPP messages processed by the main server thread per iteration.	Integer			1024	int	r	r	r, w
rpp_retry In a fault-tolerant setup (multiple `pbs_comm`s), when the first `pbs_comm` fails partway through a message, this is number of times TPP tries to use the first `pbs_comm`.	*Integer*	Greater than or equal to zero		*10*	int	r	r	r, w

Server Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
scheduler_iteration The time between scheduling iterations.	*Integer.* Units: *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
server_host The name of the host on which the active server is running. If the secondary server takes over, this attribute is set to the name of the secondary server's host. When the primary server takes control again, this attribute shows the name of the primary server's host.	*String*. Syntax: *<host-name>.<domain name>* If the server is listening to a non-standard port, the port number is appended, with a colon, to the hostname: *<host-name>.<domain name>:<port number>*			No default	str	r	r	r

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*	*Active*	The server is running. The scheduler is not in a scheduling cycle.	No default	Server state constant pbs.SV_STATE_ACTIVE	r	r	r
		Hot_Start	The server will run first any jobs that were running when it was shut down.		Server state constant pbs.SV_STATE_HOT			
		Idle	The server is running. Scheduling has been turned off.		Server state constant pbs.SV_STATE_IDLE			
		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			
single_signon_password_enable Only used on systems requiring passwords, such as Windows. 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.	Linux: *False* Windows: *True*	bool	r	r, w	r, w
		False	Users submitting jobs must specify a password for each job.					
state_count List of the number of jobs in each state in the complex. Suspended jobs are counted as running.	*String.* Syntax: *transiting=<value>, queued=<value>, ...*			No default	pbs.state_count	r	r	r

Server Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
system_cost No longer used.						-	-	-
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*			No default	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
comment For certain scheduler errors, PBS sets the scheduler's comment attribute to specific error messages. You can use the comment attribute to notify another administrator of something, but PBS does overwrite the value of comment under certain circumstances.	*String*							
do_not_span_psets Specifies whether or not this 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	This scheduler first attempts to place the job in a single 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 there is no existing placement set, occupied or empty, into which the job could fit, the job runs regardless of placement sets, running on whichever vnodes can satisfy the job's resource request.					
job_sort_formula_threshold Lower bound for calculated priority for job. If job priority is at or below this value, the job is not eligible to run in the current scheduler cycle.	*Float*			No default	float	-	r	r, w
only_explicit_psets Specifies whether placement sets are created for unset resources.	*Boolean*	*True*	Placement sets are not created from vnodes whose value for a resource is unset.	*False*		r	r, w	r, w
		False	Placement sets are created from vnodes whose value for a resource is unset.					
opt_backfill_fuzzy Sets the trade-off between scheduling cycle speed and granularity of estimated start time calculation.	*String*	*off*	Finest granularity, no speedup	*unset*; behaves like *low*	None	r	r	r, w
		low	Fairly fine granularity, some speedup					
		medium	Medium granularity, medium speedup					
		high	Coarse granularity, greatest speedup					

Scheduler Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
partition Comma-separated list of named partitions for which this scheduler is to run jobs. Cannot be set on default scheduler.	*String_array*			"*None*"				
pbs_version The version of PBS for this scheduler.	*String*			No default	None	-	r	r
scheduler_iteration Time in seconds between scheduling iterations. If you set the server's scheduler_iteration attribute, that value is assigned to the default scheduler's scheduler_iteration attribute, and vice versa.	*Integer.* Units: *Seconds*			*600*				
scheduling Enables scheduling of jobs. If you set the server's scheduling attribute, that value is assigned to the default scheduler's scheduling attribute, and vice versa.	*Boolean*			For default scheduler: *True* For multischeds: *False*				
sched_cycle_length This 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
sched_host The hostname of the machine on which this scheduler runs. Cannot be set on default scheduler; value for default scheduler is server hostname. Must be set by administrator.	*String*			*Server s host*	None	-	r	r
sched_log Directory where this scheduler writes its logs. Permissions should be *755*. Must be owned by root. Cannot be shared with another scheduler. Use default value; do not set.	*String*			*$PBS_HOME/sched_logs_<scheduler name>*				
sched_port Port on which this scheduler listens. Cannot be set on default scheduler. Must be set by administrator.	*String*			No default				
sched_preempt_enforce_resumption Controls whether this scheduler treats preempted jobs as top jobs. When *True*, these are top jobs.	*Boolean*			False		r	r	r, w

Scheduler Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	User	Oper	Mgr
sched_priv Directory where this scheduler keeps fairshare `usage`, `resource_group`, `holidays`, and `sched_config` files. Must be owned by root. Use default value; do not set.	*String*			*$PBS_HOME/sched_priv_<scheduler name>*				
state State of this scheduler.	*String*	down	Scheduler is not running	For default scheduler: idle For multi-sched: down		r	r	r
		idle	Scheduler is running and is waiting for a scheduling cycle to be triggered					
		scheduling	Scheduler is running and is in a scheduling cycle					
throughput_mode Allows scheduler to run faster; it doesn't have to wait for each job to be accepted, and doesn't wait for execjob_begin hooks to finish. Also allows jobs that were changed via `qalter`, server_dyn_res scripts, or peering to run in the same scheduling cycle where they were changed.	*Boolean*	*True*	Scheduler runs asynchronously and faster. Only available when PBS complex is in TPP mode.	True		r	r, w	r, w
		False	Scheduler does not run asynchronously					

6.8 Reservation Attributes

Reservation Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	User	Oper	Mgr
Account_Name No longer used.						-	-	-
Authorized_Groups List of groups who can or cannot submit jobs to this reservation. Group names are interpreted 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.* Syntax: *[+|-]<group name> [, [+|-]<group name> ...]* where '-' means "deny" and '+' means "allow".			No default. (Jobs can be submitted by all groups)	pbs.acl	r, w	r, w	r, w
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.* Syntax: *[+|-]<hostname> [, [+|-]<hostname> ...]* where '-' means "deny" and '+' means "allow". Hostnames may be wildcarded using an asterisk, according to the following rules: A hostname can contain at most one asterisk The asterisk must be the leftmost label Examples: `*.test.example.com` `*.example.com` `*.com`			No default. (Jobs can be submitted 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. List is evaluated left-to-right; first match in list is used. This list is used to set the reservation queue's acl_users attribute. See the `U` option to the `pbs_rsub` command.	*String*. Syntax: *[+|-]<username>[@<hostname>.<domain>] [, [+|-]<username>[@<hostname>.<domain>] ...]* where '-' means "deny" and '+' means "allow". Hostnames may be wildcarded using an asterisk, according to the following rules: A hostname can contain at most one asterisk The asterisk must be the leftmost label in the hostname Examples: `*.test.example.com` `*.example.com` `*.com`			Reservation owner only	pbs.acl	r, w	r, w	r, w
ctime Timestamp; time at which the reservation was created.	*Timestamp.* Printed by `qstat` in human-readable *Date* format. Output in hooks as seconds since epoch.			No default	int	r	r	r
group_list No longer used.						-	-	-
hashname No longer used.						-	-	-

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.					
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.* Syntax: *<username>@<hostname>[,<username>@<hostname>, ...]*			Reservation owner only	pbs.user_list	r, w	r, w	r, w
mtime Timestamp: the time that the reservation was last modified.	*Timestamp.* Printed by `qstat` in human-readable *Date* format. Output in hooks as seconds since epoch.				int	r	r	r
Priority No longer used.						-	-	-

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.	*String*. Format for an advance reservation: *R<unique integer>* Format for a standing reservation: *S<unique integer>*				pbs.queue	r	r	r
reserve_count The count of occurrences in a 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.	*Timestamp.* Printed by `qstat` in human-readable *Date* format. Output in hooks as seconds since epoch.				int	r, w	r, w	r, w
reserve_ID The reservation identifier.	*String*. 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
Reserve_Name The name assigned to the reservation during creation, if specified. See the `N` option to the `pbs_rsub` command.	*String*. Syntax: up to 236 characters. First character is alphabetic			No default	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_Owner The login name on the submission host of the user who created the reservation.	*String.* Syntax: *<username>@<hostname>*			Login name of creator	str	r	r	r
reserve_retry If this reservation becomes degraded, this is the next time that PBS will attempt to reconfirm this reservation.	*Timestamp.* Printed by `qstat` in human-readable *Date* format. Output in hooks as seconds since epoch.			No default	int	r	r	r
reserve_rrule The rule that describes the recurrence pattern of a standing reservation. See the `r` option to the `pbs_rsub` command.	*String*. Syntax: 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*	No default	str	r, s	r, w	r, w
		count_spec	The exact number of occurrences. Number up to 4 digits in length. Format: *integer.*	No default				
		interval_spec	Specifies interval. Format is one or both of: *BYDAY = MO|TU|WE|TH|FR|SA|SU or BYHOUR = 0|1|2|...|23*	No default				
		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*.	No default				

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.	*Timestamp.* Printed by `qstat` in human-readable *Date* format. Output in hooks as seconds since epoch.			No default	int	r, w	r, w	r, w
reserve_state The state of the reservation.	*String*	*NO RESV_NONE*	No reservation yet.	No default	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_CONFIRMED	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_RUNNING	Resv. period has started; reservation is running.		Reservation state constant: pbs.RESV_STATE_RUNNING			
		FN RESV_FINISHED	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*			No default	int	r	r	r
reserve_type No longer used.						-	-	-
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*. Syntax: *Resource_List.<resource name>=<value>[, Resource_List.<resource name>=<value>, ...]*			No default	pbs.pbs_resource Syntax: *Resource_List["<resource name>"]=<value>* where *resource name* is any built-in or custom resource	r, w	r, w	r, w
resv_nodes The list of each vnode and the resources allocated from it to satisfy the chunks requested for this reservation or occurrence.	*String*. Syntax: *(<vnode name>:<resource name>=<value>[:<resource name>=<value>]...)[+(<vnode name>:<resource name>=<value>[:<resource name>=<value>])+...]*			No default	pbs.exec_vnode	r	r	r
server Name of server.	*String*			No default	pbs.server	r	r	r
User_List No longer used.						-	-	-
Variable_List Not used						-	-	-

6.9 Queue Attributes

In the following table, Queue Type indicates the type of queue to which the attribute applies: R (routing), E (execution):

Queue Attributes									
Name Description	Format	Queue Type	Value or Option	Value or Option Description	Default Value	Python Type	User	Oper	Mgr
acl_group_enable Controls whether group access to the queue obeys the access control list defined in the acl_groups queue attribute.	*Boolean*	R, E		When set to *True*, group access to the queue is limited according to the group 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*. Syntax: *[+|-] <group name>[, ...]*	R, E			No default	pbs.acl	r	r, w	r, w
acl_host_enable Controls whether host access to the queue obeys the access control list defined in the acl_hosts queue attribute.	*Boolean*	R, E		When set to *True*, host access to the queue is limited according to the host access control list.	*False*; all hosts allowed access.	bool	r	r, w	r, w
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.* Syntax: *[+|-]<hostname>[. ...]*	R, E			No default	pbs.acl	r	r, w	r, w
acl_user_enable Controls whether user access to the queue obeys the access control list defined in the acl_users queue attribute.	*Boolean*	R, E		When set to *True*, user access to the queue is limited according to the user 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.* Syntax: *[+|-]<username> [@<hostname>][, ...]*	R, E			No default	pbs.acl	r	r, w	r, w
alt_router No longer used.							-	-	-
backfill_depth Specifies backfilling behavior for this queue. Sets the number of jobs that are to be backfilled around in this queue. Overrides backfill_depth server attribute. Recommendation: set this to less than 100.	*Integer.* Must be >=0.	E	*>=0*	PBS backfills around the specified number of jobs.	Unset. When unset, backfill depth is *1*	int	r, w	r, w	r, w
			Unset	Backfill depth is set to *1*					

Queue Attributes									
Name Description	Format	Queue Type	Value or Option	Value or Option Description	Default Value	Python Type	User	Oper	Mgr
checkpoint_min 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			No default	pbs.duration	r	r, w	r, w
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.* Syntax: *default_chunk.<resource name>=<value>[, default_chunk.<resource name>=<value>, ...]*	E			No default	pbs.pbs_resource Syntax: *default_chunk["<resource name>"]=<value>* where *resource name* is any built-in or custom resource	r	r, w	r, w
enabled Specifies whether this queue accepts new jobs.	*Boolean*	R, E	*True*	This queue is *enabled*. This queue accepts new jobs; new jobs can be enqueued.	*False*	bool	r	r, w	r, w
			False	This 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 as well as directly from submitter.					
hasnodes Indicates whether vnodes are associated with this queue. Set by PBS.	*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

Queue Attributes									
Name Description	Format	Queue Type	Value or Option	Value or Option Description	Default Value	Python Type	User	Oper	Mgr
kill_delay The time delay between sending SIGTERM and SIGKILL when a `qdel` command is issued against a running job.	*Integer*. Units: *Seconds*. Must be greater than or equal to *zero*.	E			*10 seconds*	pbs.duration	r	r, w	r, w
max_array_size The maximum number of subjobs that are allowed in an array job.	*Integer*	R, E			No default	int	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.* Syntax: *max_group_res.<resource name>=<value>* Example: `set queue workq max_group_res.ncpus=6`	E	Any PBS resource, e.g. “ncpus”, “mem”, “pmem”, etc.		No default	pbs.pbs_resource Syntax: *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 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.* Syntax: *max_group_res_soft.<resource name>=<value>* Example: `set queue workq max_group_res_soft.ncpus=3`	E	Any valid PBS resource, e.g. “ncpus”, “mem”, “pmem”, etc.		No default	pbs.pbs_resource Syntax: *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 users in a single group that are allowed to be running from this queue at one time.	*Integer*	E			No default	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			No default	int	r	r, w	r, w

Queue Attributes									
Name Description	Format	Queue Type	Value or Option	Value or Option Description	Default Value	Python Type	User	Oper	Mgr
max_queuable Old limit attribute. Incompatible with new limit attributes. The maximum number of jobs allowed to reside in this queue at any given time.	*Integer*	R, E			No default (no limit)	int	r	r, w	r, w
max_queued 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 343	R, E			No default	pbs.pbs_resource Syntax: *max_queued[“<resource name>”]=<value>* where *resource name* is any built-in or custom resource	r	r, w	r, w
max_queued_res 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 343 Syntax: *max_queued_res.<resource name>=<value>*	R, E			No default	pbs.pbs_resource Syntax: *max_queued_res[“<resource name>”]=<value>* where *resource name* is any built-in or custom resource	r	r, w	r, w
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 343	E			No default	pbs.pbs_resource Syntax: *max_run[“<resource name>”]=<value>* where *resource name* is any built-in or custom resource	r	r, w	r, w
max_run_res 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 343. Syntax: *max_run_res.<resource name>=<value>*	E			No default	pbs.pbs_resource Syntax: *max_run_res[“<resource name>”]=<value>* where *resource name* is any built-in or custom resource	r	r, w	r, w

Queue Attributes									
Name Description	Format	Queue Type	Value or Option	Value or Option Description	Default Value	Python Type	User	Oper	Mgr
max_run_res_soft 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 343. Syntax: *max_run_res_soft.<resource name>=<value>*	E			No default	pbs.pbs_resource Syntax: *max_run_res_soft[“<resource name>”]=<value>* where *resource name* is any built-in or custom resource	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.	Limit specification. See Chapter 7, "Formats", on page 343.	E			No default	pbs.pbs_resource Syntax: *max_run_soft[“<resource name>”]=<value>* where *resource name* is any built-in or custom resource	r	r, w	r, w
max_running 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			No default	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.	*String.* Syntax: *max_user_res.<resource name>=<value>* Example: `set queue workq max_user_res.ncpus=6`	E	any PBS resource, e.g. “ncpus”, “mem”, “pmem”, etc		No default	pbs.pbs_resource Syntax: *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.* Syntax: *max_user_res_soft.<resource name>=<value>* Example: `set queue workq max_user_res_soft.ncpus=3`	E	any valid PBS resource, e.g. “ncpus”, “mem”, “pmem”, etc		No default	pbs.pbs_resource Syntax: *max_user_res_soft[“<resource name>”]=<value>* where *resource name* is any built-in or custom resource	r	r, w	r, w

Queue Attributes									
Name Description	Format	Queue Type	Value or Option	Value or 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 that are allowed to be running from this queue at one time.	*Integer*	E			No default	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 any 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			No default	int	r	r, w	r, w
node_group_key Specifies the resources to use for placement sets (node grouping). Overrides server's node_group_key attribute. Specified resources must be of type string_array.	*String_array.* Syntax: Comma-separated list of resource names. When specifying multiple resources, enclose value in double quotes.	R, E			No default	pbs.node_group_key	r	r, w	r, w
partition Name of partition to which this queue is assigned. Cannot be set for routing queue. An execution queue cannot be changed to a routing queue while this attribute is set.	*String*	E			No default	str	r	r	r, w
Priority The priority of this queue compared to other queues of the same type in this PBS complex. Priority can define a queue as an express queue. See preempt_queue_prio in Chapter 4, "Scheduler Parameters", on page 243. Used for execution queues only; the value of Priority has no meaning for routing queues.	*Integer*	E	Valid values: -*1024 to 1023*		No default	int	r	r, w	r, w
queued_jobs_threshold Limit attribute. The maximum number of jobs allowed to be queued in 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 343.	R, E			No default	pbs.pbs_resource Syntax: *queued_jobs_threshold["<resource name>"]=<value>* where *resource name* is any built-in or custom resource	r	r, w	r, w

Queue Attributes									
Name Description	Format	Queue Type	Value or Option	Value or Option Description	Default Value	Python Type	User	Oper	Mgr
queued_jobs_threshold_res Limit attribute. The maximum amount of the specified resource allowed to be allocated to jobs queued in 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 343. Syntax: *"queued_jobs_threshold_res.<resource name>=<value>"*	R, E			No default	pbs.pbs_resource Syntax: *queued_jobs_threshold_res["<resource name>"]=<value>* where *resource name* is any built-in or custom resource	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	No default	PBS queue type constant: pbs.QUEUETYPE_EXECUTION	r	r, w	r, w
			r, route	Routing queue		PBS queue type constant: pbs.QUEUETYPE_ROUTE			
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						
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_assigned The total for each kind of resource allocated to running and exiting jobs in this queue.	*String.* Syntax: *resources_assigned.<resource name>=<value><newline>resources_assigned.<resource name>=<value><newline>...*	E			No default	pbs.pbs_resource Syntax: *resources_assigned["<resource name>"]=<value>* where *resource name* is any built-in or custom resource	r	r	r

Queue Attributes									
Name Description	Format	Queue Type	Value or Option	Value or Option Description	Default Value	Python Type	User	Oper	Mgr
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.	*String.* Syntax: *resources_available.<resource name>=<value><newline> resources_available.<resource name>=<value><newline>...*	E			No default	pbs.pbs_resource Syntax: *resources_available["<resource name>"]=<value>* where *resource name* is any built-in or custom resource	r	r, w	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.* Syntax: *resources_default.<resource name>=<value>, resources_default.<resource_name>=<value>, ...*	R, E			No default	pbs.pbs_resource Syntax: *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 that can be requested by a single job in this queue. This queue value supersedes any server wide maximum limit.	*String.* Syntax: *resources_max.<resource name>=<value>, resources_max.<resource name>=<value>, ...*	R, E			No default; infinite usage	pbs.pbs_resource Syntax: *resources_max["<resource name>"]=<value>* where *resource name* is any built-in or custom resource	r	r, w	r, w
resources_min The minimum amount of each resource that can be requested by a single job in this queue.	*String.* Syntax: *resources_max.<resource_name>=<value>, resources_max.<resource name>=<value>, ...*	R, E			No default; zero usage	pbs.pbs_resource Syntax: *resources_min["<resource name>"]=<value>* where *resource name* is any built-in or custom resource	r	r, w	r, w

Queue Attributes									
Name Description	Format	Queue Type	Value or Option	Value or Option Description	Default Value	Python Type	User	Oper	Mgr
route_destinations The list of destinations to which jobs may be routed. Must be set to at least one valid destination.	*String.* Syntax: comma-separated strings: *<queue name>[@<server host>[:port]]* Example: `Q1, Q2@remote, Q3@remote:15501`	R			No default	pbs.route_destinations	r	r	r, w
route_held_jobs Specifies whether jobs in the held state can be routed from this queue.	*Boolean*	R		When *True*, jobs with a hold can be routed from this queue.	*False*	bool	r	r, w	r, w
route_lifetime The maximum time a job is allowed to reside in this routing queue. If a job cannot be routed in this amount of time, the job is aborted.	*Integer.* Units: *Seconds*	R	*>0*	Jobs can reside for specified number of seconds	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.* Units: *Seconds*	R			*30 seconds*	pbs.duration	r	r, w	r, w
route_waiting_jobs Specifies whether jobs whose Execution_Time attribute value is in the future can be routed from this queue.	*Boolean*	R		When *True*, jobs with a future Execution_Time attribute can be routed from this queue.	*False*	bool	r	r, w	r, w
started Specifies whether jobs in this queue can be scheduled for execution.	*Boolean*	R, E		When *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.* Syntax: *transiting=<value>, exiting=<value>, ...*	R, E			No default	pbs.state_count	r	r	r
total_jobs The number of jobs currently residing in this queue.	*Integer*	R, E			No default	int	r	r	r

6.10 Vnode Attributes

Vnode Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
comment 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* Limit: 80 characters			No default	str	r	r	r, w
current_aoe The AOE currently instantiated on this vnode. Case-sensitive. Cannot be set on server's host.	*String*			Unset	str	r	r	r, w
current_eoe Current value of eoe on this vnode. We do not recommend setting this attribute manually.	*String*			Unset	str	r	r	r, w
in_multivnode_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 this vnode.	*String*. Syntax: *<processor number>/<job ID>, ...*				str	r	r	r
last_state_change_time Records the most recent time that this node changed state.	*Timestamp.* Printed by `qstat` in human-readable *Date* format. Output in hooks as seconds since epoch.			No default	int	r	r, w	r, w

Vnode Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
last_used_time Records the most recent time that this node finished being used for a job or reservation. Set at creation or reboot time. Updated when node is released early from a running job. Reset when node is ramped up.	*Timestamp.* Printed by `qstat` in human-readable *Date* format. Output in hooks as seconds since epoch.			Time of vnode creation or node reboot.	int	r	r, w	r, w
license Indicates whether this vnode is socket-licensed. Set by PBS.	*Character*	*l*	This vnode is socket licensed.	Unset	str	r	r	r
license_info Number of socket licenses assigned to this vnode. Set by PBS.	*Integer*			Unset	int	r	r	r
lictype No longer used.					none	-	-	-
maintenance_jobs List of jobs that were running on this vnode, but have been suspended via the *admin-suspend* signal to `qsig`. Set by server.	*String_array*			No default	str	-	-	r
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*		.	Value of vnode resource (vnode name)	str	r	r	r, w
name The name of this vnode.	*String*			No default	str	r	r	r, w
no_multinode_jobs 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	*False*	bool	r	r	r, w
ntype The type of this vnode.	*String*	*PBS*	Normal vnode	*PBS*	pbs.ND_PBS	r	r	r, w

Vnode Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
partition Name of partition to which this vnode is assigned. A vnode can be assigned to at most one partition.	*String*			No default	str	r	r, w	r, w
pbs_version The version of PBS for this MoM	*String*			No default	str	r	r	r
pcpus **Deprecated**. The number of physical CPUs on this vnode. This is set to the number of CPUs available when MoM starts. For a multiple-vnode MoM, only the natural vnode has pcpus.	*Integer*			Number of CPUs on startup	int	r	r	r
pnames The list of resources being used for placement sets. Not used for scheduling; advisory only.	*String*. Syntax: comma-separated list of resource names.			No default	str	r	r	r, w
Port Port number on which MoM daemon listens. Can be explicitly set only via `qmgr`, and only at vnode creation.	*Integer*			*15002*	int	-	r, w	r, w
poweroff_eligible Enables powering this vnode up and down by PBS.	*Boolean*	*True* *False*	PBS can power this vnode on and off. PBS cannot power this vnode on and off.	*False*	bool	r	r	r, w
power_provisioning Specifies whether this node is eligible to have its power managed by PBS, including whether it can use power profiles.	*Boolean*	*True* *False*	Power provisioning is enabled at this vnode. Power provisioning is disabled at this vnode.	*False*	bool	r	r	r, w
Priority The priority of this vnode compared with other vnodes.	*Integer*	*[-1024, +1023]* inclusive		No default	int	r	r, w	r, w
provision_enable Controls whether this vnode can be provisioned. Cannot be set on server's host.	*Boolean*	*True* *False*	This vnode may be provisioned. This vnode may not be provisioned.	*False*	bool	r	r	r, w

Vnode Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
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>* Unset	Only jobs in specified queue may run on this vnode. Any job in any queue that does not have associated vnodes can run on this vnode.	No default	pbs.queue	r	r	r, w
resources_assigned The total amount of each resource allocated to running and exiting jobs and started reservations on this vnode.	*String*. Syntax: *resources_assigned.<resource name>=<value>[,resources_assigned.<resource name>=<value>*			No default	pbs.pbs_resource Syntax: *resources_assigned['<resource name>'] = <val>* where *resource name* is any built-in or custom resource	r	r	r
resources_available 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 this vnode. If a resource value is explicitly set, that value is retained across restarts.	*String*. Syntax: *resources_available.<resource name>=<value>, resources_available.<resource name> = <value>, ...*			No default	pbs.pbs_resource Syntax: *resources_available['<resource name>'] = <val>* where *resource name* is any built-in or custom resource	r	r, w	r, w
resv List of advance and standing reservations pending on this vnode.	*String*. Comma-separated list of reservation IDs. Syntax: *<reservation ID>[, <reservation ID>, ...]*			No default	str	r	r	r

Vnode Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
resv_enable 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
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. Can be set using `pbs_mom -s insert` only. Behavior of a vnode or host is determined by a combination of the sharing attribute and a job's placement directive, defined as follows:	*String*. Example: `vnodename: sharing=force_excl`	*default_shared*	Defaults to *shared*	*default_shared*	pbs.ND_DEFAULT_SHARED	r	r, w	r, w
		default_excl	Defaults to *exclusive*		pbs.ND_DEFAULT_EXCL			
		default_exclhost	Entire host is assigned to the job unless the job's sharing request specifies otherwise		pbs.ND_DEFAULT_EXCLHOST			
		ignore_excl	Overrides any job *place=excl* setting		pbs.ND_IGNORE_EXCL			
		force_excl	Overrides any job *place=shared* setting		pbs.ND_FORCE_EXCL			
		force_exclhost	The entire host is assigned to the job, regardless of the job's sharing request		pbs.ND_FORCE_EXCLHOST			
		Unset	Defaults to *shared*					

Behavior of vnode:

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
force_exclhost	exclusive	exclusive	exclusive	exclusive	exclusive

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: *<state>[, <state>, ...]*	*busy*	Vnode is reporting load average greater than allowed max. Can combine with *offline*.	No default	int	r	r	r
		down	Node is not responding to queries from the server. Cannot be combined with *free*, *provisioning*			r	r	r
		free	Vnode is up and capable of accepting new job(s). Cannot be combined with other states.			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*, *resv_exclusive*			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
		provisioning	Vnode is being provisioned. Cannot be combined with any other states.			r	r	r
		resv-exclusive	Running reservation has requested exclusive use of vnode. Can combine with *job-exclusive*, *offline*			r	r	r
		stale	Vnode was previously reported to server, but is no longer reported to server. Cannot combine with *free*, *provisioning*			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
		unresolvable	The server cannot resolve the name of the vnode.			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.			r	r	r
topology_info Contains information intended to be used in hooks. Visible in and usable by hooks only.	*XML string*			*Unset*	str	-	-	-

Vnode Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
vnode_pool Cray only. Allows just one MoM, instead of all, to report inventory upon startup, allowing faster startup and less network communication between server and non-reporting MoMs. On each Cray, all MoMs must have same setting for this attribute. Can be set only at vnode creation; valid only on login nodes running a MoM. Not supported on non-Cray machines.	*Integer*	*0*	Unset; each MoM reports inventory separately	*0* (Unset)	int	r	r	r, w
		>0	Only one MoM per Cray reports inventory					

6.11 Job Attributes

Job Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
Account_Name String used for accounting purposes. Can be used for fairshare.	*String*. Can contain any character.			No default	str	r, w	r, w	r, w
accounting_id Accounting ID for tracking accounting data not produced by PBS.	*String*			No default	str	r	r	r
accrue_type Indicates what kind of time the job is accruing.	*Integer*	*0 (initial_time)*	Job is accruing initial time. Can occur when job is blocked by a runjob hook.	2 (eligible_time)	int	-	-	r
		1 (ineligible_time)	Job is accruing ineligible time. Occurs when job or owner has hit limit.					
		2 (eligible_time)	Job is accruing eligible time. Occurs when job is blocked on resources.					
		3 (run_time)	Job is accruing run time. Occurs when job is running.					
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 is also 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; this is *cpuset=<path to cpuset>*. On Windows, holds PBS home directory.	*String.* May contain white spaces.			No default	str	r	r	r

Job Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
argument_list Job executable's argument list. Shown if job is submitted with "`-- <executable> [<argument list>]`"	*JSDL-encoded string.* *<jsdl-hpcpa:Argument> <1st arg> </jsdl-hpcpa:Argument> <jsdl-hpcpa:Argument> <2nd arg> </jsdl-hpcpa:Argument> <jsdl-hpcpa:Argument> <nth arg> </jsdl-hpcpa:Argument>* Example: if arguments are "A B": `<jsdl-hpcpa:Argument>A</jsdl-hpcpa:Argument> <jsdl-hpcpa:Argument>B</jsdl-hpcpa:Argument>`			No default	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_id Applies only to subjobs. Array identifier of subjob.	*String*			No default	str	r	r	r
array_index Applies only to subjobs. Index number of subjob.	*String*			No default	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`.			No default	str	r	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`			No default	pbs.range	r, s	r	r
array_state_count Applies only to job arrays. Lists number of subjobs in each state.	*String*			No default	pbs.state_count	r	r	r

Job Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
block Specifies whether `qsub` will wait for the job to complete and return the exit value of the job. For X11 forwarding jobs, and jobs with interactive and/or block attributes set to *True*, the job's exit status is not returned.	*Boolean*			*False*	int	r, s	r	r
Checkpoint Determines when the job will be checkpointed. 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.checkpoint	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 = <*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 job's execution queue, 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*			No default	str	r	r, w	r, w

Job Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
ctime Timestamp; time at which the job was created.	*Timestamp.* Printed by `qstat` in human-readable *Date* format. Output in hooks as seconds since epoch.			No default	int	r	r	r
depend Specifies inter-job dependencies. No limit on number of dependencies.	*String*. Syntax: *<type>:<job ID>[,<job ID> ...],[<type>:<job ID>[,<job ID> ...] ...]* Must be enclosed in double quotes if it contains commas. Example: "`before:123,456`"	after:<job ID list>	This job may run at any point after all jobs in *job ID list* have started execution.	No default; no dependencies	pbs.depend	r, w	r, w	r, w
		afterok:<job ID list>	This job may run only after all jobs in *job ID list* have terminated with no errors.					
		afternotok:<job ID list>	This job may run only after all jobs in *job ID list* have terminated with errors.					
		afterany:<job ID list>	This job can run after all jobs in *job ID list* have finished execution, with or without errors. This job will not run if a job in the *job ID list* was deleted without ever having been run.					
		before:<job ID list>	Jobs in *job ID list* may start once this job has started.					
		beforeok:<job ID list>	Jobs in *job ID list* may start once this job terminates without errors.					
		beforenotok:<job ID list>	If this job terminates execution with errors, jobs in *job ID list* may begin.					
		beforeany:<job ID list>	Jobs in *job ID list* may begin execution once this job terminates execution, with or without errors.					
		on:<count>	This job may run after *count* dependencies on other jobs have been satisfied. 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
egroup If the job is queued, this attribute is set to the group name under which the job is to be run.	*String*			No default	str	-	-	r
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. See the qsub and qalter commands.	*String*. Syntax: *[<hostname>:]<path>*	*<relative path>*	Path is relative to the current working directory of command executing on current host.	Default path is current working directory where `qsub` is run. If the output path is specified, but does not include a filename, the default filename is *<job ID>.ER*. If the path name is not specified, the default filename is *<job name>.e<sequence number>*.	str	r, w	r, w	r, w
		<absolute path>	Path is absolute path on current host where command is executing.					
		<hostname>:<relative path>	Path is relative to user's home directory on specified host.					
		<hostname>:<absolute path>	Path is absolute path on named host.					
		No path	Path is current working directory where `qsub` is executed.					

Job Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
estimated List of estimated values for job. Used to report job's exec_vnode, start_time, and soft_walltime. Can be set in a hook or via `qalter`, but PBS will overwrite the values.	Syntax: *estimated.<resource name>=<value>, estimated.<resource name>=<value>.* *exec_vnode* is a *string*. *soft_walltime* is a *duration*. *start_time* is printed by `qstat` in human-readable *Date* format; *start_time* is output in hooks as seconds since epoch.	*exec_vnode*	The estimated vnodes used by this job.	Unset	pbs.pbs_resource Syntax: *estimated.[<resource name>]=<value>*. *exec_vnode* is a *pbs.exec_vnode*. *soft_walltime* is a *duration*. *start_time* is an *int*.	r	r, w	r, w
		soft_walltime	The estimated soft walltime for this job. Calculated when a job exceeds its soft_walltime resource.	Unset		r	r	r, w
		start_time	The estimated start time for this job.	Unset		r	r, w	r, w
etime Timestamp; time 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, or is held then released. Not affected by qaltering.	*Timestamp.* Printed by `qstat` in human-readable *Date* format. Output in hooks as seconds since epoch.			No default	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*			No default	str	-	-	r
executable JSDL-encoded listing of job's executable. Shown if job is submitted with "`-- <executable> [<arg list>]`".	*JSDL-encoded string.* *<jsdl-hpcpa:Executable> <name of executable>* Example: if the executable is `ping`: `<jsdl-hpcpa:Executable>ping</jsdl-hpcpa:Executable>`			No default	str	r, w	r, w	r, w

Job Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
Execution_Time Timestamp; 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, "Formats", on page 343.			Unset; no delay	int	r, w	r, w	r, w
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*. Syntax: <*host-name*>/*N*[**C*][+...], where *N* is task slot number starting at 0, on that host, and *C* is the number of CPUs allocated to the job. **C* does not appear if its value is *1*.			No default	pbs.exec_host	r	r, i	r, i
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. Example: 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)`. For a job which requested one chunk and exclusive use of a 2-vnode host, where the chunk was satisfied by resources from one vnode, exec_vnode is `(vnodeA:ncpus=N:mem=X)+(vnodeB)`.			No default	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
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*			No default	int	r	r	r
forward_x11_cookie Contains the X authorization cookie.	*String*			No default	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*			No default	int	r	r	r
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*. Syntax: *<group name>[@<host-name>] [,<group name>[@<host-name>]...]* Must be enclosed in double quotes if it contains commas.			No default	pbs.group_list	r, w	r, w	r, w
hashname No longer used.						-	-	-
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 *held* state. The *held* state takes precedence over the *wait* state.	*String*, made up of the letters ‘n’, ‘o’, ‘p’, ‘s’, ‘u’	*n* *o* *p* *s* *u*	No hold Other hold Bad password System hold User hold	*n*	pbs.hold_types	r, w	r, w	r, w

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. When both this attribute and the block attribute are *True*, no exit status is returned. For X11 forwarding jobs, the job's exit status is not returned. Cannot be set using a PBS directive. Job arrays cannot be interactive.	*Boolean*		Set to *True* if this is an interactive job.	*False*	int	r, w	r	r
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*			No default	str	r	r	r
Job_Name The job name. See the `qalter` and `qsub` commands.	*String* up to 236 characters, first character must be alphabetic or numeric			Base name of job script, or STDIN	str	r, w	r, w	r, w
Job_Owner The login name on the submitting host of the user who submitted the batch job.	*String.* Syntax: *<Username>@<submission host>*			No default	str	r	r	r

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*	*B (Begun)*	Job arrays only. Job array has begun execution.	No default	pbs.JOB_STATE_BEGUN	r, i	r, i	r, i
		E (Exiting)	The job has finished, with or without errors, and PBS is cleaning up post-execution.		pbs.JOB_STATE_EXITING			
		F (Finished)	Job is finished. Job has completed execution, job failed during execution, or job was deleted.		pbs.JOB_STATE_FINISHED			
		H (Held)	The job is held.		pbs.JOB_STATE_HELD			
		M (Moved)	The job has been moved to another server.		pbs.JOB_STATE_MOVED			
		Q (Queued)	The job resides in an execution or routing queue pending execution or routing. It is not in held or waiting state.		pbs.JOB_STATE_QUEUED			
		R (Running)	The job is in an execution queue and is running.		pbs.JOB_STATE_RUNNING			
		S (Suspended)	The job was executing and has been suspended. The job does not use CPU cycles or walltime.		pbs.JOB_STATE_SUSPEND			
		T (Transiting)	The job is being routed or moved to a new destination.		pbs.JOB_STATE_TRANSIT			
		U (User suspended)	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.		pbs.JOB_STATE_SUSPEND_USERACTIVE			
		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.		pbs.JOB_STATE_WAITING			
		X (Expired)	Subjobs only. Subjob is finished (expired.)		pbs.JOB_STATE_EXPIRED			

Job Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
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.	*String* One of "*oe*", "*eo*", or "*n*".	*eo*	Standard output and standard error are merged, intermixed, into a single stream, which becomes standard error.	*n*	pbs.join_path	r, w	r, w	r, w
		oe	Standard output and standard error are merged, intermixed, into a single stream, which becomes standard output.					
		n	Standard output and standard error are not merged.					
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* One of "*o*", "*e*", "*oe*", "*eo*", or "*n*".	*o*	The standard output stream is retained. The filename is: `job_name.o<sequence number>`	*n*	pbs.keep_files	r, w	r, w	r, w
		e	The standard error stream is retained. The filename is: `job_name.e<sequence number>`					
		eo, oe	Both standard output and standard error streams are retained.					
		d	Output and error are written directly to their final destination					
		n	Neither stream is retained. Files are 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*", with optional "*j*", or "*n*".	*a*	Mail is sent when job is aborted	*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					
		j	Mail is sent for subjobs. Must be combined with one or more of a, b, and e options					
		n	No mail is sent. Cannot be combined 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* Syntax: *"<username>@<hostname>[,<username>@<hostname>]"* Must be enclosed in double quotes if it contains commas.			Job owner only	pbs.email_list	r, w	r, w	r, w

Job Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
mtime Timestamp; the time that the job was last modified, changed state, or changed locations.	*Timestamp.* Printed by `qstat` in human-readable *Date* format. Output in hooks as seconds since epoch.			No default	int	r	r	r
no_stdio_sockets Not used.						-	-	-
Output_Path The final path name for the file containing the job's standard output stream. See the `qsub` and `qalter` commands.	*String*. Syntax: *[<hostname>:]<path>*	*<relative path>*	Path is relative to the current working directory of command executing on current host.	Default path is current working directory where `qsub` is run. If the output path is specified, but does not include a filename, the default filename is *<job ID>.OU*. If the path name is not specified, the default filename is *<job name>.o<sequence number>*.	str	r, w	r, w	r, w
		<absolute path>	Path is absolute path on current host where command is executing.					
		<hostname>:<relative path>	Path is relative to user's home directory on specified host.					
		<hostname>:<absolute path>	Path is absolute path on named host.					
		No path	Path is current working directory where `qsub` is executed.					
pcap_accelerator Power attribute. Power cap for an accelerator. Corresponds to Cray `capmc set_power_cap --accel` setting. See `capmc` documentation.	*Integer* Units*: Watts*			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
pcap_node Power attribute. Power cap for a node. Corresponds to Cray `capmc set_power_cap --node` setting. See `capmc` documentation.	*Integer* Units*: Watts*			Unset	int	r, w	r, w	r, w
pgov Power attribute. Cray ALPS reservation setting for CPU throttling corresponding to p-governor. See BASIL 1.4 documentation. We do not recommend using this attribute.	*String*			Unset	str	r, w	r, w	r, w
Priority The scheduling priority for the job. Higher value indicates greater priority.	*Integer*. Syntax: *[+|-]nnnn*	*[-1024, +1023]* inclusive		Unset	int	r, w	r, w	r, w
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 (""""), semicolon (";"), colon (":"), vertical bar ("|"), left angle bracket ("<"), right angle bracket (">"), plus ("+"), comma (","), question mark ("?"), and asterisk ("*").			*_pbs_project_default*	str	r, w	r, w	r, w
pset Name of placement set used by the job.	*String*				str	r	r	r, w
pstate Power attribute. Cray ALPS reservation setting for CPU frequency corresponding to p-state. See BASIL 1.4 documentation.	*String* Units: *Hertz*			Unset	str	r, w	r, w	r, w

Job Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
qtime Timestamp; the time that the job entered the current queue.	*Timestamp.* Printed by `qstat` in human-readable *Date* format. Output in hooks as seconds since epoch.			No default	int	r	r	r
queue The name of the queue in which the job currently resides.	*String*			No default	pbs.queue	r	r	r
queue_rank A number indicating the job's position within its queue. Only used internally by PBS.	*Integer*			No default	int	-	-	r
queue_type The type of queue in which the job is currently residing.	*Character*	*E*	Execution queue	No default	pbs.QTYPE_EXECUTION	-	-	r
		R	Routing queue		pbs.QTYPE_ROUTE			
release_nodes_on_stageout Controls whether job vnodes are released when stageout begins. Cannot be used with vnodes managed by cpuset MoMs, (whose arch is linux_cpuset), or with vnodes tied to Cray X* series systems. When cgroups is enabled and this is used with some but not all vnodes from one MoM, resources on those vnodes that are part of a cgroup are not released until the entire cgroup is released. The job's stageout attribute must be set for the release_nodes_on_stageout attribute to take effect.	*Boolean*	*True*	All of the job's vnodes not on the primary execution host are released when stageout begins	*False*	bool	r, w	r, w	r, w
		False	Job's vnodes are released when the job finishes and MoM cleans up the job					

Job Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
Remove_Files Specifies whether standard output and/or standard error files are automatically removed upon job completion.	*String*	*e*	Standard error is removed upon job completion	Unset	str	r, w	r, w	r, w
		o	Standard output is removed upon job completion					
		eo	Standard output and standard error are removed upon job completion					
		oe	Standard output and standard error are removed upon job completion					
		unset	Neither is removed					
Rerunable Specifies whether the job can be rerun. Does not affect how a job is treated if the job could not begin execution. See "Allowing Your Job to be Re-run", on page 120 of the PBS Professional User's Guide. Job arrays are required to be rerunnable and are rerunnable by default.	*Character*	*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 *<resource 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 determined by a queue or server default established by the administrator. See Chapter 5, "List of Built-in Resources", on page 255.	*String*. Syntax: *Resource_List.<resource name>=<value>], Resource_List.<resource name>=<value>, ...]*			No default	pbs.pbs_resource Syntax: *Resource_List["<resource name>"]=<value>* where *resource name* is any built-in or custom resource	r, w	r, w	r, w
resources_released Listed by vnode, consumable resources that were released when the job was suspended. Populated only when restrict_res_to_release_on_suspend server attribute is set. Set by server.	*String*. Syntax: *(<vnode>:<resource name>=<value>:<resource name>=<value>:...)+(<vnode>:<resource name>=<value>:...)*			No default	str	r	r	r

Job Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
resource_released_list Sum of each consumable resource requested by the job that was released when the job was suspended. Populated only when restrict_res_to_release_on_suspend server attribute is set. Set by server.	*String*. Syntax: *resource_released_list.<resource name>=<value>,resource_released_list.<resource name>=<value>, ...*			No default	pbs.pbs_resource	--	r	r
resources_used The amount of each resource used by the job.	*String.* Syntax: List of *resources_used.<resource name>=<value>,resources_used.<resource name>=<value>* pairs. Example: `resources_used.mem=2mb`			No default	pbs.pbs_resource Syntax: *resources_used["<resource name>"]=<value>* where *resource name* is any built-in or custom resource	r	r	r
run_count The number of times the server thinks the job has been executed. The run_count attribute starts at zero. Job is held after 21 tries. Can be set via `qsub`, `qalter`, or a hook.	*Integer.* Must be greater than or equal to *zero*.			*Zero*	int	-	r, w	r, w
run_version Used internally by PBS to track the instance of the job.	*Integer*				int	--	--	r
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.					
schedselect The union of the select specification of the job, and the queue and server defaults for resources in a chunk.	*String*			No default	pbs.select	-	-	r

Job Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
sched_hint No longer used.						-	-	-
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*			No default	pbs.server	r	r	r
session_id If the job is running, this is set to the session ID of the first executing task.				No default	int	r	r	r
Shell_Path_List One or more absolute paths to the program(s) to process the job's script file.	*String*. Syntax: *"<path>[@<hostname>][,<path>[@<hostname>]...]"* 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*. Syntax: *"<execution path>@<storage host>:<storage path>[, <execution path>@<storage host>:<storage path>, ...]"* Must be enclosed in double quotes if it contains commas.			No default	pbs.staging_list	r, w	r, w	r, w

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*. Syntax: *"<execution path>@<storage host>:<storage path>[, <execution path>@<storage host>:<storage path>, ...]"* Must be enclosed in double quotes if it contains commas.			No default	pbs.staging_list	r, w	r, w	r, w
Stageout_status 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. Available only for finished jobs.	*Integer*			No default	int	r	r	r
stime Timestamp; time when the job started execution. Changes when job is restarted.	*Timestamp.* Printed by `qstat` in human-readable *Date* format. Output in hooks as seconds since epoch.			No default	int	r	r	r
Submit_arguments Job submission arguments given on the `qsub` command line. Available for all jobs.	*String*			No default	str	r, w	r, w	r, w
substate The substate of the job. The substate is used internally by PBS.	*Integer*			No default	int	r	r	r
sw_index No longer used.						-	-	-

Job Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Def Val	Python Type	Usr	Opr	Mgr
tolerate_node_failures Specifies whether job can have extra vnodes allocated, and whether for startup only or for the life of the job.	*String*	*none, unset*	No extra vnodes are allocated to the job.	None	str	r, s	r, s	r, s
		job_start	Extra vnodes are allocated only long enough to start the job successfully. Tolerate vnode failures that occur only during job start, just before executing the job's top level shell or executable or any execjob_launch hooks. Failures tolerated are those such as an assigned sister MoM failing to join the job and communication errors between MoMs.					
		all	Extra vnodes are allocated for the life of the job. Tolerate all node failures resulting from communication problems, such as polling problems, between the primary MoM and the sister MoMs assigned to the job Tolerate failures due to rejections from execjob_begin or execjob_prologue hooks run at sister MoMs.					
topjob_ineligible Allows administrators to mark this job as ineligible to be a top job.	*Boolean*	*True*	This job is not eligible to be a top job.	*Unset*, behaves like *False*	bool	-	-	r, w
		False	This job is eligible to be a top job.					
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. No length limit. 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 value of Job_Owner as the user name.	*String*. Syntax: *"<username>@<hostname> [,<username>@<hostname>...]"* Must be enclosed in double quotes if it contains commas. May be up to 256 characters in length.			Value of Job_Owner job attribute	pbs.user_list	r, w	r, w	r, w
Variable_List List of environment variables set in the job's execution environment. See the `qsub(1B)` command.	*String*. Syntax: *"<variable name>=<value> [,<variable name>=<value>...]"* Must be enclosed in double quotes if it contains commas.			No default	pbs.pbs_resource Syntax: *Variable_List["<variable name>"]=<value>*	r, w	r, w	r, w

6.12 Hook Attributes

An unset hook attribute takes the default value for that attribute.

Hook attributes can be set by root or the Admin at the local server only.

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*				
debug Specifies whether or not the hook produces debugging files under `PBS_HOME/server_priv/hooks/tmp` or `PBS_HOME/mom_priv/hooks/tmp`. Files are named *hook_<hook event>_<hook name>_<unique ID>.in*, *.data*, and *.out*. See "Producing Files for Debugging" on page 149 in the PBS Professional Hooks Guide.	*Boolean*	*True*	The hook leaves debugging files when it runs.	*False*				
		False	The hook does not leave debugging files when it runs.					
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 does not run when triggering event occurs.					

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	str			
		modifyjob	Triggered when job is modified					
		movejob	Triggered when job is moved					
		resv_end	Triggered when reservation ends					
		resvsub	Triggered when reservation is created					
		runjob	Triggered when job is run					
		periodic	Triggered periodically at server					
		provision	Hook is master provisioning hook					
		execjob_begin	Triggered when MoM receives job					
		execjob_prologue	Triggered just before first job process					
		execjob_launch	Triggered just before executing user's program					
		execjob_attach	Triggered before running any execjob_prologue hooks, on each vnode where `pbs_attach()` runs					
		execjob_end	Triggered after job finishes or is killed					
		execjob_preterm	Triggered just before job is killed					
		execjob_epilogue	Triggered just after job runs successfully					
		exechost_periodic	Triggered at periodic interval on execution hosts					
		exechost_startup	Triggered when MoM starts up or receives SIGHUP (Linux)					
		""	Hook is not triggered					

Hook Attributes								
Name Description	Format	Val / Opt	Value/Option Description	Default Value	Python Type	Usr	Opr	Mgr
fail_action Specifies the action to be taken when hook fails due to alarm call or unhandled exception, or to an internal error such as not enough disk space or memory. Can also specify a subsequent action to be taken when hook runs successfully. Value can be either "*none*" or one or more of "*offline_vnodes*", "*clear_vnodes_upon_recovery*", and "*scheduler_restart_cycle*". If this attribute is set to multiple values, scheduler restart happens last. See "Offlining and Clearing Vnodes Using the fail_action Hook Attribute" on page 62 in the PBS Professional Hooks Guide and "Restarting Scheduler Cycle After Hook Failure" on page 59 in the PBS Professional Hooks Guide.	*String_array*	*none*	No action is taken.	*none*				
		offline_vnodes	After unsuccessful hook execution, offlines the vnodes managed by the MoM executing the hook. Only available for **execjob_prologue**, **exechost_startup** and **execjob_begin** hooks.					
		clear_vnodes_upon_recovery	After successful hook execution, clears vnodes previously offlined via *offline_vnodes* fail action. Only available for **exechost_startup** hooks.					
		scheduler_restart_cycle	After unsuccessful hook execution, restarts scheduling cycle. Only available for **execjob_begin** and **execjob_prologue** hooks.					
freq Number of seconds between **periodic** or **exechost_periodic** triggers.	*Integer*		Number of seconds between triggers	*120*				
order Indicates relative order of hook execution, for hooks of the same type sharing a trigger. Hooks with lower **order** values execute before those with higher values. Does not apply to **periodic** or **exechost_periodic** hooks.	*Integer*	*Range:* *built-in hooks: [-1000, 2000]* *site hooks: [1,1000]*		*1*				
type The type of the hook. Cannot be set for a built-in hook.	*String*	*pbs*	Hook is built in	*site*				
		site	Hook is custom (site-defined)					
user Specifies who executes the hook.	*String*	*pbsadmin*	Hook runs as root	*pbsadmin*				
		pbsuser	Hook runs as owner of job					

7 Formats

This chapter describes the formats used in PBS Professional.

7.1 List of Formats

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. Cannot be used for a vnode name.

Boolean

Name of Boolean 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

A datetime is

[[[[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.

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

An integer whose units are seconds

or

[[hours:]minutes:]seconds[.milliseconds]

in the form:

[[HH:]MM:]SS[.milliseconds]

Milliseconds are rounded to the 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 hostname cannot contain a colon, ":".

The maximum length of a hostname supported by PBS is defined by PBS_MAXHOSTNAME, and is *255*.

Job Array ID, Job Array Identifier

Job array identifiers are a sequence number followed by square brackets:

<sequence number>[][.<server name>][@<server name>]

Example:

```
1234[]
```

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

The largest value that *sequence number* can be is set in the max_job_sequence_id server attribute. This attribute defaults to *9999999*. Minimum value for this attribute is *9999999*, and maximum is *999999999999.* After maximum for sequence number has been reached, job array IDs start again at *0*.

Job Array Range

<sequence number>[<first>-<last>][.<server name>][@<server name>]

first and *last* are the first and last indices of the subjobs.

Job ID, Job Identifier

<sequence number>[.<server name>][@<server name>]

The largest value that *sequence number* can be is set in the max_job_sequence_id server attribute. This attribute defaults to *9999999*. Minimum value for this attribute is *9999999*, and maximum is *999999999999.* After maximum for sequence number has been reached, job IDs start again at *0*.

Job Name, Job Array Name

A job name or job array name can be at most 230 characters. It must consist only of alphabetic, numeric, plus sign (“+”), dash or minus or hyphen (“-”), underscore (“_”), and dot or period (“.”) 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*”.

Limit Specification

<limit specification>=<limit value>[, <limit specification>=<limit value>, ...]

where *limit specification* is:

Table 7-2: Limit Specification Syntax

Limit Specification	Limit
o:PBS_ALL	Overall limit
u:PBS_GENERIC	Generic users
u:<username>	An individual user
g:PBS_GENERIC	Generic groups
g:<group name>	An individual group
p:PBS_GENERIC	Generic projects
p:<project name>	An individual project

- The *limit specification* can contain spaces anywhere except after the colon (":").
- If there are comma-separated *limit specifications*, the entire string must be enclosed in double quotes.
- A username, group name, or project name containing spaces must be enclosed in quotes.
- If a username, group name, or project name is quoted using double quotes, and the entire string requires quotes, the outer enclosing quotes must be single quotes. Similarly, if the inner quotes are single quotes, the outer quotes must be double quotes.
- *PBS_ALL* is a keyword which indicates that this limit applies to the usage total.
- *PBS_GENERIC* is a keyword which indicates that this limit applies to generic users, groups, or projects.
- When removing a limit, the *limit value* does not need to be specified.
- *PBS_ALL* and *PBS_GENERIC* are case-sensitive.

Format for setting a limit attribute:

set server <limit attribute> = "<limit specification>=<limit value>[, <limit specification>=<limit value>], ..."

set queue <queue name> <limit attribute> = "<limit specification>=<limit value>[, <limit specification>=<limit value>], ..."

For example, to set the max_queued limit on QueueA to *5* for total usage, and to limit user bill to *3*:

```
Qmgr: s q QueueA max_queued = "[o:PBS_ALL=5], [u:bill =3]"
```

Examples of setting, adding, and removing:

```
Qmgr: set server max_run="[u:PBS_GENERIC=2], [g:group1=10], [o:PBS_ALL = 100]"
Qmgr: set server max_run+="[u:user1=3], [g:PBS_GENERIC=8]"
Qmgr: set server max_run-="[u:user2], [g:group3]"
Qmgr: set server max_run_res.ncpus="[u:PBS_GENERIC=2], [g:group1=8], [o:PBS_ALL = 64]"
```

See "How to Set Limits at Server and Queues" on page 293 in the PBS Professional Administrator's Guide.

Event logfile-date-time

Date and time stamp in the format:

mm/dd/yyyy hh:mm:ss[.xxxxxx]

If microsecond logging is enabled, microseconds are logged using the *.xxxxxx* portion. Microseconds may be preceded by zeroes. Microsecond logging is controlled per host via the PBS_LOG_HIGHRES_TIMESTAMP configuration parameter or environment variable.

Long

Long integer. Allowable values: 0-9 [[0-9] ...], and + and -

pathname

All printable characters except for colon (":"), quotes(""""), and ampersand ("&")

PBS NAME

This is a generic term, used to describe various PBS entities. For example, attribute names are PBS NAMEs.

Must start with an alphabetic character, and may contain only the following: alpha-numeric, underscore ("_"), or dash ("-").

PBS Password

The `pbs_ds_password` command generates passwords containing the following characters:

0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!@#$%^&()_+*

When creating a password manually, do not use \ (backslash) or ‘ (backquote). This can prevent certain commands such as `pbs_server`, `pbs_ds_password`, and `printjob` from functioning properly, as they rely on connecting to the database.

Project Name

A 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*".

Queue ID, 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

Reservation ID, Reservation Identifier

R<sequence number>[.<server name>][@<server name>]

The largest value that *sequence number* can be is set in the max_job_sequence_id server attribute. This attribute defaults to *9999999*. Minimum value for this attribute is *9999999*, and maximum is *999999999999*. After maximum for sequence number has been reached, reservation IDs start again at *0*.

Reservation Name

Same as Job Name. See "Job Name, Job Array Name" on page 345.

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 344.

Size

Number of bytes or words. The size of a word is 64 bits.

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 megawords	2 to the 20th, or 1,048,576
gb or gw	Gigabytes or gigawords	2 to the 30th, or 1,073,741,824
tb or tw	Terabytes or terawords	2 to the 40th, or 1024 gigabytes
pb or pw	Petabytes or petawords	2 to the 50th, or 1,048,576 gigabytes

Default: *bytes*

Note that a scheduler rounds all resources of type size up to the nearest kb.

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. No limit on length. Python type is *str*.

Subjob Identifier

Subjob identifiers are a sequence number followed by square brackets enclosing the subjob's index:

<sequence number>[<index>][.<server name>][@<server name>]

Example:

```
1234[99]
```

Timestamp

Output format varies depending on context:

- Printed by `qstat` in human-readable *Date* format
- Output in hooks as seconds since epoch

Username

String up to 16 characters in length. PBS supports usernames containing any printable, non-whitespace character except the at sign (“@”). Your platform may place additional limitations on usernames.

Username, 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, 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, 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 can be alphanumeric and any of these:

  - (dash)

  _ (underscore)

  @ (at sign)

  [(left bracket)

  ] (right bracket)

  # (hash)

  ^ (caret)

  / (slash)

  \ (backslash)

  . (period)
- Cannot be the same as an attribute name

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	Numeric	Description
B	*7*	Job arrays only: job array is begun, meaning that at least one subjob has started
E	*5*	Job is exiting after having run
F	*9*	Job is finished. Job has completed execution, job failed during execution, or job was deleted.
H	*2*	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	*8*	Job was moved to another server
Q	*1*	Job is queued, eligible to run or be routed
R	*4*	Job is running
S	*None; sub-state of Running*	Job is suspended by scheduler. A job is put into the suspended state when a higher priority job needs the resources.
T	*0*	Job is in transition (being moved to a new location)
U	*None; sub-state of Running*	Job is suspended due to workstation becoming busy
W	*3*	Job is waiting for its requested execution time to be reached or job specified a stagein request which failed for some reason.
X	*6*	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
13	Job waiting on sync start ready
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
21	job held waiting on sync regist
22	job held - waiting on dependency
30	job waiting until user-specified execution time
37	job held - file stage in failed
41	job sent to MoM to run
42	Running
43	Suspended by Operator or Manager
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

Table 8-2: Job Substates

Substate Number	Substate Description
62	Job to be rerun, deleting files
63	Job to be rerun, freeing resources
69	subjob is gone
70	Array job has begun
71	Job is waiting for vnode(s) to be provisioned with its requested AOE.
91	Job is terminated
92	Job is finished
93	Job failed
94	Job was moved
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	Numeric	Indication
B	*7*	The job array has started
E	*5*	All subjobs are finished and the server is cleaning up the job array
F	*9*	The job array is finished
H	*2*	The job array is held
Q	*1*	The job array is queued, or has been qrerun
T	*0*	The job array is in transit between servers
W	*3*	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	Numeric	Indication
E	*5*	Ending
F	*9*	Finished
Q	*1*	Queued
R	*4*	Running
S	*None; sub-state of Running*	Suspended
U	*None; sub-state of Running*	Suspended by keyboard activity
X	*6*	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 shown in the following table:

Table 8-5: Server States

State	Description
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
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*. A 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* *maintenance*
down	Server	Node is not usable. Existing communication lost between server and MoM.	*maintenance* 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*
maintenance	Server	A vnode enters the *maintenance* state when any of its jobs is suspended with the *admin-suspend* signal. Other jobs running on this vnode continue to run; each job must be *admin-suspend*ed. The vnode leaves the *maintenance* state when the last job is resumed with the *admin-resume* signal. A scheduler does not start or resume jobs on a node in the *maintenance* state. Any reservations on vnodes in the *maintenance* state are marked *degraded*. PBS searches for alternate vnodes for those reservations.	*down* *offline*
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*
powered-off		Indicates that this vnode was powered off by PBS via power provisioning. This tells the scheduler that it can schedule jobs on this vnode; in that case PBS powers the vnode back up.	

Table 8-6: Vnode States

State	Set By	Description	Can Combine With these States
powering-down		Indicates that this vnode is in the process of being powered down by PBS via power provisioning.	
powering-on		Indicates that this vnode is in the process of being powered up by PBS via power provisioning.	
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*
sleep	Server	Indicates that this vnode was ramped down or powered off via PBS power management. This tells the scheduler that it can schedule jobs on this vnode; in that case PBS powers the vnode back up.	
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	
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
AL	11	*RESV_BEING_ALTERED*	Transitory state; reservation is being altered
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
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

Occurrence Type	State or Substate	Reservation Time in Future	Reservation Time Is Now
Advance Reservation	State	*RESV_DEGRADED*	*RESV_RUNNING*
	Substate	*RESV_DEGRADED*	*RESV_DEGRADED*
Soonest Occurrence	State	*RESV_DEGRADED*	*RESV_RUNNING*
	Substate	*RESV_DEGRADED*	*RESV_DEGRADED*
Non-soonest Occurrence Only	State	*RESV_CONFIRMED*	N/A
	Substate	*RESV_DEGRADED*	N/A

9 The PBS Configuration File

9.1 Contents of Configuration File

The /etc/pbs.conf file contains configuration parameters for PBS. The following table describes the parameters you can use in the pbs.conf configuration file:

Table 9-1: Parameters in pbs.conf

Parameter	Description
PBS_AUTH_METHOD	Authentication method to be used by PBS. Only allowed value is "munge" (case-insensitive).
PBS_BATCH_SERVICE_PORT	Port on which server listens. Default: 15001
PBS_BATCH_SERVICE_PORT_DIS	DIS port on which server listens.
PBS_COMM_LOG_EVENTS	Communication daemon log mask. Default: *511*
PBS_COMM_ROUTERS	Tells a pbs_comm the location of the other pbs_comms.
PBS_COMM_THREADS	Number of threads for communication daemon.
PBS_CONF_REMOTE_VIEWER	Specifies remote viewer client. If not specified, PBS uses native Remote Desktop client for remote viewer. Set on submission host(s). Supported on Windows only.
PBS_CORE_LIMIT	Limit on corefile size for PBS daemons. Can be set to an integer number of bytes or to the string "unlimited". If unset, core file size limit is inherited from the shell environment.
PBS_DATA_SERVICE_PORT	Used to specify non-default port for connecting to data service. Default: 15007
PBS_ENVIRONMENT	Location of pbs_environment file.
PBS_EXEC	Location of PBS bin and sbin directories.
PBS_HOME	Location of PBS working directories.
PBS_LEAF_NAME	Tells endpoint what hostname to use for network. The value does not include a port, since that is usually set by the daemon. By default, the name of the endpoint's host is the hostname of the machine. You can set the name where an endpoint runs. This is useful when you have multiple networks configured, and you want PBS to use a particular network. TPP internally resolves the name to a set of IP addresses, so you do not affect how pbs_comm works.

Table 9-1: Parameters in `pbs.conf`

Parameter	Description
PBS_LEAF_ROUTERS	Location of endpoint's `pbs_comm` daemon(s).
PBS_LOCALLOG=<value>	Enables logging to local PBS log files. Valid values: 0: no local logging 1: local logging enabled Only available when using `syslog`.
PBS_MAIL_HOST_NAME	Used in addressing mail regarding jobs and reservations that is sent to users specified in a job or reservation's Mail_Users attribute. Optional. If specified, must be a fully qualified domain name. Cannot contain a colon (":"). For how this is used in email address, see section 2.2.2, "Specifying Mail Delivery Domain", on page 18.
PBS_MANAGER_SERVICE_PORT	Port on which MoM listens. Default: 15003
PBS_MOM_HOME	Location of MoM working directories.
PBS_MOM_NODE_NAME	Name that MoM should use for natural vnode, and if they exist, local vnodes. If this is not set, MoM defaults to using the non-canonicalized hostname returned by `gethostname()`.
PBS_MOM_SERVICE_PORT	Port on which MoM listens. Default: 15002
PBS_OUTPUT_HOST_NAME	Host to which all job standard output and standard error are delivered. If specified in `pbs.conf` on a job submission host, the value of PBS_OUTPUT_HOST_NAME is used in the host portion of the job's Output_Path and Error_Path attributes. If the job submitter does not specify paths for standard output and standard error, the current working directory for the `qsub` command is used, and the value of PBS_OUTPUT_HOST_NAME is appended after an at sign ("@"). If the job submitter specifies only a file path for standard output and standard error, the value of PBS_OUTPUT_HOST_NAME is appended after an at sign ("@"). If the job submitter specifies paths for standard output and standard error that include host names, the specified paths are used. Optional. If specified, must be a fully qualified domain name. Cannot contain a colon (":"). See "Delivering Output and Error Files" on page 62 in the PBS Professional Administrator's Guide.
PBS_PRIMARY	Hostname of primary server. Used only for failover configuration. Overrides PBS_SERVER_HOST_NAME. If you set PBS_LEAF_NAME on the primary server host, make sure that PBS_PRIMARY matches PBS_LEAF_NAME on the corresponding host. If you do not set PBS_LEAF_NAME on the server host, make sure that PBS_PRIMARY matches the hostname of the server host.
PBS_RCP	Location of `rcp` command if `rcp` is used.
PBS_SCHEDULER_SERVICE_PORT	Port on which default scheduler listens. Default value: 15004
PBS_SCP	Location of `scp` command if `scp` is used; setting this parameter causes PBS to first try `scp` rather than `rcp` for file transport.

Table 9-1: Parameters in `pbs.conf`

Parameter	Description
PBS_SECONDARY	Hostname of secondary server. Used only for failover configuration. Overrides PBS_SERVER_HOST_NAME. If you set PBS_LEAF_NAME on the secondary server host, make sure that PBS_SECONDARY matches PBS_LEAF_NAME on the corresponding host. If you do not set PBS_LEAF_NAME on the server host, make sure that PBS_SECONDARY matches the hostname of the server host.
PBS_SERVER	Hostname of host running the server. Cannot be longer than 255 characters. If the short name of the server host resolves to the correct IP address, you can use the short name for the value of the PBS_SERVER entry in `pbs.conf`. If only the FQDN of the server host resolves to the correct IP address, you must use the FQDN for the value of PBS_SERVER. Overridden by PBS_SERVER_HOST_NAME and PBS_PRIMARY.
PBS_SERVER_HOST_NAME	The FQDN of the server host. Used by clients to contact server. Overridden by PBS_PRIMARY and PBS_SECONDARY failover parameters. Overrides PBS_SERVER parameter. Optional. If specified, must be a fully qualified domain name. Cannot contain a colon (":"). See "Contacting the Server" on page 62 in the PBS Professional Administrator's Guide.
PBS_SMTP_SERVER_NAME	Name of SMTP server PBS will use to send mail. Should be a fully qualified domain name. Cannot contain a colon (":"). Available only under Windows. See section 2.2.3, "Specifying SMTP Server on Windows", on page 19.
PBS_START_COMM	Set this to *1* if a communication daemon is to run on this host.
PBS_START_MOM	Default is *0*. Set this to *1* if a MoM is to run on this host.
PBS_START_SCHED	**Deprecated**. Set this to *1* if default scheduler is to run on this host. Overridden by scheduler's scheduling attribute.
PBS_START_SERVER	Set this to *1* if server is to run on this host.

Table 9-1: Parameters in `pbs.conf`

Parameter	Description
PBS_SYSLOG=<value>	Controls use of `syslog` facility under which the entries are logged. Valid values: 0: no syslogging 1: logged via LOG_DAEMON facility 2: logged via LOG_LOCAL0 facility 3: logged via LOG_LOCAL1 facility ... 9: logged via LOG_LOCAL7 facility
PBS_SYSLOGSEVR=<value>	Filters `syslog` messages by severity. Valid values: 0: only LOG_EMERG messages are logged 1: messages up to LOG_ALERT are logged ... 7: messages up to LOG_DEBUG are logged
PBS_TMPDIR	Location of temporary files/directories used by PBS components.

10 Log Levels

10.1 Log Levels

PBS allows specification of the types of events that are logged for each daemon. Each type of log event has a different log level. All daemons use the same log level for the same type of event.

The following table lists the log level for each type of event.

Table 10-1: PBS Events and Log Levels

Name	Decimal	Hex	Event Description
PBSEVENT_ERROR	1	0x0001	Internal PBS errors
PBSEVENT_SYSTEM	2	0x0002	System (OS) errors, such as malloc failure
PBSEVENT_ADMIN	4	0x0004	Administrator-controlled events, such as changing queue attributes
PBSEVENT_JOB	8	0x0008	Job related events, e.g. submitted, ran, deleted
PBSEVENT_JOB_USAGE	16	0x0010	Job resource usage
PBSEVENT_SECURITY	32	0x0020	Security related events
PBSEVENT_SCHED	64	0x0040	When the scheduler was called and why
PBSEVENT_DEBUG	128	0x0080	Common debug messages
PBSEVENT_DEBUG2	256	0x0100	Debug event class 2
PBSEVENT_RESV	512	0x0200	Reservation-related messages
PBSEVENT_DEBUG3	1024	0x0400	Debug event class 3. Debug messages rarer than event class 2.
PBSEVENT_DEBUG4	2048	0x0800	Debug event class 4. Limit-related messages.

11
Job Exit Status

11.1 Job Exit Status

The exit status of a job may fall in one of three ranges, listed in the following table:

Table 11-1: Job Exit Status Ranges

Exit Status Range	Reason	Description
X < 0	The job could not be executed	This is the exit value of the top process in the job, typically the shell. This may be the exit value of the last command executed in the shell or the `.logout` script if the user has such a script (`csh`).
0 <=X < 128	Exit value of shell The exit status of an interactive job is always recorded as 0 (zero), regardless of the actual exit status.	This means the job was killed with a signal. The signal is given by X modulo 128 (or 256). For example an exit value of 137 means the job's top process was killed with signal 9 (137 % 128 = 9). The exit status values greater than 128 (or 256) indicate which signal killed the job. Depending on the system, values greater than 128 (or on some systems 256; see `wait(2)` or `waitpid(2)` for more information), are the value of the signal that killed the job. To interpret (or "decode") the signal contained in the exit status value, subtract the base value from the exit status. For example, if a job had an exit status of 143, that indicates the job was killed via a SIGTERM (e.g. 143 - 128 = 15, signal 15 is SIGTERM). See the `kill(1)` manual page for a mapping of signal numbers to signal name on your operating system.
X >=128	Job was killed with a signal	The exit status of jobs is recorded in the PBS server logs and the accounting logs.

Negative exit status indicates that the job could not be executed. Negative exit values are listed in the table below:

Table 11-2: Job Exit Codes

Exit Code	Name	Description
0	*JOB_EXEC_OK*	Job execution was successful
-1	*JOB_EXEC_FAIL1*	Job execution failed, before files, no retry
-2	*JOB_EXEC_FAIL2*	Job execution failed, after files, no retry
-3	*JOB_EXEC_RETRY*	Job execution failed, do retry
-4	*JOB_EXEC_INITABT*	Job aborted on MoM initialization
-5	*JOB_EXEC_INITRST*	Job aborted on MoM initialization, checkpoint, no migrate

Table 11-2: Job Exit Codes

Exit Code	Name	Description
-6	*JOB_EXEC_INITRMG*	Job aborted on MoM initialization, checkpoint, ok migrate
-7	*JOB_EXEC_BADRESRT*	Job restart failed
-10	*JOB_EXEC_FAILUID*	Invalid UID/GID for job
-11	*JOB_EXEC_RERUN*	Job was rerun
-12	*JOB_EXEC_CHKP*	Job was checkpointed and killed
-13	*JOB_EXEC_FAIL_PASSWORD*	Job failed due to a bad password
-14	*JOB_EXEC_RERUN_ ON_SIS_FAIL*	Job was requeued (if rerunnable) or deleted (if not) due to a communication failure between Mother Superior and a Sister
-15	JOB_EXEC_QUERST	Requeue job for restart from checkpoint
-16	JOB_EXEC_FAILHOOK_RERUN	Job execution failed due to hook rejection; requeue for later retry
-17	JOB_EXEC_FAILHOOK_DELETE	Job execution failed due to hook rejection; delete the job at end
-18	JOB_EXEC_HOOK_RERUN	A hook requested for job to be requeued
-19	JOB_EXEC_HOOK_DELETE	A hook requested for job to be deleted
-20	JOB_EXEC_RERUN_MS_FAIL	Mother superior connection failed

12 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

12.1 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. In this model, all PBS components run on the same host, which is the same host on which jobs will be executed. The following illustration shows how communication works when PBS is on a single host in TPP mode. For more on TPP mode, see Chapter 4, "Communication", on page 47.

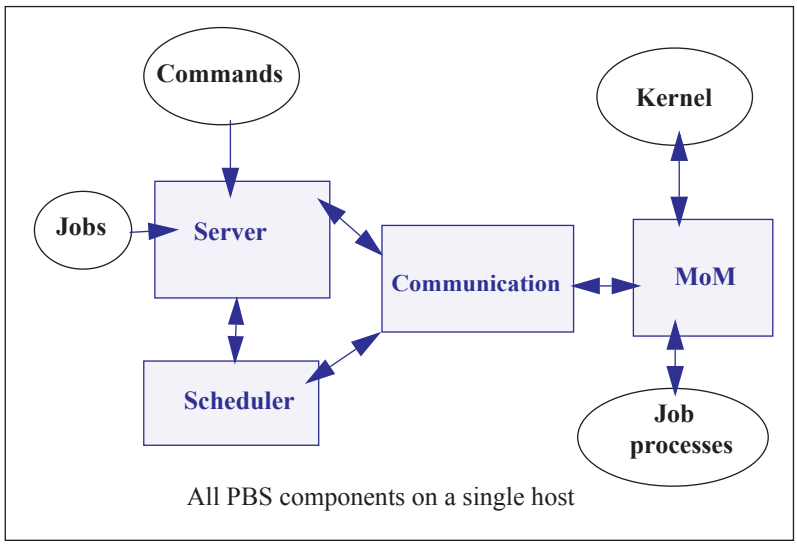

Figure 12-1:PBS daemons on a single execution host

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.

12.2 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, scheduler, and communication daemons, 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. The following illustration shows how communication works when the PBS server and scheduler are on a front-end system and MoM is on a separate host, in TPP mode. For more on TPP mode, see Chapter 4, "Communication", on page 47.

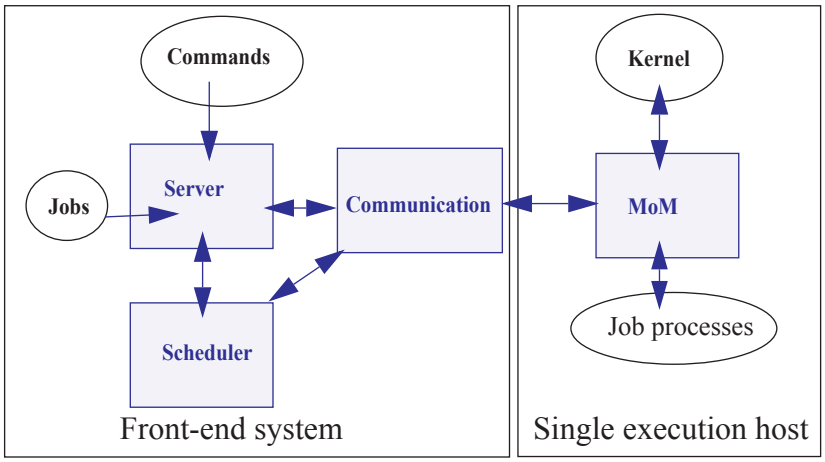

Figure 12-2:PBS daemons on single execution system with front end

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, 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 partition scheduler's `sched_priv/sched_config` file, the following options need to be set:

```
load_balancing: True all
```

12.3 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, scheduler, and communication daemons 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, 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 partition scheduler to load balance the workload across the available vnodes, making sure not to run two jobs in a row on the same vnode . We accomplish this by editing the scheduler configuration file and enabling load balancing:

```
load_balancing: True all
smp_cluster_dist: round_robin
```

The following diagram illustrates this for an eight-host complex in TPP mode.

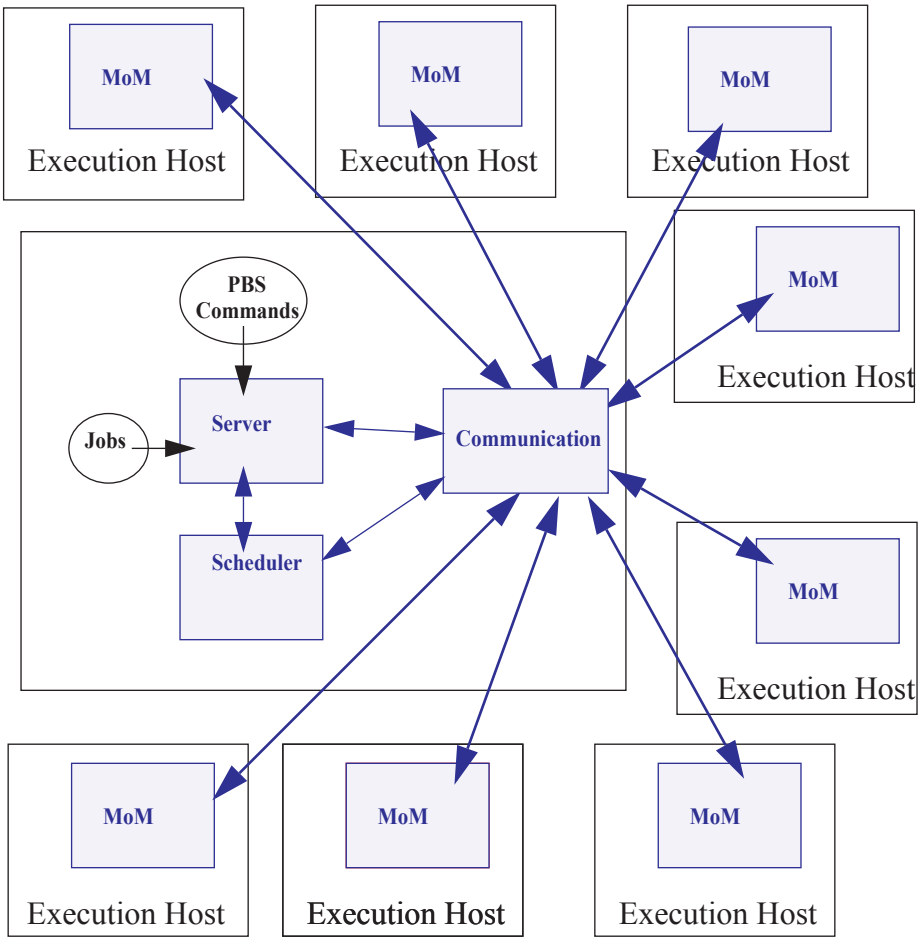

Figure 12-3:Typical PBS daemon locations for multiple execution hosts

This diagram illustrates a multi-vnode complex TPP configuration wherein the server and scheduler daemons communicate with the MoMs on the execution hosts via the communication daemon. Jobs are submitted to the server, scheduled for execution by the partition scheduler, and then transferred to a MoM when it's time to be run. MoM periodically sends status information back to the server, and answers resource requests from the scheduler.

12.4 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 = astronomy
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
```

12.5 External Software License Management

PBS Professional can be configured to schedule jobs based on externally-controlled licensed software. A detailed example is provided in "Example of Floating, Externally-managed License with Features" on page 273 in the PBS Professional Administrator's Guide.

12.6 Multiple User ACL Example

A site may have a need to restrict individual users to particular queues. In the previous example we set up queues with group-based ACLs, in this example we show user-based ACLs. Say a site has two different groups of users, and 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
```

13
Run Limit Error Messages

This chapter lists the error messages generated when limits are exceeded. See "Managing Resource Usage By Users, Groups, and Projects, at Server & Queues" on page 284 in the PBS Professional Administrator's Guide.

13.1 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.

Table 13-1: Job Run Limit Error Messages

Attribute	Where Applied	To What Applied	Message
max_run	queue	o: PBS_ALL	Not Running: Queue <queue name> 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.
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 <queue name> 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 <queue name> 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 <queue name> 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 <queue name> 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 <queue name> job limit reached on resource <resource name>
max_run_res	server	o: PBS_ALL	Server job limit reached on resource <resource name>

Table 13-1: Job Run Limit Error Messages

Attribute	Where Applied	To What Applied	Message
max_run_res	queue	p:PBS_GENERIC	Not Running: Queue <queue name> per-project limit reached on resource <resource name>
max_run_res	server	p:PBS_GENERIC	Not Running: Server per-project limit reached on resource <resource name>
max_run_res	server	p:<project name>	Not Running: would exceed project <project_name>'s limit on resource <resource name> in complex
max_run_res	queue	p:<project name>	Not Running: would exceed project <project_name>'s limit on resource <resource name> in queue <queue-name>
max_run_res	queue	g: PBS_GENERIC	Queue <queue name> per-group limit reached on resource <resource name>
max_run_res	server	g: PBS_GENERIC	Server per-group limit reached on resource <resource name>
max_run_res	queue	u: PBS_GENERIC	Queue <queue name> per-user limit reached on resource <resource name>
max_run_res	server	u: PBS_GENERIC	Server per-user limit reached on resource <resource name>
max_run_res	queue	g:<group name>	would exceed group <G>'s limit on resource <resource name> in queue <queue name>
max_run_res	server	g:<group name>	would exceed group <G>'s limit on resource <resource name> in complex
max_run_res	queue	u:<user name>	would exceed user <U>'s limit on resource <resource name> in queue <queue name>
max_run_res	server	u:<user name>	would exceed user <U>'s limit on resource <resource name> in complex

14
Error Codes

The following table lists all the PBS error codes, their textual names, and a description of each.

Table 14-1: 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
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

Table 14-1: Error Codes

Error Name	Error Code	Description
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
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
PBSE_EXLIMIT	15048	Limit exceeds allowable
PBSE_BADACCT	15049	Bad Account attribute value
PBSE_ALRDYEXIT	15050	Job already in exit state

Table 14-1: Error Codes

Error Name	Error Code	Description
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
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

Table 14-1: Error Codes

Error Name	Error Code	Description
PBSE_RESVAUTH_H	15092	Host not authorized to make AR
PBSE_RESVAUTH_G	15093	Group not authorized to make AR
PBSE_RESVAUTH_U	15094	User not authorized to make AR
PBSE_R_UID	15095	Bad effective UID for reservation
PBSE_R_GID	15096	Bad effective GID for reservation
PBSE_IBMSPSWITCH	15097	IBM SP Switch error
PBSE_LICENSEUNAV	15098	Floating License unavailable
	15099	UNUSED
PBSE_RESCNOTSTR	15100	Resource is not of type string
PBSE_SSIGNON_UNSET_REJECT	15101	rejected if SVR_ssignon_enable not set
PBSE_SSIGNON_SET_REJECT	15102	rejected if SVR_ssignon_enable set
PBSE_SSIGNON_BAD_TRANSITION1	15103	bad attempt: true to false
PBSE_SSIGNON_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 BlueGene
PBSE_NODESTALE	15115	Cannot change state of stale vnode
PBSE_DUPRESC	15116	cannot dupe 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
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

Table 14-1: Error Codes

Error Name	Error Code	Description
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
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_SCRIPTS	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
PBSE_HOOK_REJECT_DELETEJOB	15171	Hook rejection requiring a job to be deleted
PBSE_JOBNBIG	15173	Submitted job or reservation name is too long
Resource monitor specific error codes		

Table 14-1: Error Codes

Error Name	Error Code	Description
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_TRYAGAIN	15208	Try the request again later
PBSE_ALPSRELERR	15209	PBS is unable to release the ALPS reservation

15
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 15-1: Request Codes

Numeric Value	Name
0	PBS_BATCH_Connect
1	PBS_BATCH_QueueJob
2	UNUSED
3	PBS_BATCH_jobscript
4	PBS_BATCH_RdytoCommit
5	PBS_BATCH_Commit
6	PBS_BATCH_DeleteJob
7	PBS_BATCH_HoldJob
8	PBS_BATCH_LocateJob
9	PBS_BATCH_Manager
10	PBS_BATCH_MessJob
11	PBS_BATCH_ModifyJob
12	PBS_BATCH_MoveJob
13	PBS_BATCH_ReleaseJob
14	PBS_BATCH_Rerun
15	PBS_BATCH_RunJob
16	PBS_BATCH_SelectJobs
17	PBS_BATCH_Shutdown
18	PBS_BATCH_SignalJob
19	PBS_BATCH_StatusJob
20	PBS_BATCH_StatusQue
21	PBS_BATCH_StatusSvr
22	PBS_BATCH_TrackJob
23	PBS_BATCH_AsyrunJob
24	PBS_BATCH_Rescq
25	PBS_BATCH_ReserveResc

Table 15-1: Request Codes

Numeric Value	Name
26	PBS_BATCH_ReleaseResc
27	PBS_BATCH_FailOver
48	PBS_BATCH_StageIn
49	PBS_BATCH_AuthenUser
50	PBS_BATCH_OrderJob
51	PBS_BATCH_SelStat
52	PBS_BATCH_RegistDep
54	PBS_BATCH_CopyFiles
55	PBS_BATCH_DelFiles
56	PBS_BATCH_JobObit
57	PBS_BATCH_MvJobFile
58	PBS_BATCH_StatusNode
59	PBS_BATCH_Disconnect
60	UNUSED
61	UNUSED
62	PBS_BATCH_JobCred
63	PBS_BATCH_CopyFiles_Cred
64	PBS_BATCH_DelFiles_Cred
65	PBS_BATCH_GSS_Context
66	UNUSED
67	UNUSED
68	UNUSED
69	UNUSED
70	PBS_BATCH_SubmitResv
71	PBS_BATCH_StatusResv
72	PBS_BATCH_DeleteResv
73	PBS_BATCH_UserCred
74	PBS_BATCH_UserMigrate
75	PBS_BATCH_ConfirmResv
80	PBS_BATCH_DefSchReply
81	PBS_BATCH_StatusSched
82	PBS_BATCH_StatusRsc
83	PBS_BATCH_StatusHook

Table 15-1: Request Codes

Numeric Value	Name
84	PBS_BATCH_PySpawn
85	PBS_BATCH_CopyHookFile
86	PBS_BATCH_DelHookFile
87	PBS_BATCH_MomRestart
88	PBS_BATCH_AuthExternal
89	PBS_BATCH_HookPeriodic
90	PBS_BATCH_RelnodesJob
91	PBS_BATCH_ModifyResv

16
PBS Environment Variables

16.1 PBS Environment Variables

The following table lists the PBS environment variables:

Table 16-1: PBS Environment Variables

Variable	Origin	Meaning
NCPUS		Number of threads, defaulting to number of CPUs, on the vnode
OMP_NUM_THREADS		Same as NCPUS.
PBS_ARRAY_ID	Server	Identifier for job arrays. Consists of sequence number.
PBS_ARRAY_INDEX	Server	Index number of subjob in job array.
PBS_CONF_FILE		Path to `pbs.conf`
PBS_CPUSET_DEDICATED	Set by `mpiexec`	Asserts exclusive use of resources in assigned cpuset.
PBS_DEFAULT		Name of default PBS server
PBS_DATA_SERVICE_USER	Admin, during installation	Account used by data service.
PBS_ENVIRONMENT		Indicates job type: *PBS_BATCH* or *PBS_INTERACTIVE*
PBS_JOBCOOKIE		Unique identifier for inter-MoM job-based communication.
PBS_JOBDIR		Pathname of job-specific staging and execution directory
PBS_JOBID	Server	The job identifier assigned to the job or job array by the batch system.
PBS_JOBNAME	User	The job name supplied by the user.
PBS_LICENSE_INFO	Admin	Location of license info
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	Submission environment	Value of HOME from submission environment.
PBS_O_HOST	Submission environment; set by PBS	The host name on which the `qsub` command was executed.
PBS_O_LANG	Submission environment	Value of LANG from submission environment
PBS_O_LOGNAME	Submission environment	Value of LOGNAME from submission environment

Table 16-1: PBS Environment Variables

Variable	Origin	Meaning
PBS_O_MAIL	Submission environment	Value of MAIL from submission environment
PBS_O_PATH	Submission environment	Value of PATH from submission environment
PBS_O_QUEUE	Submission environment	The original queue name to which the job was submitted.
PBS_O_SHELL	Submission environment	Value of SHELL from submission environment
PBS_O_SYSTEM	Submission environment	The operating system name where `qsub` was executed.
PBS_O_TZ	Submission environment	Value of TZ from submission environment
PBS_O_WORKDIR	Submission environment	The absolute path of directory where `qsub` was executed.
PBS_QUEUE		The name of the queue from which the job is executed.
PBS_SERVER	Submission environment	The name of the default PBS server.
PBS_TASKNUM		The task (process) number for the job on this vnode.
PBS_TMPDIR		Root of temporary directories/files for PBS components.
TMPDIR		The job-specific temporary directory for this job.

17
File Listing

The following table lists all the PBS files and directories; owner and permissions are specific to Linux systems.

Table 17-1: File Listing

Directory / File	Owner	Permission	Average Size
/opt/pbs/default/etc/pbs_bootcheck.py	root	-rw-r--r--	4111
/var/tmp/pbs_bootcheck.py	root	-rw-r--r--	4111
/var/tmp/pbs_boot_check See "Discovering Last Reboot Time of Server" on page 548 in the PBS Professional Administrator's Guide.	root	-rw-r--r--	188
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
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

Table 17-1: File Listing

Directory / File	Owner	Permission	Average Size
PBS_EXEC/bin/qrerun	root	-rwxr-xr-x	157228
PBS_EXEC/bin/qrls	root	-rwxr-xr-x	165361
PBS_EXEC/bin/qrun	root	-rwxr-xr-x	160978
PBS_EXEC/bin/qselect	root	-rwxr-xr-x	163266
PBS_EXEC/bin/qsig	root	-rwxr-xr-x	160083
PBS_EXEC/bin/qstart	root	-rwxr-xr-x	139589
PBS_EXEC/bin/qstat	root	-rwxr-xr-x	207532
PBS_EXEC/bin/qstop	root	-rwxr-xr-x	139584
PBS_EXEC/bin/qsub	root	-rwxr-xr-x	275460
PBS_EXEC/bin/qterm	root	-rwxr-xr-x	132188
PBS_EXEC/bin/tracejob	root	-rwxr-xr-x	64730
PBS_EXEC/etc	root	drwxr-xr-x	4096
PBS_EXEC/etc/modulefile	root	-rw-r--r--	749
PBS_EXEC/etc/pbs_db_schema.sql	root	-rw-r--r--	10522
PBS_EXEC/etc/pbs_dedicated	root	-rw-r--r--	557
PBS_EXEC/etc/pbs_holidays	root	-rw-r--r--	2612
PBS_EXEC/etc/pbs_resource_group	root	-rw-r--r--	657
PBS_EXEC/etc/pbs_sched_config	root	-r--r--r--	9791
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
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

Table 17-1: File Listing

Directory / File	Owner	Permission	Average Size
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/libexec/au-nodeupdate.pl	root	-rw-r--r--	
PBS_EXEC/libexec/install_db	root	-rwx------	10506
PBS_EXEC/libexec/pbs_habitat	root	-rwx------	10059
PBS_EXEC/libexec/pbs_init.d	root	-rwx------	25568
PBS_EXEC/libexec/pbs_postinstall	root	-rwx------	29104
PBS_EXEC/share/man	root	drwxr-xr-x	4096
PBS_EXEC/share/man/man1	root	drwxr-xr-x	4096
PBS_EXEC/share/man/man1/nqs2pbs	root	-rw-r--r--	3276
PBS_EXEC/share/man/man1/pbs.1B	root	-rw-r--r--	5376
PBS_EXEC/share/man/man1/pbsdsh.1B	root	-rw-r--r--	2978
PBS_EXEC/share/man/man1/pbs_rdel.1B	root	-rw-r--r--	2342
PBS_EXEC/share/man/man1/pbs_rstat.1B	root	-rw-r--r--	2682
PBS_EXEC/share/man/man1/pbs_rsub.1B	root	-rw-r--r--	9143
PBS_EXEC/share/man/man1/qalter.1B	root	-rw-r--r--	21569
PBS_EXEC/share/man/man1/qdel.1B	root	-rw-r--r--	3363
PBS_EXEC/share/man/man1/qhold.1B	root	-rw-r--r--	4323
PBS_EXEC/share/man/man1/qmove.1B	root	-rw-r--r--	3343
PBS_EXEC/share/man/man1/qmsg.1B	root	-rw-r--r--	3244
PBS_EXEC/share/man/man1/qorder.1B	root	-rw-r--r--	3028
PBS_EXEC/share/man/man1/qrerun.1B	root	-rw-r--r--	2965
PBS_EXEC/share/man/man1/qrls.1B	root	-rw-r--r--	3927
PBS_EXEC/share/man/man1/qselect.1B	root	-rw-r--r--	12690
PBS_EXEC/share/man/man1/qsig.1B	root	-rw-r--r--	3817

Table 17-1: File Listing

Directory / File	Owner	Permission	Average Size
PBS_EXEC/share/man/man1/qstat.1B	root	-rw-r--r--	15274
PBS_EXEC/share/man/man1/qsub.1B	root	-rw-r--r--	36435
PBS_EXEC/share/man/man3	root	drwxr-xr-x	4096
PBS_EXEC/share/man/man3/pbs_alterjob.3B	root	-rw-r--r--	5475
PBS_EXEC/share/man/man3/pbs_connect.3B	root	-rw-r--r--	3493
PBS_EXEC/share/man/man3/pbs_default.3B	root	-rw-r--r--	2150
PBS_EXEC/share/man/man3/pbs_deljob.3B	root	-rw-r--r--	3081
PBS_EXEC/share/man/man3/pbs_disconnect.3B	root	-rw-r--r--	1985
PBS_EXEC/share/man/man3/pbs_geterrmsg.3B	root	-rw-r--r--	2473
PBS_EXEC/share/man/man3/pbs_holdjob.3B	root	-rw-r--r--	3006
PBS_EXEC/share/man/man3/pbs_manager.3B	root	-rw-r--r--	4337
PBS_EXEC/share/man/man3/pbs_movejob.3B	root	-rw-r--r--	3220
PBS_EXEC/share/man/man3/pbs_msgjob.3B	root	-rw-r--r--	2912
PBS_EXEC/share/man/man3/pbs_orderjob.3B	root	-rw-r--r--	2526
PBS_EXEC/share/man/man3/pbs_rerunjob.3B	root	-rw-r--r--	2531
PBS_EXEC/share/man/man3/pbs_rlsjob.3B	root	-rw-r--r--	3043
PBS_EXEC/share/man/man3/pbs_runjob.3B	root	-rw-r--r--	3484
PBS_EXEC/share/man/man3/pbs_selectjob.3B	root	-rw-r--r--	7717
PBS_EXEC/share/man/man3/pbs_sigjob.3B	root	-rw-r--r--	3108
PBS_EXEC/share/man/man3/pbs_statjob.3B	root	-rw-r--r--	4618
PBS_EXEC/share/man/man3/pbs_statnode.3B	root	-rw-r--r--	3925
PBS_EXEC/share/man/man3/pbs_statque.3B	root	-rw-r--r--	4009
PBS_EXEC/share/man/man3/pbs_statserver.3B	root	-rw-r--r--	3674
PBS_EXEC/share/man/man3/pbs_submit.3B	root	-rw-r--r--	6320
PBS_EXEC/share/man/man3/pbs_submitresv.3B	root	-rw-r--r--	3878
PBS_EXEC/share/man/man3/pbs_terminate.3B	root	-rw-r--r--	3322
PBS_EXEC/share/man/man3/tm.3B	root	-rw-r--r--	11062
PBS_EXEC/share/man/man7	root	drwxr-xr-x	4096
PBS_EXEC/share/man/man7/pbs_job_attributes.7B	root	-rw-r--r--	15920
PBS_EXEC/share/man/man7/pbs_node_attributes.7B	root	-rw-r--r--	7973
PBS_EXEC/share/man/man7/pbs_queue_attributes.7B	root	-rw-r--r--	11062
PBS_EXEC/share/man/man7/pbs_resources.7B	root	-rw-r--r--	22124

Table 17-1: File Listing

Directory / File	Owner	Permission	Average Size
PBS_EXEC/share/man/man7/pbs_resv_attributes.7B	root	-rw-r--r--	11662
PBS_EXEC/share/man/man7/pbs_server_attributes.7B	root	-rw-r--r--	14327
PBS_EXEC/share/man/man8	root	drwxr-xr-x	4096
PBS_EXEC/share/man/man8/mpiexec.8B	root	-rw-r--r--	4701
PBS_EXEC/share/man/man8/pbs-report.8B	root	-rw-r--r--	19221
PBS_EXEC/share/man/man8/pbsfs.8B	root	-rw-r--r--	3703
PBS_EXEC/share/man/man8/pbsnodes.8B	root	-rw-r--r--	3441
PBS_EXEC/share/man/man8/pbsrun.8B	root	-rw-r--r--	20937
PBS_EXEC/share/man/man8/pbsrun_unwrap.8B	root	-rw-r--r--	2554
PBS_EXEC/share/man/man8/pbsrun_wrap.8B	root	-rw-r--r--	3855
PBS_EXEC/share/man/man8/pbs_attach.8B	root	-rw-r--r--	3790
PBS_EXEC/share/man/man8/pbs_hostn.8B	root	-rw-r--r--	2781
PBS_EXEC/share/man/man8/pbs_idled.8B	root	-rw-r--r--	2628
PBS_EXEC/share/man/man8/pbs_lamboot.8B	root	-rw-r--r--	2739
PBS_EXEC/share/man/man8/pbs_migrate_users.8B	root	-rw-r--r--	2519
PBS_EXEC/share/man/man8/pbs_mom.8B	root	-rw-r--r--	23496
PBS_EXEC/share/man/man8/pbs_mpihp.8B	root	-rw-r--r--	4120
PBS_EXEC/share/man/man8/pbs_mpilam.8B	root	-rw-r--r--	2647
PBS_EXEC/share/man/man8/pbs_mpirun.8B	root	-rw-r--r--	3130
PBS_EXEC/share/man/man8/pbs_password.8B	root	-rw-r--r--	3382
PBS_EXEC/share/man/man8/pbs_probe.8B	root	-rw-r--r--	3344
PBS_EXEC/share/man/man8/pbs_sched_cc.8B	root	-rw-r--r--	6731
PBS_EXEC/share/man/man8/pbs_server.8B	root	-rw-r--r--	7914
PBS_EXEC/share/man/man8/pbs_tclsh.8B	root	-rw-r--r--	2475
PBS_EXEC/share/man/man8/pbs_tmrsh.8B	root	-rw-r--r--	3556
PBS_EXEC/share/man/man8/pbs_wish.8B	root	-rw-r--r--	2123
PBS_EXEC/share/man/man8/printjob.8B	root	-rw-r--r--	2823
PBS_EXEC/share/man/man8/qdisable.8B	root	-rw-r--r--	3104
PBS_EXEC/share/man/man8/qenable.8B	root	-rw-r--r--	2937
PBS_EXEC/share/man/man8/qmgr.8B	root	-rw-r--r--	7282
PBS_EXEC/share/man/man8/qrun.8B	root	-rw-r--r--	2850
PBS_EXEC/share/man/man8/qstart.8B	root	-rw-r--r--	2966

Table 17-1: File Listing

Directory / File	Owner	Permission	Average Size
PBS_EXEC/share/man/man8/qstop.8B	root	-rw-r--r--	2963
PBS_EXEC/share/man/man8/qterm.8B	root	-rw-r--r--	4839
PBS_EXEC/share/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
PBS_EXEC/tcltk/include	root	drwxr-xr-x	4096
PBS_EXEC/tcltk/include/tcl.h	root	-rw-r--r--	57222
PBS_EXEC/tcltk/include/tclDecls.h	root	-rw-r--r--	123947
PBS_EXEC/tcltk/include/tk.h	root	-rw-r--r--	47420
PBS_EXEC/tcltk/include/tkDecls.h	root	-rw-r--r--	80181
PBS_EXEC/tcltk/lib	root	drwxr-xr-x	4096
PBS_EXEC/tcltk/lib/libtcl8.3.a	root	-rw-r--r--	777558
PBS_EXEC/tcltk/lib/libtclstub8.3.a	root	-rw-r--r--	1832
PBS_EXEC/tcltk/lib/libtk8.3.a	root	-rw-r--r--	1021024
PBS_EXEC/tcltk/lib/libtkstub8.3.a	root	-rw-r--r--	3302
PBS_EXEC/tcltk/lib/tcl8.3	root	drwxr-xr-x	4096

Table 17-1: File Listing

Directory / File	Owner	Permission	Average Size
PBS_EXEC/tcltk/lib/tclConfig.sh	root	-rw-r--r--	7076
PBS_EXEC/tcltk/lib/tk8.3	root	drwxr-xr-x	4096
PBS_EXEC/tcltk/lib/tkConfig.sh	root	-rw-r--r--	3822
PBS_EXEC/tcltk/license.terms	root	-rw-r--r--	2233
PBS_HOME	root	drwxr-xr-x	4096
PBS_HOME/aux	root	drwxr-xr-x	4096
PBS_HOME/checkpoint	root	drwx------	4096
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_log	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/accounting	root	drwxr-xr-x	4096
PBS_HOME/server_priv/acl_groups	root	drwxr-x---	4096
PBS_HOME/server_priv/acl_hosts	root	drwxr-x---	4096
PBS_HOME/server_priv/acl_svr	root	drwxr-x---	4096
PBS_HOME/server_priv/acl_svr/managers	root	-rw-------	13
PBS_HOME/server_priv/acl_users	root	drwxr-x---	4096
PBS_HOME/server_priv/jobs	root	drwxr-x---	4096

Table 17-1: File Listing

Directory / File	Owner	Permission	Average Size
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/resourcedef	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

18 Introduction to PBS

18.1 Acknowledgements

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

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

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

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

Index

B

C

D

E

F

M

N

O

P

Q

R

S