Commands Overview

Client Commands

Command	Description
momctl	Manage/diagnose MOM (node execution) daemon
pbsdsh	Launch tasks within a parallel job
pbsnodes	View/modify batch status of compute nodes
qalter	Modify queued batch jobs
qchkpt	Checkpoint batch jobs
qdel	Delete/cancel batch jobs
qgpumode	Specifies new mode for GPU
qgpureset	Reset the GPU
qhold	Hold batch jobs
qmgr	Manage policies and other batch configuration
qmove	Move batch jobs
qorder	Exchange order of two batch jobs in any queue
qrerun	Rerun a batch job
qrls	Release batch job holds
qrun	Start a batch job
qsig	Send a signal to a batch job

Command	Description
qstat	View queues and jobs
qsub	Submit jobs
qterm	Shutdown pbs server daemon
tracejob	Trace job actions and states recorded in Torque logs (see Using "tracejob" to Locate Job Failures)

Binary Executables

Command	Description
pbs_mom	Start MOM (node execution) daemon
pbs_server	Start server daemon
pbs_track	Tell pbs_mom to track a new process

Related Topics

Node Manager (MOM) Configuration

Server Parameters

momctl

(PBS MOM Control)

Synopsis

```
momctl -c { <JOBID> | all }
momctl -C
momctl -d { <INTEGER> | <JOBID> }
momctl -f <FILE>
momctl -h <HOST>[,<HOST>]...
momctl -l
momctl -p <PORT_NUMBER>
momctl -q <ATTRIBUTE>
momctl -r { <FILE> | LOCAL:<FILE> }
momctl -s
momctl -u
```

Overview

The momctl command allows remote shutdown, reconfiguration, diagnostics, and querying of the pbs_mom daemon.

Format

-c — Clear	
Format	{ <JOBID> | *all* }
Default	---
Description	Makes the MOM unaware of the job's existence. It does not clean up any processes associated with the job.
Example	`momctl -c 2 -h node1`

-C — Cycle	
Format	---
Default	---
Description	Cycle pbs_mom (force the MOM to send a status update to pbs_server).
Example	`momctl -C -h node1` *Cycle pbs_mom on node1.*

-d — Diagnose	
Format	{ <INTEGER> | <JOBID> }
Default	0
Description	Diagnose MOM(s). (For more details, see Diagnose detail below.)

-d — Diagnose

Example	`momctl -d 2 -h node1` *Print level 2 and lower diagnostic information for the MOM on node1.*

-f — Host File

Format	<FILE>
Default	---
Description	A file containing only comma or whitespace (space, tab, or new line) delimited hostnames.
Example	`momctl -f hosts.txt -d 0` *Print diagnose information for the MOMs running on the hosts specified in* hosts.txt .

-h — Host List

Format	<HOST>[,<HOST>]...
Default	localhost
Description	A comma separated list of hosts.
Example	`momctl -h node1,node2,node3 -d 0` *Print diagnose information for the MOMs running on node1, node2, and node3.*

-l — Layout

Format	---
Default	---

-l — Layout

Description	Display the layout of the machine as it is understood by Torque.
Example	

```
[root@c04a numa]# momctl -l
Machine (50329748KB)
 Socket 0 (33552532KB)
  Chip 0 (16775316KB)
   Core 0 (1 threads)
   Core 1 (1 threads)
   Core 2 (1 threads)
   Core 3 (1 threads)
   Core 4 (1 threads)
   Core 5 (1 threads)
   Core 6 (1 threads)
   Core 7 (1 threads)
  Chip 1 (16777216KB)
   Core 8 (1 threads)
   Core 9 (1 threads)
   Core 10 (1 threads)
   Core 11 (1 threads)
   Core 12 (1 threads)
   Core 13 (1 threads)
   Core 14 (1 threads)
   Core 15 (1 threads)
 Socket 1 (16777216KB)
  Chip 2 (8388608KB)
   Core 16 (1 threads)
   Core 17 (1 threads)
   Core 18 (1 threads)
   Core 19 (1 threads)
   Core 20 (1 threads)
   Core 21 (1 threads)
   Core 22 (1 threads)
   Core 23 (1 threads)
  Chip 3 (8388608KB)
   Core 24 (1 threads)
   Core 25 (1 threads)
   Core 26 (1 threads)
   Core 27 (1 threads)
   Core 28 (1 threads)
   Core 29 (1 threads)
   Core 30 (1 threads)
   Core 31 (1 threads)
```

-p — Port

Format	<PORT_NUMBER>
Default	Torque's default port number.
Description	The port number for the specified MOM(s).

-p — Port

Example	`momctl -p 5455 -h node1 -d 0` *Request diagnose information over port 5455 on node1.*

-q — Query

Format	<ATTRIBUTE>
Default	---
Description	Query or set <ATTRIBUTE>, where <*ATTRIBUTE*> is a property listed by **pbsnodes** `-a` (see Query attributes for a list of attributes).

Example	`momctl -q physmem`
	Print the amount of physmem on localhost.
	`momctl -h node2 -q loglevel=7`
	Change the current MOM logging on node2 to level 7.

-r — Reconfigure

Format	{ <FILE> | LOCAL:<FILE> }
Default	---
Description	Reconfigure MOM(s) with remote or local config file, <FILE>. This does not work if $remote_reconfig is not set to true when the MOM is started.
Example	`momctl -r /home/user1/new.config -h node1` *Reconfigure MOM on node1 with* /home/user1/new.cofig *on node1.*

-s — Shutdown	
Format	---
Default	---
Description	Have the MOM shut itself down gracefully (this includes reporting to server so that **pbsnodes** marks the node down).
Example	`momctl -s` *Shut down the pbs_mom process on localhost.*

-u — Update	
Format	---
Default	---
Description	Update the hardware configuration on pbs_server with a layout from the MOM.
Example	`momctl -u` *Update pbs_server hardware configuration.*

Query attributes

Attribute	Description
arch	node hardware architecture
availmem	available RAM
loadave	1 minute load average
ncpus	number of CPUs available on the system
netload	total number of bytes transferred over all network interfaces

Attribute	Description
nsessions	number of sessions active
nusers	number of users active
physmem	configured RAM
sessions	list of active sessions
totmem	configured RAM plus configured swap

Diagnose detail

Level	Description
0	Display the following information: ı Local hostname ı Expected server hostname ı Execution version ı MOM home directory ı MOM config file version (if specified) ı Duration MOM has been executing ı Duration since last request from pbs_server daemon ı Duration since last request to pbs_server daemon ı RM failure messages (if any) ı Log verbosity level ı Local job list
1	All information for level 0 plus the following: ı Interval between updates sent to server ı Number of initialization messages sent to pbs_server daemon ı Number of initialization messages received from pbs_server daemon ı Prolog/epilog alarm time ı List of trusted clients

Level	Description
2	All information from level 1 plus the following: - PID - Event alarm status
3	All information from level 2 plus the following: - syslog enabled

Example A-1: MOM diagnostics

```
momctl -d 1

Host: nsrc/nsrc.fllcl.com    Server: 10.10.10.113    Version: torque_1.1.0p4
HomeDirectory:                  /usr/spool/PBS/mom_priv
ConfigVersion:          147
MOM active:             7390 seconds
Last Msg From Server:   7389 seconds (CLUSTER_ADDRS)
Server Update Interval: 20 seconds
Server Update Interval: 20 seconds
Init Msgs Received:       0 hellos/1 cluster-addrs
Init Msgs Sent:               1 hellos
LOGLEVEL:                           0 (use SIGUSR1/SIGUSR2 to adjust)
Prolog Alarm Time:      300 seconds
Trusted Client List:    12.14.213.113,127.0.0.1
JobList:                             NONE

diagnostics complete
```

Example A-2: System shutdown

```
> momctl -s -f /opt/clusterhostfile

shutdown request successful on node001
shutdown request successful on node002
shutdown request successful on node003
shutdown request successful on node004
shutdown request successful on node005
shutdown request successful on node006
```

pbs_mom

Start a pbs batch execution mini-server.

Synopsis

pbs_mom [-a alarm] [-A alias] [-C chkdirectory] [-c config] [-d directory] [-f] [-F] [-h help] [-H hostname] [-L logfile] [-M MOMport] [-R RMPport] [-p|-r] [-P purge] [-w] [-x]

Description

The pbs_mom command is located within the `TORQUE_HOME` directory and starts the operation of a batch Machine Oriented Mini-server (MOM) on the execution host. To ensure that the pbs_mom command is not runnable by the general user community, the server will only execute if its real and effective uid is zero.

The first function of pbs_mom is to place jobs into execution as directed by the server, establish resource usage limits, monitor the job's usage, and notify the server when the job completes. If they exist, pbs_mom will execute a prologue script before executing a job and an epilogue script after executing the job.

The second function of pbs_mom is to respond to resource monitor requests. This was done by a separate process in previous versions of PBS but has now been combined into one process. It provides information about the status of running jobs, memory available, etc.

The last function of pbs_mom is to respond to task manager requests. This involves communicating with running tasks over a TCP socket as well as communicating with other MOMs within a job (a.k.a. a "sisterhood").

pbs_mom will record a diagnostic message in a log file for any error occurrence. The log files are maintained in the mom_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.

Options

Flag	Name	Description
-a	alarm	Specifies the alarm timeout in seconds for computing a resource. Every time a resource request is processed, an alarm is set for the given amount of time. If the request has not completed before the given time, an alarm signal is generated. The default is 5 seconds.
-A	alias	Specifies this multimom's alias name. The alias name needs to be the same name used in the mom.hierarchy file. It is only needed when running multiple MOMs on the same machine. For more information, see Torque Multi-MOM.
-c	config	Specifies an alternative configuration file, see description below. If this is a relative file name it will be relative to `TORQUE_HOME/mom_priv` (see the **-d** option). If the specified file cannot be opened, pbs_mom will abort. If the **-c** option is not supplied, pbs_mom will attempt to open the default configuration file "config" in `TORQUE_HOME/mom_priv`. If this file is not present, pbs_mom will log the fact and continue.
-C	chkdirectory	Specifies the path of the directory used to hold checkpoint files. (Currently this is only valid on Cray systems.) The default directory is `TORQUE_HOME/spool/checkpoint` (see the **-d** option). The directory specified with the -C option must be owned by root and accessible (rwx) only by root to protect the security of the checkpoint files.

Flag	Name	Description
-d	directory	Specifies the path of the directory which is the home of the server's working files, `TORQUE_HOME` This option is typically used along with -M when debugging MOM. The default directory is given by `$PBS_SERVER_HOME`which is typically `/usr/spool/PBS`
-f	force_ update	Forces the server to accept an update of the hardware on the node. Should be used the first time pbs_mom is run after a hardware update on a node.
-F	fork	Do not fork. This option is useful when running under systemd (Red Hat 7-based or SUSE 12-based systems).
-h	help	Displays the help/usage message.
-H	hostname	Sets the MOM's hostname. This can be useful on multi-homed networks.
-L	logfile	Specifies an absolute path name for use as the log file. If not specified, MOM will open a file named for the current date in the `TORQUE_HOME/mom_logs`directory (see the -d option).
-M	port	Specifies the port number on which the mini-server (MOM) will listen for batch requests.
-p	n/a	Specifies the impact on jobs which were in execution when the mini-server shut down. On any restart of MOM, the new mini-server will not be the parent of any running jobs, MOM has lost control of her offspring (not a new situation for a mother). With the -p option, MOM will allow the jobs to continue to run and monitor them indirectly via polling. This flag is redundant in that this is the default behavior when starting the server. The -p option is mutually exclusive with the -r and -q options.
-P	purge	Specifies the impact on jobs which were in execution when the mini-server shut down. With the -P option, it is assumed that either the entire system has been restarted or the MOM has been down so long that it can no longer guarantee that the pid of any running process is the same as the recorded job process pid of a recovering job. Unlike the -p option, no attempt is made to try and preserve or recover running jobs. All jobs are terminated and removed from the queue.
-q	n/a	Specifies the impact on jobs which were in execution when the mini-server shut down. With the -q option, MOM will allow the processes belonging to jobs to continue to run, but will not attempt to monitor them. The -q option is mutually exclusive with the -p and -r options.

Flag	Name	Description
-r	n/a	Specifies the impact on jobs which were in execution when the mini-server shut down. With the -r option, MOM will kill any processes belonging to jobs, mark the jobs as terminated, and notify the batch server which owns the job. The -r option is mutually exclusive with the **-p** and **-q** options. Normally the mini-server is started from the system boot file without the **-p** or the -r option. The mini-server will make no attempt to signal the former session of any job which may have been running when the mini-server terminated. It is assumed that on reboot, all processes have been killed. If the -r option is used following a reboot, process IDs (pids) may be reused and MOM may kill a process that is not a batch session.
-R	port	Specifies the port number on which the mini-server (MOM) will listen for resource monitor requests, task manager requests and inter-MOM messages. Both a UDP and a TCP port of this number will be used.
-w	wait_for_ server	When started with -w, pbs_moms wait until they get their MOM hierarchy file from pbs_ server to send their first update, or until 10 minutes pass. This reduces network traffic on startup and can bring up clusters faster.
-x	n/a	Disables the check for privileged port resource monitor connections. This is used mainly for testing since the privileged port is the only mechanism used to prevent any ordinary user from connecting.

Configuration file

The configuration file, located at `mom_priv/config` by default, can be specified on the command line at program start with the **-c** flag. The use of this file is to provide several types of run time information to pbs_mom: static resource names and values, external resources provided by a program to be run on request via a shell escape, and values to pass to internal set up functions at initialization (and re-initialization).

See the Parameters page for a full list of pbs_mom parameters.

Each item type is on a single line with the component parts separated by white space. If the line starts with a hash mark (pound sign, #), the line is considered to be a comment and is skipped.

Static Resources

For static resource names and values, the configuration file contains a list of resource names/values pairs, one pair per line and separated by white space. An example of static resource names and values could be the number of tape drives of different types and could be specified by:

- tape3480 4
- tape3420 2

- tapedat 1
- tape8mm 1

Shell Commands

If the first character of the value is an exclamation mark (*!*), the entire rest of the line is saved to be executed through the services of the system(3) standard library routine.

The shell escape provides a means for the resource monitor to yield arbitrary information to the scheduler. Parameter substitution is done such that the value of any qualifier sent with the query, as explained below, replaces a token with a percent sign (%) followed by the name of the qualifier. For example, here is a configuration file line which gives a resource name of "escape":

```
escape !echo %xxx %yyy
```

If a query for "escape" is sent with no qualifiers, the command executed would be `echo %xxx %yyy` .

If one qualifier is sent, `escape[xxx=hi there]` , the command executed would be `echo hi there %yyy` .

If two qualifiers are sent, `escape[xxx=hi][yyy=there]` , the command executed would be `echo hi there` .

If a qualifier is sent with no matching token in the command line, `escape [zzz=snafu]` , an error is reported.

Resources

Resource Manager queries can be made with **momctl** -q options to retrieve and set pbs_mom options. Any configured static resource may be retrieved with a request of the same name. These are resource requests not otherwise documented in the PBS ERS.

Request	Description
cycle	Forces an immediate MOM cycle.
status_update_time	Retrieve or set the $status_update_time parameter.
check_poll_time	Retrieve or set the $check_poll_time parameter.
configversion	Retrieve the config version.
jobstartblocktime	Retrieve or set the $jobstartblocktime parameter.

Request	Description
enablemomrestart	Retrieve or set the $enablemomrestart parameter.
loglevel	Retrieve or set the $loglevel parameter.
down_on_error	Retrieve or set the $down_on_error parameter.
diag0 - diag4	Retrieves varied diagnostic information.
rcpcmd	Retrieve or set the $rcpcmd parameter.
version	Retrieves the pbs_mom version.

Health check

The health check script is executed directly by the pbs_mom daemon under the root user id. It must be accessible from the compute node and may be a script or compiled executable program. It may make any needed system calls and execute any combination of system utilities but should not execute resource manager client commands. Also, the pbs_mom daemon blocks until the health check is completed and does not possess a built-in timeout. Consequently, it is advisable to keep the launch script execution time short and verify that the script will not block even under failure conditions.

If the script detects a failure, it should return the `ERROR` keyword to `stdout` followed by an error message. The message (up to 1024 characters) immediately following the **ERROR** string will be assigned to the node attribute message of the associated node.

If the script detects a failure when run from "jobstart", then the job will be rejected. You can use this behavior with an advanced scheduler, such as Moab Workload Manager, to cause the job to be routed to another node. Torque currently ignores Error messages by default, but you can configure an advanced scheduler to react appropriately.

If the `$down_on_error` MOM setting is enabled, the MOM will set itself to state down and report to pbs_server. Additionally, the `$down_on_error server` attribute can be enabled which has the same effect but moves the decision to pbs_server. It is redundant to have MOM's `$down_on_error` and pbs_server's `down_on_error` features enabled. Also see down_on_error (in **Server Parameters**).

See Creating the Health Check Script on page 218 for more information.

Files

File	Description
$PBS_SERVER_HOME/server_name	Contains the hostname running pbs_server
$PBS_SERVER_HOME/mom_priv	The default directory for configuration files, typically (`/usr/spool/pbs`)`/mom_priv`
$PBS_SERVER_HOME/mom_logs	Directory for log files recorded by the server
$PBS_SERVER_HOME/mom_priv/-prologue	The administrative script to be run before job execution
$PBS_SERVER_HOME/mom_priv/e-pilogue	The administrative script to be run after job execution

Signal handling

pbs_mom handles the following signals:

Signal	Description
SIGHUP	Causes pbs_mom to re-read its configuration file, close and reopen the log file, and reinitialize resource structures.
SIGALRM	Results in a log file entry. The signal is used to limit the time taken by certain children processes, such as the prologue and epilogue.
SIGINT and SIGTERM	Results in pbs_mom exiting without terminating any running jobs. This is the action for the following signals as well: SIGXCPU, SIGXFSZ, SIGCPULIM, and SIGSHUTDN.
SIGUSR1, SIGUSR2	Causes the MOM to increase and decrease logging levels, respectively.
SIGPIPE, SIGINFO	Are ignored.
SIGBUS, SIGFPE, SIGILL, SIGTRAP, and SIGSYS	Cause a core dump if the PBSCOREDUMP environmental variable is defined.

All other signals have their default behavior installed.

Exit status

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

Related Topics

pbs_server(8B)

Non-Adaptive Computing topics

- pbs_scheduler_basl(8B)
- pbs_scheduler_tcl(8B)
- PBS External Reference Specification
- PBS Administrators Guide

pbs_server

(*PBS Server*) pbs batch system manager

Synopsis

```
pbs_server [-a active] [-A acctfile] [-c] [-d config_path] [-
F] [-f force overwrite] [-H hostname] [--ha] [-l location] [-L
logfile] [-n don't send hierarchy] [-p port] [-S scheduler_
port] [-t type] [-v] [--about] [--version]
```

Description

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

The server will record a diagnostic message in a log file for any error occurrence. The log files are maintained in the server_logs directory below the home directory of the server. If the log file cannot be opened, the diagnostic message is written to the system console.

As of Torque 4.0, the pbs_server is multi-threaded which leads to quicker response to client commands, is more robust, and allows for higher job throughput.

Options

Option	Name	Description
-a	active	Specifies if scheduling is active or not. This sets the server attribute scheduling. If the option argument is "true" ("True", "t", "T", or "1"), the server is active and the PBS job scheduler will be called. If the argument is "false" ("False", "f", "F", or "0), the server is idle, and the scheduler will not be called and no jobs will be run. If this option is not specified, the server will retain the prior value of the scheduling attribute.
-A	acctfile	Specifies an absolute path name of the file to use as the accounting file. If not specified, the file name will be the current date in the `PBS_HOME/server_priv/accounting` directory.
-c	wait_for_ moms	This directs pbs_server to send the MOM hierarchy only to MOMs that request it for the first 10 minutes. After 10 minutes, it attempts to send the MOM hierarchy to MOMs that haven't requested it already. This greatly reduces traffic on start up.
-d	config_dir- ectory	Specifies the path of the directory which is home to the server's configuration files, PBS_HOME. A host may have multiple servers. Each server must have a different configuration directory. The default configuration directory is given by the symbol `$PBS_SERVER_HOME` which is typically `var/spool/torque`
-f	force over- write	Forces an overwrite of the server database. This can be useful to bypass the `yes/no` prompt when running something like pbs_server`-t create` and can ease installation and configuration of Torque via scripts.
-F	fork	Do not fork. This option is useful when running under systemd (Red Hat 7-based or SUSE 12-based systems).
--ha	high_avail- ability	Starts server in high availability mode (for details, see Server High Availability).
-H	hostname	Causes the server to start under a different hostname as obtained from gethostname (2). Useful for servers with multiple network interfaces to support connections from clients over an interface that has a hostname assigned that differs from the one that is returned by gethost name(2).
-l	location	Specifies where to find the scheduler (for example, Moab) when it does not reside on the same host as Torque. `pbs_server -l <other_host>:<other_port>`

Option	Name	Description
-L	logfile	Specifies an absolute path name of the file to use as the log file. If not specified, the file will be the current date in the `PBS_HOME/server_logs`directory (see the **-d** option).
-n	no send	This directs pbs_serverto not send the hierarchy to all the MOMs on startup. Instead, the hierarchy is only sent if a MOM requests it. This flag works only in conjunction with the local MOM hierarchy feature. See Setting Up the MOM Hierarchy (Optional) on page 47.
-p	port	Specifies the port number on which the server will listen for batch requests. If multiple servers are running on a single host, each must have its own unique port number. This option is for use in testing with multiple batch systems on a single host.
-S	scheduler_ port	Specifies the port number to which the server should connect when contacting the scheduler. The argument scheduler_conn is of the same syntax as under the -M option.
-t	type	If the job is rerunnable or restartable, and -t createis specified, the server will discard any existing configuration files, queues, and jobs, and initialize configuration files to the default values. The server is idled. If -t is not specified, the job states will remain the same.

Files

File	Description
TORQUE_HOME/server_ priv	Default directory for configuration files, typically `/usr/spool/pbs/server_ priv`
TORQUE_HOME/server_ logs	Directory for log files recorded by the server

Signal handling

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

Action	Description
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.

Action	Description
SIGINT	Causes an orderly shutdown of pbs_server.
SIGUSR1, SIGURS2	Causes server to increase and decrease logging levels, respectively.
SIGTERM	Causes an orderly shutdown of pbs_server.
SIGSHUTDN	On systems (Unicos) where SIGSHUTDN is defined, it also causes an orderly shutdown of the server.
SIGPIPE	This signal is ignored.

All other signals have their default behavior installed.

Exit status

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

Related Topics

pbs_mom(8B)

pbsnodes(8B)

qmgr(1B)

qrun(8B)

qsub(1B)

qterm(8B)

Non-Adaptive Computing topics

- pbs_connect(3B)
- pbs_sched_basl(8B)
- pbs_sched_tcl(8B)
- qdisable(8B)
- qenable(8B)
- qstart(8B)
- qstop(8B)
- PBS External Reference Specification

pbs_track

Starts a new process and informs pbs_mom to start tracking it.

Synopsis

```
pbs_track -j <JOBID> [-b] <executable> [args]
```

Description

The pbs_track command tells a pbs_mom daemon to monitor the lifecycle and resource usage of the process that it launches using exec(). The pbs_mom is told about this new process via the Task Manager API, using tm_adopt(). The process must also be associated with a job that already exists on the pbs_mom.

By default, pbs_track will send its PID to Torque via tm_adopt(). It will then perform an exec(), causing <executable> to run with the supplied arguments. pbs_track will not return until the launched process has completed because it becomes the launched process.

This command can be considered related to the **pbsdsh** command which uses the tm_spawn() API call. The pbsdsh command asks a pbs_mom to launch and track a new process on behalf of a job. When it is not desirable or possible for the pbs_mom to spawn processes for a job, pbs_track can be used to allow an external entity to launch a process and include it as part of a job.

This command improves integration with Torque and SGI's MPT MPI implementation.

Options

Option	Description
-j <JOBID>	Job ID the new process should be associated with.
-b	Instead of having pbs_track send its PID to Torque, it will fork() first, send the child PID to Torque, and then execute from the forked child. This essentially "backgrounds" pbs_track so that it will return after the new process is launched.

Operands

The pbs_track command accepts a path to a program/executable (<executable>) and, optionally, one or more arguments to pass to that program.

Exit status

Because the pbs_track command becomes a new process (if used without **-b**), its exit status will match that of the new process. If the **-b** option is used, the exit status will be zero if no errors occurred before launching the new process.

If pbs_track fails, whether due to a bad argument or other error, the exit status will be set to a non-zero value.

Related Topics

pbsdsh(1B)

Non-Adaptive Computing topics

- tm_spawn(3B)

pbsdsh

The pbsdsh command distributes tasks to nodes under pbs.

> Some limitations exist in the way that pbsdsh can be used. Please note the following situations are not currently supported:
>
> - Running multiple instances of pbsdsh concurrently within a single job.
> - Using the -o and -s options concurrently; although requesting these options together is permitted, only the output from the first node is displayed rather than output from every node in the chain.

Synopsis

```
pbsdsh [-c copies] [-o] [-s] [-u] [-v] program [args]
pbsdsh [-n node] [-o] [-s] [-u] [-v] program [args]
pbsdsh [-h nodename] [-o] [-v] program [args]
```

Description

Executes (spawns) a normal Unix program on one or more nodes under control of the Portable Batch System, PBS. Pbsdsh uses the Task Manager API (see tm_spawn(3)) to distribute the program on the allocated nodes.

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

Users will find the PBS_TASKNUM, PBS_NODENUM, and the PBS_VNODENUM environmental variables useful. They contain the TM task id, the node identifier, and the cpu (virtual node) identifier.

> Note that under particularly high workloads, the pbsdsh command may not function properly.

Options

Option	Name	Description
-c	copies	The program is spawned on the first Copies nodes allocated. This option is mutually exclusive with **-n**.
-h	hostname	The program is spawned on the node specified.
-n	node	The program is spawned on one node which is the n-th node allocated. This option is mutually exclusive with **-c**.
-o	---	Capture stdout of the spawned program. Normally stdout goes to the job's output.
-s	---	If this option is given, the program is run in turn on each node, one after the other.
-u	---	The program is run once on each node (unique). This ignores the number of allocated processors on a given node.
-v	---	Verbose output about error conditions and task exit status is produced.

Operands

The first operand, program, is the program to execute.

Additional operands are passed as arguments to the program.

Standard error

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

Exit status

Upon successful processing of all the operands presented to the command, the exit status will be a value of zero.

If the pbsdsh command fails to process any operand, or fails to contact the MOM daemon on the localhost the command exits with a value greater than zero.

Related Topics

qsub(1B)

Non-Adaptive Computing topics

- tm_spawn(3B)

pbsnodes

PBS node manipulation.

Synopsis

```
pbsnodes [-{a|x}] [-q] [-s server] [node|:property]
pbsnodes -l [-q] [-s server] [state] [nodename|:property ...]
pbsnodes -m <running|standby|suspend|hibernate|shutdown> < host
list >
pbsnodes [-{c|d|o|r}] [-q] [-s server] [-n -l] [-N "note"] [-A
"append note"] [node|:property]
```

Description

The pbsnodes command is used to mark nodes down, free or offline. It can also be used to list nodes and their state. Node information is obtained by sending a request to the PBS job server. Sets of nodes can be operated on at once by specifying a node property prefixed by a colon. (For more information, see Node States.)

Nodes do not exist in a single state, but actually have a set of states. For example, a node can be simultaneously "busy" and "offline". The "free" state is the absence of all other states and so is never combined with other states.

In order to execute pbsnodes with other than the **-a** or **-l** options, the user must have PBS Manager or Operator privilege.

NUMA-Awareness

When Torque is configured with NUMA-awareness and configured with --enable-groups, the number of total *and* the number of available sockets, numachips (numa nodes), cores, and threads are returned when the status of nodes are queried by Moab (a call is made to pbsnodes).

See pbsnodes with NUMA-Awareness on page 198 for additional information and examples.

Options

Option	Description
-A	Append a note attribute to existing note attributes. The -N note option will overwrite exiting note attributes. -A will append a new note attribute to the existing note attributes delimited by a ',' and a space.
-a	All attributes of a node or all nodes are listed. This is the default if no flag is given.

Option	Description
-c	Clear OFFLINE from listed nodes.
-d	Print MOM diagnosis on the listed nodes. Not yet implemented. Use **momctl** instead.
-m	Set the hosts in the specified host list to the requested power state. If a compute node does not support the energy-saving power state you request, the command returns an error and leaves the state unchanged. In order for the command to wake a node from a low-power state, Wake-on-LAN (WOL) must be enabled for the node. In order for the command to wake a node from a low-power state, Wake-on-LAN must be enabled for the node and it must support the g WOL packet. For more information, see Changing Node Power States. The allowable power states are: ı **Running**: The node is up and running. ı **Standby**: CPU is halted but still powered. Moderate power savings but low latency entering and leaving this state. ı **Suspend**: Also known as Suspend-to-RAM. Machine state is saved to RAM. RAM is put into self-refresh mode. Much more significant power savings with longer latency entering and leaving state. ı **Hibernate**: Also known as Suspend-to-disk. Machine state is saved to disk and then powered down. Significant power savings but very long latency entering and leaving state. ı **Shutdown**: Equivalent to `shutdown now` command as root. The host list is a space-delimited list of node host names. See Examples on page 251.
-o	Add the OFFLINE state. This is different from being marked DOWN. OFFLINE prevents new jobs from running on the specified nodes. This gives the administrator a tool to hold a node out of service without changing anything else. The OFFLINE state will never be set or cleared automatically by pbs_server; it is purely for the manager or operator.
-p	Purge the node record from pbs_server. Not yet implemented.
-r	Reset the listed nodes by clearing OFFLINE and adding DOWN state. pbs_server will ping the node and, if they communicate correctly, free the node.

Option	Description
-l	List node names and their state. If no state is specified, only nodes in the DOWN, OFFLINE, or UNKNOWN states are listed. Specifying a state string acts as an output filter. Valid state strings are "active", "all", "busy", "down", "free", "job-exclusive", "job-sharing", "offline", "reserve", "state-unknown", "time-shared", and "up". - Using *all* displays all nodes and their attributes. - Using *active* displays all nodes which are job-exclusive, job-sharing, or busy. - Using *up* displays all nodes in an "up state". Up states include job-exclusive, job-sharing, reserve, free, busy and time-shared. - All other strings display the nodes which are currently in the state indicated by the string.
-N	Specify a "note" attribute. This allows an administrator to add an arbitrary annotation to the listed nodes. To clear a note, use `-N ""` or `-N n`.
-n	Show the "note" attribute for nodes that are DOWN, OFFLINE, or UNKNOWN. This option requires -l.
-q	Suppress all error messages.
-s	Specify the PBS server's hostname or IP address.
-x	Same as -A, but the output has an XML-like format.

Examples

Example A-3: host list

```
pbsnodes -m shutdown node01 node02 node03 node04
```

With this command, pbs_server tells the pbs_mom associated with nodes01-04 to shut down the node.

The pbsnodes output shows the current power state of nodes. In this example, note that pbsnodes returns the MAC addresses of the nodes.

```
pbsnodes
nuc1
    state = free
    power_state = Running
    np = 4
    ntype = cluster
    status = rectime=1395765676,macaddr=0b:25:22:92:7b:26
,cpuclock=Fixed,varattr=,jobs=,state=free,netload=1242652020,gres=,loadave=0.16,ncpus=
6,physmem=16435852kb,availmem=24709056kb,totmem=33211016kb,idletime=4636,nusers=3,nses
sions=12,sessions=2758 998 1469 2708 2797 2845 2881 2946 4087 4154 4373
6385,uname=Linux bdaw 3.2.0-60-generic #91-Ubuntu SMP Wed Feb 19 03:54:44 UTC 2014
x86_64,opsys=linux
    note = This is a node note
    mom_service_port = 15002
    mom_manager_port = 15003

nuc2
    state = free
    power_state = Running
    np = 4
    ntype = cluster
    status = rectime=1395765678,macaddr=2c:a8:6b:f4:b9:35
,cpuclock=OnDemand:800MHz,varattr=,jobs=,state=free,netload=12082362,gres=,loadave=0.0
0,ncpus=4,physmem=16300576kb,availmem=17561808kb,totmem=17861144kb,idletime=67538,nuse
rs=2,nsessions=7,sessions=2189 2193 2194 2220 2222 2248 2351,uname=Linux nuc2 2.6.32-
431.el6.x86_64 #1 SMP Fri Nov 22 03:15:09 UTC 2013 x86_64,opsys=linux
    mom_service_port = 15002
    mom_manager_port = 15003
```

Related Topics

pbs_server(8B)

Non-Adaptive Computing topics

- PBS External Reference Specification

qalter

Alter batch job.

Synopsis

```
qalter [-a date_time][-A account_string][-c interval][-e path_
name]
 [-h hold_list][-j join_list][-k keep_list][-l resource_list]
 [-m mail_options][-M mail_list][-n][-N name][-o path_name]
 [-p priority][-r y|n][-S path_name_list][-u user_list]
 [-v variable_list][-W additional_attributes]
 [-t array_range]
 job_identifier ...
```

Description

The qalter command modifies the attributes of the job or jobs specified by `job_identifier` on the command line. Only those attributes listed as options on the command will be modified. If any of the specified attributes cannot be modified for a job for any reason, none of that job's attributes will be modified.

The qalter command accomplishes the modifications by sending a Modify Job batch request to the batch server which owns each job.

Options

Option	Name	Description
-a	date_time	Replaces the time at which the job becomes eligible for execution. The date_time argument syntax is: `[[[[CC]YY]MM]DD]hhmm[.SS]` If the month, MM, is not specified, it will default to the current month if the specified day DD, is in the future. Otherwise, the month will be set to next month. Likewise, if the day, DD, is not specified, it will default to today if the time hhmm is in the future. Otherwise, the day will be set to tomorrow. This attribute can be altered once the job has begun execution, but it will not take effect unless the job is rerun.
-A	account_ string	Replaces the account string associated with the job. This attribute cannot be altered once the job has begun execution.
-c	checkpoint_ interval	Replaces the interval at which the job will be checkpointed. If the job executes upon a host which does not support checkpointing, this option will be ignored. The interval argument is specified as: - *n* – No checkpointing is to be performed. - *s* – Checkpointing is to be performed only when the server executing the job is shutdown. - *c* – Checkpointing is to be performed at the default minimum cpu time for the queue from which the job is executing. - *c=minutes* – Checkpointing is performed at intervals of the specified amount of time in minutes. Minutes are the number of minutes of CPU time used, not necessarily clock time. This value must be greater than zero. If the number is less than the default checkpoint time, the default time will be used. This attribute can be altered once the job has begun execution, but the new value does not take effect unless the job is rerun.

Option	Name	Description
-e	path_name	Replaces the path to be used for the standard error stream of the batch job. The path argument is of the form: `[hostname:]path_name` where *hostname* is the name of a host to which the file will be returned and *path_name* is the path name on that host in the syntax recognized by POSIX 1003.1. The argument will be interpreted as follows: • *path_name* – Where *path_name* is not an absolute path name, then the qalter command will expand the path name relative to the current working directory of the command. The command will supply the name of the host upon which it is executing for the hostname component. • *hostname:path_name* – Where *path_name* is not an absolute path name, then the qalter command will not expand the path name. The execution server will expand it relative to the home directory of the user on the system specified by hostname. This attribute can be altered once the job has begun execution, but it will not take effect unless the job is rerun.
-h	hold_list	Updates the types of holds on the job. The hold_list argument is a string of one or more of the following characters: • *u* – Add the USER type hold. • *s* – Add the SYSTEM type hold if the user has the appropriate level of privilege. (Typically reserved to the batch administrator.) • *o* – Add the OTHER (or OPERATOR) type hold if the user has the appropriate level of privilege. (Typically reserved to the batch administrator and batch operator.) • *n* – Set to none and clear the hold types which could be applied with the user's level of privilege. Repetition of characters is permitted, but "n" may not appear in the same option argument with the other three characters. This attribute can be altered once the job has begun execution, but the hold will not take effect unless the job is rerun.
-j	join	Declares which standard streams of the job will be merged together. The join argument value may be the characters "oe" orf"eo", or the single character "n". An argument value of oe directs that the standard output and standard error streams of the job will be merged, intermixed, and returned as the standard output. An argument value of eo directs that the standard output and standard error streams of the job will be merged, intermixed, and returned as the standard error. A value of n directs that the two streams will be two separate files. This attribute can be altered once the job has begun execution, but it will not take effect unless the job is rerun. If using either the -e or the -o option and the `-j eo|oe` option, the `-j` option takes precedence and all standard error and output messages go to the chosen output file.

Option	Name	Description
-k	keep	Defines which if either of standard output or standard error of the job will be retained on the execution host. If set for a stream, this option overrides the path name for that stream. The argument is either the single letter "e", "o", or "n", or one or more of the letters "e" and "o" combined in either order. • *n* – No streams are to be retained. • *e* – The standard error stream is to retained on the execution host. The stream will be placed in the home directory of the user under whose user id the job executed. The file name will be the default file name given by: `job_name.esequence` where job_name is the name specified for the job, and sequence is the sequence number component of the job identifier. • *o* – The standard output stream is to be retained on the execution host. The stream will be placed in the home directory of the user under whose user id the job executed. The file name will be the default file name given by: `job_name.osequence` where job_name is the name specified for the job, and sequence is the sequence number component of the job identifier. • *eo* – Both the standard output and standard error streams will be retained. • *oe* – Both the standard output and standard error streams will be retained. This attribute cannot be altered once the job has begun execution.
-l	resource_ list	Modifies the list of resources that are required by the job. The resource_list argument is in the following syntax: `resource_name[=[value]][,resource_name[=[value]],...]` For the complete list of resources that can be modified, see Requesting Resources. If a requested modification to a resource would exceed the resource limits for jobs in the current queue, the server will reject the request. If the job is running, only certain resources can be altered. Which resources can be altered in the run state is system dependent. A user may only lower the limit for those resources.
-m	mail_ options	Replaces the set of conditions under which the execution server will send a mail message about the job. The mail_options argument is a string which consists of the single character "n", or one or more of the characters "a", "b", and "e". If the character "n" is specified, no mail will be sent. For the letters "a", "b", and "e": • *a* – Mail is sent when the job is aborted by the batch system. • *b* – Mail is sent when the job begins execution. • *e* – Mail is sent when the job ends.

Option	Name	Description
-M	user_list	Replaces the list of users to whom mail is sent by the execution server when it sends mail about the job. The user_list argument is of the form: `user[@host][,user[@host],...]`

-n	node-exclusive	Sets or unsets exclusive node allocation on a job. Use the y and n options to enable or disable the feature. This affects only cpusets and compatible schedulers.
		`> qalter ... -n y #enables exclusive node allocation on a job`
		`> qalter ... -n n #disables exclusive node allocation on a job`

-N	name	Renames the job. The name specified may be up to and including 15 characters in length. It must consist of printable, nonwhite space characters with the first character alphabetic.
-o	path	Replaces the path to be used for the standard output stream of the batch job. The path argument is of the form: `[hostname:]path_name` where *hostname* is the name of a host to which the file will be returned and *path_name* is the path name on that host in the syntax recognized by POSIX. The argument will be interpreted as follows: • *path_name* – Where *path_name* is not an absolute path name, then the `qalter` command will expand the path name relative to the current working directory of the command. The command will supply the name of the host upon which it is executing for the hostname component. • *hostname:path_name* – Where *path_name* is not an absolute path name, then the `qalter`command will not expand the path name. The execution server will expand it relative to the home directory of the user on the system specified by hostname. This attribute can be altered once the job has begun execution, but it will not take effect unless the job is rerun.
-p	priority	Replaces the priority of the job. The priority argument must be an integer between -1024 and +1023 inclusive. This attribute can be altered once the job has begun execution, but it will not take effect unless the job is rerun.
-r	[y/n]	Declares whether the job is rerunable (see the **qrerun** command). The option argument c is a single character. PBS recognizes the following characters: y and n. If the argument is "y", the job is marked rerunable. If the argument is "n", the job is marked as not rerunable.

Option	Name	Description
-S	path	Declares the shell that interprets the job script. The option argument path_list is in the form: `path[@host][,path[@host],...]` Only one path may be specified for any host named. Only one path may be specified without the corresponding host name. The path selected will be the one with the host name that matched the name of the execution host. If no matching host is found, then the path specified (without a host) will be selected. If the `-S` option is not specified, the option argument is the null string, or no entry from the path_list is selected, the execution will use the login shell of the user on the execution host. This attribute can be altered once the job has begun execution, but it will not take effect unless the job is rerun.
-t	array_ range	The array_range argument is an integer id or a range of integers. Multiple ids or id ranges can be combined in a comma delimited list. Examples: `-t 1-100` or `-t 1,10,50-100` If an array range isn't specified, the command tries to operate on the entire array. The command acts on the array (or specified range of the array) just as it would on an individual job. An optional "slot limit" can be specified to limit the amount of jobs that can run concurrently in the job array. The default value is unlimited. The slot limit must be the last thing specified in the array_request and is delimited from the array by a percent sign (%). `qalter 15.napali[] -t %20` Here, the array 15.napali[] is configured to allow a maximum of 20 concurrently running jobs. Slot limits can be applied at job submit time with **qsub**, or can be set in a global server parameter policy with **max_slot_limit**.
-u	user_list	Replaces the user name under which the job is to run on the execution system. The user_list argument is of the form: `user[@host][,user[@host],...]` Only one user name may be given for per specified host. Only one of the user specifications may be supplied without the corresponding host specification. That user name will be used for execution on any host not named in the argument list. This attribute cannot be altered once the job has begun execution.
-W	additional_ attributes	The `-W` option allows for the modification of additional job attributes. Note if white space occurs anywhere within the option argument string or the equal sign, "=", occurs within an attribute_value string, then the string must be enclosed with either single or double quote marks. To see the attributes PBS currently supports within the `-W` option, see -W additional_ attributes.

-W additional_attributes

The following table lists the attributes PBS currently supports with the -W option.

Attribute	Description
depend=dependency_ list	Redefines the dependencies between this and other jobs. The dependency_list is in the form: `type[:argument[:argument...][,type:argument...]` The argument is either a numeric count or a PBS job id according to type. If argument is a count, it must be greater than 0. If it is a job id and is not fully specified in the form: `seq_number.server.name`, it will be expanded according to the default server rules. If argument is null (the preceding colon need not be specified), the dependency of the corresponding type is cleared (unset). - *synccount:count* – This job is the first in a set of jobs to be executed at the same time. Count is the number of additional jobs in the set. - *syncwith:jobid* – This job is an additional member of a set of jobs to be executed at the same time. In the above and following dependency types, jobid is the job identifier of the first job in the set. - *after:jobid [:jobid...]* – This job may be scheduled for execution at any point after jobs jobid have started execution. - *afterok:jobid [:jobid...]* – This job may be scheduled for execution only after jobs jobid have terminated with no errors. See the csh warning under "Extended Description". - *afternotok:jobid [:jobid...]* – This job may be scheduled for execution only after jobs jobid have terminated with errors. See the csh warning under "Extended Description". - *afterany:jobid [:jobid...]* – This job may be scheduled for execution after jobs jobid have terminated, with or without errors. - *on:count* – This job may be scheduled for execution after count dependencies on other jobs have been satisfied. This dependency is used in conjunction with any of the 'before' dependencies shown below. If job A has on:2, it will wait for two jobs with 'before' dependencies on job A to be fulfilled before running. - *before:jobid [:jobid...]* – When this job has begun execution, then jobs jobid... may begin. - *beforeok:jobid [:jobid...]* – If this job terminates execution without errors, then jobs jobid... may begin. See the csh warning under "Extended Description". - *beforenotok:jobid [:jobid...]* – If this job terminates execution with errors, then jobs jobid... may begin. See the csh warning under "Extended Description". - *beforeany:jobid [:jobid...]* – When this job terminates execution, jobs jobid... may begin. If any of the before forms are used, the job referenced by jobid must have been submitted with a dependency type of on. If any of the before forms are used, the jobs referenced by jobid must have the same owner as the job being altered. Otherwise, the dependency will not take effect. Error processing of the existence, state, or condition of the job specified to qalter is a deferred service, i.e. the check is performed after the job is queued. If an error is detected, the job will be deleted by the server. Mail will be sent to the job submitter stating the error.

Attribute	Description
group_list=g_list	Alters the group name under which the job is to run on the execution system. The g_list argument is of the form: `group[@host][,group[@host],...]` Only one group name may be given per specified host. Only one of the group specifications may be supplied without the corresponding host specification. That group name will used for execution on any host not named in the argument list.
stagein=file_list **stageout=file_list**	Alters which files are staged (copied) in before job start or staged out after the job completes execution. The file_list is in the form: `local_file@hostname:remote_file[,...]` The name local_file is the name on the system where the job executes. It may be an absolute path or a path relative to the home directory of the user. The name remote_file is the destination name on the host specified by hostname. The name may be absolute or relative to the user's home directory on the destination host.

Operands

The qalter command accepts one or more job_identifier operands of the form:

```
sequence_number[.server_name][@server]
```

Standard error

Any error condition, either in processing the options or the operands, or any error received in reply to the batch requests will result in an error message being written to standard error.

Exit status

Upon successful processing of all the operands presented to the qalter command, the exit status will be a value of zero.

If the qalter command fails to process any operand, the command exits with a value greater than zero.

Copyright

Standard can be obtained online at
http://www.opengroup.org/unix/online.html.

Related Topics

- qdel
- qhold
- qrls
- qsub

Non-Adaptive Computing topics

- Batch Environment Services
- qmove
- touch

qchkpt

Checkpoint pbs batch jobs.

Synopsis

```
qchkpt <JOBID>[ <JOBID>] ...
```

Description

The qchkpt command requests that the PBS MOM generate a checkpoint file for a running job.

This is an extension to POSIX.2d.

The qchkpt command sends a Chkpt Job batch request to the server as described in the general section.

Options

None.

Operands

The qchkpt command accepts one or more job_identifier operands of the form:

```
sequence_number[.server_name][@server]
```

Examples

```
$ # request a checkpoint for job 3233
$ qchkpt 3233
```

Standard error

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

Exit status

Upon successful processing of all the operands presented to the qchkpt command, the exit status will be a value of zero.

If the qchkpt command fails to process any operand, the command exits with a value greater than zero.

Related Topics

qhold(1B)

qrls(1B)

qalter(1B)

qsub(1B)

Non-Adaptive Computing topics

- pbs_alterjob(3B)
- pbs_holdjob(3B),
- pbs_rlsjob(3B)
- pbs_job_attributes(7B)
- pbs_resources_unicos8(7B)

qdel

(delete job)

Synopsis

```
qdel [{-a <asynchronous delete>|-b <secs>|-m <message>|-p
<purge>|-t <array_range>|-W <delay>}]
<JOBID>[ <JOBID>]... | 'all' | 'ALL'
```

Description

The qdel command deletes jobs in the order in which their job identifiers are presented to the command. A job is deleted by sending a Delete Job batch request to the batch server that owns the job. A job that has been deleted is no longer subject to management by batch services.

A batch job may be deleted by its owner, the batch operator, or the batch administrator.

A batch job being deleted by a server will be sent a SIGTERM signal following by a SIGKILL signal. The time delay between the two signals is an attribute of the execution queue from which the job was run (set table by the administrator). This delay may be overridden by the **-W** option.

See the PBS ERS section 3.1.3.3, "Delete Job Request", for more information.

Options

Option	Name	Description
-a	asynchronous delete	Performs an asynchronous delete. The server responds to the user before contacting the MOM. The option `qdel -a all` performs `qdel all` due to restrictions from being single-threaded.
-b	seconds	Defines the maximum number of seconds qdel will block attempting to contact pbs_server. If pbs_server is down, or for a variety of communication failures, qdel will continually retry connecting to pbs_server for job submission. This value overrides the CLIENTRETRY parameter in `torque.cfg`. This is a non-portable Torque extension. Portability-minded users can use the PBS_CLIENTRETRY environmental variable. A negative value is interpreted as infinity. The default is 0.
-p	purge	Forcibly purges the job from the server. This should only be used if a running job will not exit because its allocated nodes are unreachable. The admin should make every attempt at resolving the problem on the nodes. If a job's mother superior recovers after purging the job, any epilogue scripts may still run. This option is only available to a batch operator or the batch administrator.
-t	array_range	The array_range argument is an integer id or a range of integers. Multiple ids or id ranges can be combined in a comma delimited list (examples: -t 1-100 or -t 1,10,50-100). The command acts on the array (or specified range of the array) just as it would on an individual job. When deleting a range of jobs, you must include the subscript notation after the job ID (for example, "qdel -t 1-3 98432[]").
-m	message	Specify a comment to be included in the email. The argument message specifies the comment to send. This option is only available to a batch operator or the batch administrator.
-W	delay	Specifies the wait delay between the sending of the SIGTERM and SIGKILL signals. The argument is the length of time in seconds of the delay.

Operands

The qdel command accepts one or more job_identifier operands of the form:

`sequence_number[.server_name][@server]`

or

`all`

Examples

```
# Delete a job array
$ qdel 1234[]

# Delete *one* job from an array
$ qdel 1234[1]

# Delete all jobs, including job arrays
$ qdel all

# Delete selected jobs from an array
$ qdel -t 2-4,6,8-10 64[]
```

There is not an option that allows you to delete all job arrays without deleting jobs.

Standard error

The qdel command will write a diagnostic messages to standard error for each error occurrence.

Exit status

Upon successful processing of all the operands presented to the qdel command, the exit status will be a value of zero.

If the qdel command fails to process any operand, the command exits with a value greater than zero.

Related Topics

qsub(1B)

qsig(1B)

Non-Adaptive Computing topics

- pbs_deljob(3B)

qgpumode

This command is deprecated, use the nvidia-smi utility instead. See https://developer.nvidia.com/nvidia-system-management-interface and http://developer.download.nvidia.com/compute/cuda/6_0/rel/gdk/nvidia-smi.331.38.pdf for more information.

(GPU mode)

Synopsis

```
qgpumode -H host -g gpuid -m mode
```

Description

The qgpumode command specifies the mode for the GPU. This command triggers an immediate update of the pbs_server.

For additional information about options for configuring GPUs, see NVIDIA GPUs in the Moab Workload Manager Administrator Guide.

Options

Option	Description
-H	Specifies the host where the GPU is located.
-g	Specifies the ID of the GPU. This varies depending on the version of the Nvidia driver used. For driver 260.x, it is 0, 1, and so on. For driver 270.x, it is the PCI bus address, i.e., 0:5:0.

Option	Description

-m	Specifies the new mode for the GPU:
	• **0 (Default/Shared)**: Default/shared compute mode. Multiple threads can use `cudaSetDevice()` with this device.
	• **1 (Exclusive Thread)**: Compute-exclusive-thread mode. Only one thread in one process is able to use `cudaSetDevice()` with this device.
	• **2 (Prohibited)**: Compute-prohibited mode. No threads can use `cudaSetDevice()` with this device.
	• **3 (Exclusive Process)**: Compute-exclusive-process mode. Many threads in one process are able to use `cudaSetDevice()` with this device.
	`qgpumode -H node01 -g 0 -m 1`
	This puts the first GPU on node01 into mode 1 (exclusive)
	`qgpumode -H node01 -g 0 -m 0`
	This puts the first GPU on node01 into mode 0 (shared)

Related Topics

qgpureset

qgpureset

(reset GPU)

Synopsis

```
qgpureset -H host -g gpuid -p -v
```

Description

The qgpureset command resets the GPU.

Options

Option	Description
-H	Specifies the host where the GPU is located.

Option	Description
-g	Specifies the ID of the GPU. This varies depending on the version of the Nvidia driver used. For driver 260.x, it is 0, 1, and so on. For driver 270.x, it is the PCI bus address, i.e., 0:5:0.
-p	Specifies to reset the GPU's permanent ECC error count.
-v	Specifies to reset the GPU's volatile ECC error count.

Related Topics

qgpumode

qhold

(hold job)

Synopsis

```
qhold [{-h <HOLD LIST>|-t <array_range>}] <JOBID>[ <JOBID>]
...
```

Description

The qhold command requests that the server place one or more holds on a job. A job that has a hold is not eligible for execution. There are three supported holds: USER, OTHER (also known as operator), and SYSTEM.

A user may place a USER hold upon any job the user owns. An "operator", who is a user with "operator privilege," may place ether an USER or an OTHER hold on any job. The batch administrator may place any hold on any job.

If no **-h** option is given, the USER hold will be applied to the jobs described by the job_identifier operand list.

If the job identified by job_identifier is in the queued, held, or waiting states, then the hold type is added to the job. The job is then placed into held state if it resides in an execution queue.

If the job is in running state, then the following additional action is taken to interrupt the execution of the job. If checkpoint/restart is supported by the host system, requesting a hold on a running job will (1) cause the job to be checkpointed, (2) the resources assigned to the job will be released, and (3) the job is placed in the held state in the execution queue.

If checkpoint/restart is not supported, qhold will only set the requested hold attribute. This will have no effect unless the job is rerun with the **qrerun** command.

Options

Option	Name	Description
-h	hold_list	The hold_list argument is a string consisting of one or more of the letters "u", "o", or "s" in any combination. The hold type associated with each letter is: • *u* – USER • *o* – OTHER • *s* – SYSTEM
-t	array_range	The array_range argument is an integer id or a range of integers. Multiple ids or id ranges can be combined in a comma delimited list (examples: -t 1-100 or -t 1,10,50-100) . If an array range isn't specified, the command tries to operate on the entire array. The command acts on the array (or specified range of the array) just as it would on an individual job.

Operands

The qhold command accepts one or more job_identifier operands of the form:

`sequence_number[.server_name][@server]`

Example

```
> qhold -h u 3233 place user hold on job 3233
```

Standard error

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

Exit status

Upon successful processing of all the operands presented to the qhold command, the exit status will be a value of zero.

If the qhold command fails to process any operand, the command exits with a value greater than zero.

Related Topics

qrls(1B)

qalter(1B)

qsub(1B)

Non-Adaptive Computing topics

- pbs_alterjob(3B)
- pbs_holdjob(3B)
- pbs_rlsjob(3B)
- pbs_job_attributes(7B)
- pbs_resources_unicos8(7B)

qmgr

(PBS Queue Manager) PBS batch system manager.

Synopsis

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

Description

The qmgr command provides an administrator interface to query and configure batch system parameters (see Server Parameters).

The command reads directives from standard input. The syntax of each directive is checked and the appropriate request is sent to the batch server or servers.

The list or print subcommands of qmgr can be executed by general users. Creating or deleting a queue requires PBS Manager privilege. Setting or unsetting server or queue attributes requires PBS Operator or Manager privilege.

> By default, the user root is the only PBS Operator and Manager. To allow other users to be privileged, the server attributes operators and managers will need to be set (i.e., as root, issue `qmgr -c 'set server managers += <USER1>@<HOST> '`). See Torque/PBS Integration Guide - RM Access Control in the *Moab Workload Manager Administrator Guide*.

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

Commands can be abbreviated to their minimum unambiguous form. A command is terminated by a new line character or a semicolon, ";", character. Multiple commands may be entered on a single line. A command may extend across lines by escaping the new line character with a back-slash "\".

Comments begin with the "#" character and continue to end of the line. Comments and blank lines are ignored by qmgr.

Options

Option	Name	Description
-a	---	Abort qmgr on any syntax errors or any requests rejected by a server.
-c	command	Execute a single command and exit qmgr.
-e	---	Echo all commands to standard output.
-n	---	No commands are executed, syntax checking only is performed.
-z	---	No errors are written to standard error.

Operands

The *server* operands identify the name of the batch server to which the administrator requests are sent. Each *server* conforms to the following syntax:

```
host_name[:port]
```

where *host_name* is the network name of the host on which the server is running and *port* is the port number to which to connect. If *port* is not specified, the default port number is used.

If *server* is not specified, the administrator requests are sent to the local server.

Standard input

The qmgr command reads standard input for directives until end of file is reached, or the exit or quit directive is read.

Standard output

If Standard Output is connected to a terminal, a command prompt will be written to standard output when qmgr is ready to read a directive.

If the **-e** option is specified, qmgr will echo the directives read from standard input to standard output.

Standard error

If the **-z** option is not specified, the qmgr command will write a diagnostic message to standard error for each error occurrence.

Directive syntax

A qmgr directive is one of the following forms:

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

where *command* is the command to perform on an object.

Commands are:

Command	Description
active	Sets the active objects. If the active objects are specified, and the name is not given in a qmgr cmd the active object names will be used.
create	Is to create a new object, applies to queues and nodes.
delete	Is to destroy an existing object, applies to queues and nodes.
set	Is to define or alter attribute values of the object.
unset	Is to clear the value of attributes of the object. This form does not accept an OP and value, only the attribute name.
list	Is to list the current attributes and associated values of the object.
print	Is to print all the queue and server attributes in a format that will be usable as input to the qmgr command.
names	Is a list of one or more names of specific objects The name list is in the form: `[name][@server][,queue_name[@server]...]` with no intervening white space. The name of an object is declared when the object is first created. If the name is @server, then all the objects of specified type at the server will be affected.
attr	Specifies the name of an attribute of the object which is to be set or modified. If the attribute is one which consist of a set of resources, then the attribute is specified in the form: `attribute_name.resource_name`

Command	Description
OP	Operation to be performed with the attribute and its value: • "=" – set the value of the attribute. If the attribute has an existing value, the current value is replaced with the new value. • "+=" – increase the current value of the attribute by the amount in the new value. • "-=" – decrease the current value of the attribute by the amount in the new value.
value	The value to assign to an attribute. If the value includes white space, commas or other special characters, such as the "#" character, the value string must be enclosed in quote marks (").

The following are examples of qmgr directives:

```
create queue fast priority=10,queue_type=e,enabled = true,max_running=0
set queue fast max_running +=2
create queue little
set queue little resources_max.mem=8mw,resources_max.cput=10
unset queue fast max_running
set node state = "down,offline"
active server s1,s2,s3
list queue @server1
set queue max_running = 10       - uses active queues
```

Exit status

Upon successful processing of all the operands presented to the qmgr command, the exit status will be a value of zero.

If the qmgr command fails to process any operand, the command exits with a value greater than zero.

Related Topics

pbs_server(8B)

Non-Adaptive Computing topics

- pbs_queue_attributes (7B)
- pbs_server_attributes (7B)
- qstart (8B), qstop (8B)
- qenable (8B), qdisable (8)
- PBS External Reference Specification

qmove

Move PBS batch jobs.

Synopsis

```
qmove destination jobId [jobId ...]
```

Description

To move a job is to remove the job from the queue in which it resides and instantiate the job in another queue. The qmove command issues a Move Job batch request to the batch server that currently owns each job specified by *jobId*.

A job in the **Running**, **Transiting**, or **Exiting** state cannot be moved.

Operands

The first operand, the new *destination*, is one of the following:

`queue`

`@server`

`queue@server`

If the *destination* operand describes only a queue, then qmove will move jobs into the queue of the specified name at the job's current server. If the *destination* operand describes only a batch server, then qmove will move jobs into the default queue at that batch server. If the *destination* operand describes both a queue and a batch server, then qmove will move the jobs into the specified queue at the specified server.

All following operands are *jobId*s which specify the jobs to be moved to the new *destination*. The qmove command accepts one or more *jobId* operands of the form: `sequenceNumber[.serverName][@server]`

Standard error

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

Exit status

Upon successful processing of all the operands presented to the qmove command, the exit status will be a value of zero.

If the qmove command fails to process any operand, the command exits with a value greater than zero.

Related Topics

qsub

Related Topics(non-Adaptive Computing topics)

- pbs_movejob(3B)

qorder

Exchange order of two PBS batch jobs in any queue.

Synopsis

```
qorder job1_identifier job2_identifier
```

Description

To order two jobs is to exchange the jobs' positions in the queue(s) in which the jobs reside. The two jobs must be located on the same server. No attribute of the job, such as priority, is changed. The impact of changing the order in the queue(s) is dependent on local job schedule policy. For information about your local job schedule policy, contact your systems administrator.

A job in the **running** state cannot be reordered.

Operands

Both operands are `job_identifier` s that specify the jobs to be exchanged. The qorder command accepts two `job_identifier` operands of the following form:

```
sequence_number[.server_name][@server]
```

The two jobs must be in the same location, so the server specification for the two jobs must agree.

Standard error

The qorder command will write diagnostic messages to standard error for each error occurrence.

Exit status

Upon successful processing of all the operands presented to the qorder command, the exit status will be a value of zero.

If the qorder command fails to process any operand, the command exits with a value greater than zero.

Related Topics

qsub

qmove

Related Topics(non-Adaptive Computing topics)

- pbs_orderjob(3B)
- pbs_movejob(3B)

qrerun

(Rerun a batch job)

Synopsis

```
qrerun [{-f}] <JOBID>[ <JOBID>] ...
```

Description

The qrerun command directs that the specified jobs are to be rerun if possible. To rerun a job is to terminate the session leader of the job and return the job to the queued state in the execution queue in which the job currently resides.

If a job is marked as not rerunable then the rerun request will fail for that job. If the mini-server running the job is down, or it rejects the request, the Rerun Job batch request will return a failure unless **-f** is used.

Using **-f** violates IEEE Batch Processing Services Standard and should be handled with great care. It should only be used under exceptional circumstances. The best practice is to fix the problem mini-server host and let qrerun run normally. The nodes may need manual cleaning (see the -r option on the **qsub** and **qalter** commands).

Options

Option	Description
-f	Force a rerun on a job

```
qrerun -f 15406
```

The qrerun all command is meant to be run if all of the compute nodes go down. If the machines have actually crashed, then we know that all of the jobs need to be restarted. The behavior if you don't run this would depend on how you bring up the pbs_mom daemons, but by default would be to cancel all of the jobs.

Running the command makes it so that all jobs are requeued without attempting to contact the moms on which they should be running.

Operands

The qrerun command accepts one or more job_identifier operands of the form:

`sequence_number[.server_name][@server]`

Standard error

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

Exit status

Upon successful processing of all the operands presented to the qrerun command, the exit status will be a value of zero.

If the qrerun command fails to process any operand, the command exits with a value greater than zero.

Examples

```
> qrerun 3233
```

(Job 3233 will be re-run.)

Related Topics

qsub(1B)

qalter(1B)

Non-Adaptive Computing topics

- pbs_alterjob(3B)
- pbs_rerunjob(3B)

qrls

(Release hold on PBS batch jobs)

Synopsis

```
qrls [{-h <HOLD LIST>|-t <array_range>}] <JOBID>[ <JOBID>] ...
```

Description

The qrls command removes or releases holds which exist on batch jobs.

A job may have one or more types of holds which make the job ineligible for execution. The types of holds are USER, OTHER, and SYSTEM. The different types of holds may require that the user issuing the qrls command have special privileges. A user may always remove a USER hold on their own jobs, but only privileged users can remove OTHER or SYSTEM holds. An attempt to release a hold for which the user does not have the correct privilege is an error and no holds will be released for that job.

If no **-h** option is specified, the USER hold will be released.

If the job has no execution_time pending, the job will change to the queued state. If an execution_time is still pending, the job will change to the waiting state.

If you run qrls on an array subjob, pbs_server will correct the slot limit holds for the array to which it belongs.

Options

Command	Name	Description
-h	hold_ list	Defines the types of hold to be released from the jobs. The hold_list option argument is a string consisting of one or more of the letters "u", "o", and "s" in any combination. The hold type associated with each letter is: - *u* – USER - *o* – OTHER - *s* – SYSTEM
-t	array_ range	The array_range argument is an integer id or a range of integers. Multiple ids or id ranges can be combined in a comma delimited list. Examples: -t 1-100 or -t 1,10,50-100 If an array range isn't specified, the command tries to operate on the entire array. The command acts on the array (or specified range of the array) just as it would on an individual job.

Operands

The qrls command accepts one or more job_identifier operands of the form:

```
sequence_number[.server_name][@server]
```

Examples

```
> qrls -h u 3233 release user hold on job 3233
```

Standard error

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

Exit status

Upon successful processing of all the operands presented to the qrls command, the exit status will be a value of zero.

If the qrls command fails to process any operand, the command exits with a value greater than zero.

Related Topics

qsub(1B)

qalter(1B)

qhold(1B)

Non-Adaptive Computing topics)

- pbs_alterjob(3B)
- pbs_holdjob(3B)
- pbs_rlsjob(3B)

qrun

(Run a batch job)

Synopsis

```
qrun [{-H <HOST>|-a}] <JOBID>[ <JOBID>] ...
```

Overview

The qrun command runs a job.

Format

-H	
Format	<STRING> Host Identifier
Default	---
Description	Specifies the host within the cluster on which the job(s) are to be run. The host argument is the name of a host that is a member of the cluster of hosts managed by the server. If the option is not specified, the server will select the "worst possible" host on which to execute the job.
Example	`qrun -H hostname 15406`

-a	
Format	---
Default	---
Description	Run the job(s) asynchronously.
Example	`qrun -a 15406`

Command details

The qrun command is used to force a batch server to initiate the execution of a batch job. The job is run regardless of scheduling position or resource requirements.

In order to execute qrun, the user must have PBS Operation or Manager privileges.

Examples

```
> qrun 3233
```

(Run job 3233.)

qsig

(Signal a job)

Synopsis

```
qsig [{-s <SIGNAL>}] <JOBID>[ <JOBID>] ...
 [-a]
```

Description

The qsig command requests that a signal be sent to executing batch jobs. The signal is sent to the session leader of the job. If the **-s** option is not specified, SIGTERM is sent. The request to signal a batch job will be rejected if:

- The user is not authorized to signal the job.
- The job is not in the running state.
- The requested signal is not supported by the system upon which the job is executing.

The qsig command sends a Signal Job batch request to the server which owns the job.

Options

Option	Name	Description
-s	signal	Declares which signal is sent to the job. The signal argument is either a signal name, e.g. SIGKILL, the signal name without the SIG prefix, e.g. KILL, or an unsigned signal number, e.g. 9. The signal name SIGNULL is allowed; the server will send the signal 0 to the job which will have no effect on the job, but will cause an obituary to be sent if the job is no longer executing. Not all signal names will be recognized by qsig If it doesn't recognize the signal name, try issuing the signal number instead. Two special signal names, "suspend" and "resume", are used to suspend and resume jobs. Cray systems use the Cray-specific suspend()/resume() calls. On non-Cray system, suspend causes a SIGTSTP to be sent to all processes in the job's top task, wait 5 seconds, and then send a SIGSTOP to all processes in all tasks on all nodes in the job. This differs from Torque 2.0.0 which did not have the ability to propagate signals to sister nodes. Resume sends a SIGCONT to all processes in all tasks on all nodes. When suspended, a job continues to occupy system resources but is not executing and is not charged for walltime. The job will be listed in the "S" state. Manager or operator privilege is required to suspend or resume a job. Interactive jobs may not resume properly because the top-level shell will background the suspended child process.
-a	asynchronously	Makes the command run asynchronously.

Operands

The qsig command accepts one or more job_identifier operands of the form:

```
sequence_number[.server_name][@server]
```

Examples

```
> qsig -s SIGKILL 3233    send a SIGKILL to job 3233
> qsig -s KILL 3233       send a SIGKILL to job 3233
> qsig -s 9 3233          send a SIGKILL to job 3233
```

Standard error

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

Exit status

Upon successful processing of all the operands presented to the qsig command, the exit status will be a value of zero.

If the qsig command fails to process any operand, the command exits with a value greater than zero.

Related Topics

qsub(1B)

Non-Adaptive Computing topics

- pbs_sigjob(3B)
- pbs_resources_*(7B) where * is system type
- PBS ERS

qstat

Show status of PBS batch jobs.

Synopsis

```
qstat [-c] [-C] [-f [-1]] [-W site_specific] [job_
identifier... | destination...] [time]
qstat [-a|-i|-r|-e] [-c] [-n [-1]] [-s] [-G|-M] [-R] [-u user_
list]
[job_identifier... | destination...]
qstat -Q [-f [-1]] [-c] [-W site_specific] [destination...]
qstat -q [-c] [-G|-M] [destination...]
qstat -B [-c] [-f [-1]][-W site_specific] [server_name...]
qstat -t [-c] [-C]
```

Description

The qstat command is used to request the status of jobs, queues, or a batch server. The requested status is written to standard out.

When requesting job status, synopsis format 1 or 2, qstat will output information about each job_identifier or all jobs at each destination. Jobs for which the user does not have status privilege are not displayed.

When requesting queue or server status, synopsis format 3 through 5, qstat will output information about each destination.

You can configure Torque with `CFLAGS='DTXT'` to change the alignment of text in qstat output. This noticeably improves `qstat -r` output.

Options

Option	Description
-1	In combination with -n, the -1 option puts all of the nodes on the same line as the job ID. In combination with -f, attributes are not folded to fit in a terminal window. This is intended to ease the parsing of the qstat output.
-a	All jobs are displayed in the alternative format (see Standard output). If the operand is a destination id, all jobs at that destination are displayed. If the operand is a job id, information about that job is displayed.
-B	Specifies that the request is for batch server status and that the operands are the names of servers.
-c	Completed jobs are not displayed in the output. If desired, you can set the `PBS_QSTAT_NO_COMPLETE` environment variable to cause all qstat requests to not show completed jobs by default.
-C	Specifies that Torque will provide only a condensed output (job name, resources used, queue, state, and job owner) for jobs that have not changed recently. See job_full_report_time on page 324. Jobs that have recently changed will continue to send a full output.
-e	If the operand is a job id or not specified, only jobs in executable queues are displayed. Setting the PBS_QSTAT_EXECONLY environment variable will also enable this option.
-f	Specifies that a full status display be written to standard out. The [time] value is the amount of walltime, in seconds, remaining for the job. [time] does not account for walltime multipliers.

Option	Description
-G	Show size information in giga-bytes.
-i	Job status is displayed in the alternative format. For a destination id operand, statuses for jobs at that destination which are not running are displayed. This includes jobs which are queued, held or waiting. If an operand is a job id, status for that job is displayed regardless of its state.
-1	In combination with **-n**, the -1 option puts all of the nodes on the same line as the job ID. In combination with **-f**, attributes are not folded to fit in a terminal window. This is intended to ease the parsing of the qstat output.
-M	Show size information, disk or memory in mega-words. A word is considered to be 8 bytes.
-n	In addition to the basic information, nodes allocated to a job are listed.
-q	Specifies that the request is for queue status which should be shown in the alternative format.
-Q	Specifies that the request is for queue status and that the operands are destination identifiers.
-r	If an operand is a job id, status for that job is displayed. For a destination id operand, statuses for jobs at that destination which are running are displayed; this includes jobs which are suspended. Note that if there is no walltime given for a job, then elapsed time does not display.
-R	In addition to other information, disk reservation information is shown. Not applicable to all systems.
-s	In addition to the basic information, any comment provided by the batch administrator or scheduler is shown.
-t	Normal qstat output displays a summary of the array instead of the entire array, job for job. `qstat -t` expands the output to display the entire array. Note that arrays are now named with brackets following the array name; for example: `dbeer@napali:~/dev/torque/array_changes$ echo sleep 20 | qsub -t 0-299 189[].napali` Individual jobs in the array are now also noted using square brackets instead of dashes; for example, here is part of the output of `qstat -t` for the preceding array: `189[299].napali STDIN[299] dbeer 0 Q batch`

Option	Description
-u	Job status is displayed in the alternative format. If an operand is a job id, status for that job is displayed. For a destination id operand, statuses for jobs at that destination which are owned by the user(s) listed in user_list are displayed. The syntax of the user_list is: `user_name[@host][,user_name[@host],...]` Host names may be wild carded on the left end, e.g. "*.nasa.gov". User_name without a "@host" is equivalent to "user_name@*", that is at any host.

Operands

If neither the **-q** nor the **-B** option is given, the operands on the qstat command must be either job identifiers or destinations identifiers.

If the operand is a job identifier, it must be in the following form:

```
sequence_number[.server_name][@server]
```

where *sequence_number.server_name* is the job identifier assigned at submittal time (see **qsub**). If the *.server_name* is omitted, the name of the default server will be used. If *@server* is supplied, the request will be for the job identifier currently at that Server.

If the operand is a destination identifier, it is one of the following three forms:

- queue
- @server
- queue@server

If queue is specified, the request is for status of all jobs in that queue at the default server. If the @server form is given, the request is for status of all jobs at that server. If a full destination identifier, queue@server, is given, the request is for status of all jobs in the named queue at the named server.

If the **-Q** option is given, the operands are destination identifiers as specified above. If queue is specified, the status of that queue at the default server will be given. If queue@server is specified, the status of the named queue at the named server will be given. If @server is specified, the status of all queues at the named server will be given. If no destination is specified, the status of all queues at the default server will be given.

If the **-B** option is given, the operand is the name of a server.

Standard output

Displaying job status

If job status is being displayed in the default format and the **-f** option is not specified, the following items are displayed on a single line, in the specified order, separated by white space:

- the job identifier assigned by PBS.
- the job name given by the submitter.
- the job owner.
- the CPU time used.
- the job state:

Item	Description
C	Job is completed after having run.
E	Job is exiting after having run.
H	Job is held.
Q	Job is queued, eligible to run or routed.
R	Job is running.
T	Job is being moved to new location.
W	Job is waiting for its execution time (-a option) to be reached.
S	(Unicos only) Job is suspended.

- the queue in which the job resides.

If job status is being displayed and the **-f** option is specified, the output will depend on whether qstat was compiled to use a Tcl interpreter. See Configuration for details. If Tcl is not being used, full display for each job consists of the header line:

```
Job Id: job identifier
```

Followed by one line per job attribute of the form:

```
attribute_name = value
```

If any of the options **-a**, **-i**, **-r**, **-u**, **-n**, **-s**, **-G**, or **-M** are provided, the alternative display format for jobs is used. The following items are displayed on a single line, in the specified order, separated by white space:

- the job identifier assigned by PBS
- the job owner
- the queue in which the job currently resides

- the job name given by the submitter
- the session id (if the job is running)
- the number of nodes requested by the job
- the number of cpus or tasks requested by the job
- the amount of memory requested by the job
- either the cpu time, if specified, or wall time requested by the job, (hh:mm)
- the jobs current state
- the amount of cpu time or wall time used by the job (hh:mm)

When any of the above options or the **-r** option is used to request an alternative display format, a column with the requested memory for the job is displayed. If more than one type of memory is requested for the job, either through server or queue parameters or command line, only one value can be displayed. The value displayed depends on the order the memory types are evaluated with the last type evaluated being the value displayed. The order of evaluation is dmem, mem, pmem, pvmem, vmem.

If the **-r** option is provided, the line contains:

- the job identifier assigned by PBS
- the job owner
- the queue in which the job currently resides
- the number of nodes requested by the job
- the number of cpus or tasks requested by the job
- the amount of memory requested by the job
- either the cpu time or wall time requested by the job
- the jobs current state
- the amount of cpu time or wall time used by the job
- the amount of SRFS space requested on the big file system
- the amount of SRFS space requested on the fast file system
- the amount of space requested on the parallel I/O file system

The last three fields may not contain useful information at all sites or on all systems

Displaying queue status

If queue status is being displayed and the **-f** option was not specified, the following items are displayed on a single line, in the specified order, separated by white space:

- the queue name
- the maximum number of jobs that may be run in the queue concurrently
- the total number of jobs in the queue
- the enable or disabled status of the queue
- the started or stopped status of the queue
- for each job state, the name of the state and the number of jobs in the queue in that state
- the type of queue, execution or routing

If queue status is being displayed and the **-f** option is specified, the output will depend on whether qstat was compiled to use a Tcl interpreter. See the configuration section for details. If Tcl is not being used, the full display for each queue consists of the header line:

```
Queue: queue_name
```

Followed by one line per queue attribute of the form:

```
attribute_name = value
```

If the **-q** option is specified, queue information is displayed in the alternative format: The following information is displayed on a single line:

- the queue name
- the maximum amount of memory a job in the queue may request
- the maximum amount of cpu time a job in the queue may request
- the maximum amount of wall time a job in the queue may request
- the maximum amount of nodes a job in the queue may request
- the number of jobs in the queue in the running state
- the number of jobs in the queue in the queued state
- the maximum number (limit) of jobs that may be run in the queue concurrently
- the state of the queue given by a pair of letters:
  - either the letter *E* if the queue is Enabled or *D* if Disabled

    and
  - either the letter *R* if the queue is Running (started) or *S* if Stopped.

Displaying server status

If batch server status is being displayed and the **-f** option is not specified, the following items are displayed on a single line, in the specified order, separated by white space:

- the server name
- the maximum number of jobs that the server may run concurrently
- the total number of jobs currently managed by the server
- the status of the server
- for each job state, the name of the state and the number of jobs in the server in that state

If server status is being displayed and the **-f** option is specified, the output will depend on whether qstat was compiled to use a Tcl interpreter. See the configuration section for details. If Tcl is not being used, the full display for the server consists of the header line:

```
Server: server name
```

Followed by one line per server attribute of the form:

```
attribute_name = value
```

Standard error

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

Configuration

If qstat is compiled with an option to include a Tcl interpreter, using the **-f** flag to get a full display causes a check to be made for a script file to use to output the requested information. The first location checked is `$HOME/.qstatrc` . If this does not exist, the next location checked is administrator configured. If one of these is found, a Tcl interpreter is started and the script file is passed to it along with three global variables. The command line arguments are split into two variable named flags and operands . The status information is passed in a variable named objects . All of these variables are Tcl lists. The flags list contains the name of the command (usually "qstat") as its first element. Any other elements are command line option flags with any options they use, presented in the order given on the command line. They are broken up individually so that if two flags are given together on the command line, they are separated in the list. For example, if the user typed:

```
qstat -QfWbigdisplay
```

the flags list would contain

```
qstat -Q -f -W bigdisplay
```

The operands list contains all other command line arguments following the flags. There will always be at least one element in operands because if no operands are typed by the user, the default destination or server name is used. The objects list contains all the information retrieved from the server(s) so the Tcl interpreter can run once to format the entire output. This list has the same number of elements as the operands list. Each element is another list with two elements.

The first element is a string giving the type of objects to be found in the second. The string can take the values "server", "queue", "job" or "error".

The second element will be a list in which each element is a single batch status object of the type given by the string discussed above. In the case of "error", the list will be empty. Each object is again a list. The first element is the name of the object. The second is a list of attributes.

The third element will be the object text.

All three of these object elements correspond with fields in the structure batch_status which is described in detail for each type of object by the man pages for pbs_statjob(3), pbs_statque(3), and pbs_statserver(3). Each attribute in the second element list whose elements correspond with the attrl structure. Each will be a list with two elements. The first will be the attribute name and the second will be the attribute value.

Exit status

Upon successful processing of all the operands presented to the qstat command, the exit status will be a value of zero.

If the qstat command fails to process any operand, the command exits with a value greater than zero.

Related Topics

qalter(1B)
qsub(1B)

Non-Adaptive Computing topics

- pbs_alterjob(3B)
- pbs_statjob(3B)
- pbs_statque(3B)
- pbs_statserver(3B)
- pbs_submit(3B)
- pbs_job_attributes(7B)
- pbs_queue_attributes(7B)
- pbs_server_attributes(7B)
- qmgr query_other_jobs parameter (allow non-admin users to see other users' jobs
- pbs_resources_*(7B) where * is system type
- PBS ERS

qsub

Submit PBS job.

Synopsis

```
qsub [-a date_time][-A account_string][-b secs][-c checkpoint_
options][-C directive_prefix][-d path][-D path][-e path][-f][-
F][-h][-i idle_slot_limit][-I][-j join][-k keep][-K kill_
delay][-l resource_list][-L NUMA_resource_list][-m mail_
options][-M user_list][-n node_exclusive][-N name][-o path][-p
priority][-P user[:group]][-q destination] [-r][-S path_to_
shell(s)][-t array_request] [-T script] [-u userlist] [-v
variable_list][-V][-w path][-W additional_attributes][-x][-X]
[-z][script]
```

Description

To create a job is to submit an executable script to a batch server. The batch server will be the default server unless the **-q** option is specified. The command parses a script prior to the actual script execution; it does not execute a script itself. All script-writing rules remain in effect, including the "#!" at the head of the file (see discussion of PBS_DEFAULT under Environment variables). Typically, the script is a shell script which will be executed by a command shell such as sh or csh.

Options on the qsub command allow the specification of attributes which affect the behavior of the job.

The qsub command will pass certain environment variables in the Variable_List attribute of the job. These variables will be available to the job. The value for the following variables will be taken from the environment of the qsub command: HOME, LANG, LOGNAME, PATH, MAIL, SHELL, and TZ. These values will be assigned to a new name which is the current name prefixed with the string "PBS_O_". For example, the job will have access to an environment variable named PBS_O_HOME which have the value of the variable HOME in the qsub command environment.

In addition to the above, the following environment variables will be available to the batch job:

Variable	Description
PBS_ARRAYID	Each member of a job array is assigned a unique identifier (see **-t** option).
PBS_ENVIRONMENT	Set to PBS_BATCH to indicate the job is a batch job, or to PBS_INTERACTIVE to indicate the job is a PBS interactive job (see **qsub** option).

Variable	Description
PBS_GPUFILE	The name of the file containing the list of assigned GPUs. For more information about how to set up Torque with GPUS, see Accelerators in the Moab Workload Manager *Administrator Guide*.
PBS_JOBID	The job identifier assigned to the job by the batch system. It can be used in the stdout and stderr paths. Torque replaces `$PBS_JOBID` with the job's jobid (for example, `#PBS -o /tmp/$PBS_JOBID.output`).
PBS_JOBNAME	The job name supplied by the user.
PBS_NODEFILE	The name of the file contains the list of nodes assigned to the job (for parallel and cluster systems).
PBS_O_HOST	The name of the host upon which the `qsub` command is running.
PBS_O_QUEUE	The name of the original queue to which the job was submitted.
PBS_O_WORKDIR	The absolute path of the current working directory of the `qsub` command.
PBS_QUEUE	The name of the queue from which the job is executed.
PBS_SERVER	The hostname of the pbs_server which `qsub` submits the job to.

Options

Option	Argument	Description
-a	date_time	Declares the time after which the job is eligible for execution. The date_time argument is in the form: `[[[[CC]YY]MM]DD]hhmm[.SS]` where *CC* is the first two digits of the year (the century), *YY* is the second two digits of the year, *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. If the month (MM) is not specified, it will default to the current month if the specified day (DD) is in the future. Otherwise, the month will be set to next month. Likewise, if the day (DD) is not specified, it will default to today if the time (hhmm) is in the future. Otherwise, the day will be set to tomorrow. For example, if you submit a job at 11:15 am with a time of `-a 1110`, the job will be eligible to run at 11:10 am tomorrow.

Option	Argument	Description
-A	account_ string	Defines the account string associated with the job. The account_string is an undefined string of characters and is interpreted by the server which executes the job. See section 2.7.1 of the PBS External Reference Specification (included in the Torque download tarball in docs/v2_2_ers.pdf).
-b	seconds	Defines the maximum number of seconds qsub will block attempting to contact pbs_ server. If pbs_server is down, or for a variety of communication failures, `qsub` will continually retry connecting to pbs_server for job submission. This value overrides the CLIENTRETRY parameter in `torque.cfg`. This is a non-portable Torque extension. Portability-minded users can use the PBS_CLIENTRETRY environmental variable. A negative value is interpreted as infinity. The default is 0.
-c	checkpoint_ options	Defines the options that will apply to the job. If the job executes upon a host which does not support checkpoint, these options will be ignored. Valid checkpoint options are: • *none* – No checkpointing is to be performed. • *enabled* – Specify that checkpointing is allowed but must be explicitly invoked by either the **qhold** or **qchkpt** commands. • *shutdown* – Specify that checkpointing is to be done on a job at pbs_mom shutdown. • *periodic* – Specify that periodic checkpointing is enabled. The default interval is 10 minutes and can be changed by the $checkpoint_interval option in the MOM config file or by specifying an interval when the job is submitted • *interval=minutes* – Checkpointing is to be performed at an interval of minutes, which is the integer number of minutes of wall time used by the job. This value must be greater than zero. • *depth=number* – Specify a number (depth) of checkpoint images to be kept in the checkpoint directory. • *dir=path* – Specify a checkpoint directory (default is `/var/spool/torque/checkpoint`).
-C	directive_ prefix	Defines the prefix that declares a directive to the `qsub` command within the script file. (See the paragraph on script directives under Extended description.) If the `-C` option is presented with a directive_prefix argument that is the null string, `qsub` will not scan the script file for directives.
-d	path	Defines the working directory path to be used for the job. If the `-d` option is not specified, the default working directory is the home directory. This option sets the environment variable PBS_O_INITDIR.
-D	path	Defines the root directory to be used for the job. This option sets the environment variable PBS_O_ROOTDIR.

Option	Argument	Description
-e	path	Defines the path to be used for the standard error stream of the batch job. The path argument is of the form: `[hostname:]path_name` where *hostname* is the name of a host to which the file will be returned, and *path_name* is the path name on that host in the syntax recognized by POSIX. When specifying a directory for the location you need to include a trailing slash. The argument will be interpreted as follows: - *path_name* – where *path_name* is not an absolute path name, then the `qsub` command will expand the path name relative to the current working directory of the command. The command will supply the name of the host upon which it is executing for the hostname component. - *hostname:path_name* – where *path_name* is not an absolute path name, then the `qsub`command will not expand the path name relative to the current working directory of the command. On delivery of the standard error, the path name will be expanded relative to the user's home directory on the hostname system. - *path_name* – where *path_name* specifies an absolute path name, then the `qsub` will supply the name of the host on which it is executing for the hostname. - *hostname:path_name* – where *path_name* specifies an absolute path name, the path will be used as specified. If the `-e` option is not specified, the default file name for the standard error stream will be used. The default name has the following form: - *job_name.esequence_number* – where *job_name* is the name of the job (see the **qsub** name option) and *sequence_number* is the job number assigned when the job is submitted.
-f	---	Job is made fault tolerant. Jobs running on multiple nodes are periodically polled by mother superior. If one of the nodes fails to report, the job is canceled by mother superior and a failure is reported. If a job is fault tolerant, it will not be canceled based on failed polling (no matter how many nodes fail to report). This may be desirable if transient network failures are causing large jobs not to complete, where ignoring one failed polling attempt can be corrected at the next polling attempt. If Torque is compiled with PBS_NO_POSIX_VIOLATION (there is no config option for this), you have to use `-W fault_tolerant=true`to mark the job as fault tolerant.

Option	Argument	Description
-F	---	Specifies the arguments that will be passed to the job script when the script is launched. The accepted syntax is: `qsub -F "myarg1 myarg2 myarg3=myarg3value" myscript2.sh` Quotation marks are required. `qsub`will fail with an error message if the argument following `-F` is not a quoted value. The pbs_mom server will pass the quoted value as arguments to the job script when it launches the script.
-h	---	Specifies that a user hold be applied to the job at submission time.
-i	idle_slot_ limit	Sets an idle slot limit for the job array being submitted. If this parameter is set for a non-array job, it will be rejected. Additionally, if the user requests an idle slot limit that exceeds the server parameter's default, the job will be rejected. See also the idle_ slot_limit server parameter. `$ qsub -t 0-99 -i 10 script.sh` *The submitted array will only instantiate 10 idle jobs; instead of all 100 jobs at submission time.*
-I	---	Declares that the job is to be run "interactively". The job will be queued and scheduled as any PBS batch job, but when executed, the standard input, output, and error streams of the job are connected through `qsub`to the terminal session in which `qsub` is running. Interactive jobs are forced to not rerunable. See Extended description for additional information of interactive jobs.
-j	join	Declares if the standard error stream of the job will be merged with the standard output stream of the job. An option argument value of *oe* directs that the two streams will be merged, intermixed, as standard output. An option argument value of *eo* directs that the two streams will be merged, intermixed, as standard error. If the join argument is *n* or the option is not specified, the two streams will be two separate files. If using either the -e or the -o option and the `-j eo|oe`option, the `-j` option takes precedence and all standard error and output messages go to the chosen output file.

Option	Argument	Description
-k	keep	Defines which (if either) of standard output or standard error will be retained on the execution host. If set for a stream, this option overrides the path name for that stream. If not set, neither stream is retained on the execution host. The argument is either the single letter "e" or "o", or the letters "e" and "o" combined in either order. Or the argument is the letter "n". - *e* – The standard error stream is to be retained on the execution host. The stream will be placed in the home directory of the user under whose user id the job executed. The file name will be the default file name given by: `job_name.esequence` where *job_name* is the name specified for the job, and *sequence* is the sequence number component of the job identifier. - *o* – The standard output stream is to be retained on the execution host. The stream will be placed in the home directory of the user under whose user id the job executed. The file name will be the default file name given by: `job_name.osequence` where *job_name* is the name specified for the job, and *sequence* is the sequence number component of the job identifier. - *eo* – Both the standard output and standard error streams will be retained. - *oe* – Both the standard output and standard error streams will be retained. - *n* – Neither stream is retained.
-K	kill_delay	When set on a job, overrides server and queue kill_delay settings. The kill_delay value is a positive integer. The default is 0. See kill_delay on page 327 for more information.

Option	Argument	Description
-l	resource_list	Defines the resources that are required by the job and establishes a limit to the amount of resource that can be consumed. See Requesting Resources on page 73 for more information. If not set for a generally available resource, such as CPU time, the limit is infinite. The **resource_list** argument is of the form: `resource_name[=[value]][,resource_name[=[value]],...]` In this situation, you should request the more inclusive resource first. For example, a request for procs should come before a gres request. In Torque 3.0.2 or later, `qsub` supports the mapping of `-l gpus=X` to `-l gres=gpus:X` This allows users who are using NUMA systems to make requests such as `-l ncpus=20:gpus=5` indicating they are not concerned with the GPUs in relation to the NUMA nodes they request, they only want a total of 20 cores and 5 GPUs. -l supports some Moab-only extensions. See Requesting Resources on page 73 for more information on native Torque resources. qsub -W x= is recommended instead (supports more options). See -W for more information. For information on specifying multiple types of resources for allocation, see Multi-Req Support in the *Moab Workload Manager Administrator Guide*.
-L	NUMA_ resource_list	Available with Torque 6.0 and later. This uses a different syntax than the -l resource_list option. Defines the NUMA-aware resource requests for NUMA hardware. This option will work with non-NUMA hardware. See -L NUMA Resource Request on page 188 for the syntax and valid values.
-m	mail_options	Defines the set of conditions under which the execution server will send a mail message about the job. The mail_options argument is a string which consists of either the single character "n" or "p", or one or more of the characters "a", "b", "e", and "f". If the character "n" is specified, no normal mail is sent. Mail for job cancels and other events outside of normal job processing are still sent. If the character "p" is specified, mail will never be sent for the job. For the characters"a", "b", "e" and "f": • *a* – Mail is sent when the job is aborted by the batch system. • *b* – Mail is sent when the job begins execution. • *e* – Mail is sent when the job terminates. • *f* – Mail is sent when the job terminates with a non-zero exit code. If the `-m` option is not specified, mail will be sent if the job is aborted.

Option	Argument	Description
-M	user_list	Declares the list of users to whom mail is sent by the execution server when it sends mail about the job. The user_list argument is of the form: `user[@host][,user[@host],...]` If unset, the list defaults to the submitting user at the `qsub` host, i.e. the job owner.

-n	node_exclus-ive	Allows a user to specify an exclusive-node access/allocation request for the job. This will set `node_exclusive = True` in the output of `qstat -f <job ID>`.
		For Moab, the following options are equivalent to "-n":
		`> qsub -l naccesspolicy=singlejob jobscript.sh`
		`# OR`
		`> qsub -W x=naccesspolicy:singlejob jobscript.sh`
		By default, this only applies for cpusets, and only for compatible schedulers (see Linux Cpuset Support on page 118).
		For systems that use Moab *and* have cgroups enabled, the recommended manner for assigning all cores is to use NUMA syntax: "`-L tasks=<count>:lprocs=all:place=node`".
		With cgroups, the ("-l") syntax (lowercase L) will, by default, restrict to the number of cores requested, or to the `resources_default.procs` value (i.e., 1 core, typically). In order to override this behavior and have Moab assign all the cores on a node while using "`-l...singlejob`" and/or "`-n`" (in other words, without "`-L ...lprocs=all...`"), you must also set `RMCFG[<torque>] FLAGS=MigrateAllJobAttributes` in moab.cfg.

-N	name	Declares a name for the job. The name specified may be an unlimited number of characters in length. It must consist of printable, nonwhite space characters with the first character alphabetic. If the `-N` option is not specified, the job name will be the base name of the job script file specified on the command line. If no script file name was specified and the script was read from the standard input, then the job name will be set to STDIN.

Option	Argument	Description
-o	path	Defines the path to be used for the standard output stream of the batch job. The path argument is of the form: `[hostname:]path_name` where *hostname* is the name of a host to which the file will be returned, and *path_name* is the path name on that host in the syntax recognized by POSIX. When specifying a directory for the location you need to include a trailing slash. The argument will be interpreted as follows: - *path_name* – where *path_name* is not an absolute path name, then the qsub command will expand the path name relative to the current working directory of the command. The command will supply the name of the host upon which it is executing for the hostname component. - *hostname:path_name* – where *path_name* is not an absolute path name, then the qsubcommand will not expand the path name relative to the current working directory of the command. On delivery of the standard output, the path name will be expanded relative to the user's home directory on the hostname system. - *path_name* – where *path_name* specifies an absolute path name, then the qsub will supply the name of the host on which it is executing for the hostname. - *hostname:path_name*where *path_name* specifies an absolute path name, the path will be used as specified. If the `-o` option is not specified, the default file name for the standard output stream will be used. The default name has the following form: - *job_name.osequence_number* – where *job_name* is the name of the job (see the **qsub** name option) and *sequence_number* is the job number assigned when the job is submitted.
-p	priority	Defines the priority of the job. The priority argument must be a integer between -1024 and +1023 inclusive. The default is no priority which is equivalent to a priority of zero.
-P	user [:group]	Allows a root user or manager to submit a job as another user. Torque treats proxy jobs as though the jobs were submitted by the supplied username. This feature is available in Torque 2.4.7 and later, however, Torque 2.4.7 does not have the ability to supply the `[:group]` option; it is available in Torque 2.4.8 and later.

Option	Argument	Description
-q	destination	Defines the destination of the job. The destination names a queue, a server, or a queue at a server. The qsubcommand will submit the script to the server defined by the destination argument. If the destination is a routing queue, the job may be routed by the server to a new destination. If the -qoption is not specified, the qsubcommand will submit the script to the default server. (See Environment variables and the PBS ERS section 2.7.4, "Default Server".) If the -qoption is specified, it is in one of the following three forms: - queue - @server - queue@server If the destination argument names a queue and does not name a server, the job will be submitted to the named queue at the default server. If the destination argument names a server and does not name a queue, the job will be submitted to the default queue at the named server. If the destination argument names both a queue and a server, the job will be submitted to the named queue at the named server.
-r	y/n	Declares whether the job is rerunable (see the **qrerun** command). The option argument is a single character, either y or n. If the argument is "y", the job is rerunable. If the argument is "n", the job is not rerunable. The default value is y, rerunable.

-S	path_list	Declares the path to the desires shell for this job.
		`qsub script.sh -S /bin/tcsh`
		If the shell path is different on different compute nodes, use the following syntax:
		`path[@host][,path[@host],...]`
		`qsub script.sh -S /bin/tcsh@node1,/usr/bin/tcsh@node2`
		Only one path may be specified for any host named. Only one path may be specified without the corresponding host name. The path selected will be the one with the host name that matched the name of the execution host. If no matching host is found, then the path specified without a host will be selected, if present.
		If the -S option is not specified, the option argument is the null string, or no entry from the path_list is selected, the execution will use the user's login shell on the execution host.

Option	Argument	Description
-t	array_ request	Specifies the task ids of a job array. Single task arrays are allowed. The array_request argument is an integer id or a range of integers. Multiple ids or id ranges can be combined in a comma delimited list. Examples: `-t 1-100 or -t 1,10,50-100` An optional *slot limit* can be specified to limit the amount of jobs that can run concurrently in the job array. The default value is unlimited. The slot limit must be the last thing specified in the array_request and is delimited from the array by a percent sign (%). `qsub script.sh -t 0-299%5` This sets the slot limit to 5. Only 5 jobs from this array can run at the same time. You can use **qalter** to modify slot limits on an array. The server parameter **max_slot_ limit** can be used to set a global slot limit policy.
-T	script	Specifies a prologue or epilogue script for the job. The full name of the scripts are `prologue.<script_name>` or `epilogue.<script_name>` but you only specify the `<script_name>` portion when using the -T option. For example, `qsub -T prescript` specifies the `prologue.prescript` script file.
-u		This option is deprecated and will not work as previously documented. Use -P.
-v	variable_list	Expands the list of environment variables that are exported to the job. In addition to the variables described in the "Description" section above, variable_list names environment variables from the `qsub` command environment which are made available to the job when it executes. The variable_list is a comma separated list of strings of the form `variable` or `variable=value` These variables and their values are passed to the job. Note that `-v` has a higher precedence than `-V`, so identically named variables specified via `-v` will provide the final value for an environment variable in the job.
-V	---	Declares that all environment variables in the `qsub` commands environment are to be exported to the batch job.
-w	path	Defines the working directory path to be used for the job. If the -w option is not specified, the default working directory is the current directory. This option sets the environment variable PBS_O_WORKDIR.

Option	Argument	Description

-W	additional_ attributes	Use "-W x=" as pass-through for scheduler-only job extensions. See Resource Manager Extensions in the *Moab Workload ManagerAdministrator Guide* for a list of scheduler-only job extensions.
		For legacy purposes, qsub -l will continue to support some scheduler-only job extensions. However, when in doubt, use "-W x=".
		The `-W` option allows for the specification of additional job attributes. The general syntax of `-W` is in the form:
		`-W attr_name=attr_value`
		You can use multiple `-W` options with this syntax:
		`-W attr_name1=attr_value1 -W attr_name2=attr_value2`
		If white space occurs anywhere within the option argument string or the equal sign, "=", occurs within an attribute_value string, then the string must be enclosed with either single or double quote marks.
		PBS currently supports the following attributes within the `-W` option:
		• *depend=dependency_list* – Defines the dependency between this and other jobs. The dependency_list is in the form:
		`type[:argument[:argument...][,type:argument...]`
		The argument is either a numeric count or a PBS job id according to type. If argument is a count, it must be greater than 0. If it is a job id and not fully specified in the form `seq_number.server.name`, it will be expanded according to the default server rules which apply to job IDs on most commands. If argument is null (the preceding colon need not be specified), the dependency of the corresponding type is cleared (unset). For more information, see depend=dependency_list valid dependencies.
		• *group_list=g_list* – Defines the group name under which the job is to run on the execution system. The g_list argument is of the form:
		`group[@host][,group[@host],...]`
		Only one group name may be given per specified host. Only one of the group specifications may be supplied without the corresponding host specification. That group name will used for execution on any host not named in the argument list. If not set, the group_list defaults to the primary group of the user under which the job will be run.
		• *interactive=true* – If the interactive attribute is specified, the job is an interactive job. The **qsub** option is an alternative method of specifying this attribute.
		• *job_radix=<int>* – To be used with parallel jobs. It directs the Mother Superior of the job to create a distribution radix of size *<int>* between sisters. See Managing Multi-Node Jobs.
		• *stagein=file_list*
		• *stageout=file_list* – Specifies which files are staged (copied) in before job start or staged out after the job completes execution. On completion of the job, all staged-in and staged-out files are removed from the execution system. The

Option	Argument	Description

		file_list is in the form:
		`local_file@hostname:remote_file[,...]`
		regardless of the direction of the copy. The name local_file is the name of the file on the system where the job executed. It may be an absolute path or relative to the home directory of the user. The name remote_file is the destination name on the host specified by hostname. The name may be absolute or relative to the user's home directory on the destination host. The use of wildcards in the file name is not recommended. The file names map to a remote copy program (rcp) call on the execution system in the follow manner:
		o For stagein: `rcp hostname:remote_file local_file`
		o For stageout: `rcp local_file hostname:remote_file`
		Data staging examples:
		`-W stagein=/tmp/input.txt@headnode:/home/user/input.txt`
		`-W stageout=/tmp/output.txt@headnode:/home/user/output.txt`
		If Torque has been compiled with wordexp support, then variables can be used in the specified paths. Currently only `$PBS_JOBID`, `$HOME`, and `$TMPDIR` are supported for stagein.
		l *umask=XXX* – Sets umask used to create stdout and stderr spool files in pbs_mom spool directory. Values starting with 0 are treated as octal values, otherwise the value is treated as a decimal umask value.

-x	---	By default, if you submit an interactive job with a script, the script will be parsed for PBS directives but the rest of the script will be ignored since it's an interactive job. The `-x` option allows the script to be executed in the interactive job and then the job completes. For example:
		`script.sh`
		`#!/bin/bash`
		`ls`
		`---end script---`
		`qsub -I script.sh`
		`qsub: waiting for job 5.napali to start`
		`dbeer@napali:#`
		`<displays the contents of the directory, because of the ls command>`
		`qsub: job 5.napali completed`

-X	---	Enables X11 forwarding. The DISPLAY environment variable must be set.
-z	---	Directs that the qsubcommand is not to write the job identifier assigned to the job to the commands standard output.

depend=dependency_list valid dependencies

For job dependencies to work correctly, you must set the keep_completed server parameter.

Dependency	Description
`synccount:count`	This job is the first in a set of jobs to be executed at the same time. Count is the number of additional jobs in the set.
`syncwith:jobid`	This job is an additional member of a set of jobs to be executed at the same time. In the above and following dependency types, jobid is the job identifier of the first job in the set.
`after:jobid[:jobid...]`	This job may be scheduled for execution at any point after jobs jobid have started execution.
`afterok:jobid[:jobid...]`	This job may be scheduled for execution only after jobs jobid have terminated with no errors. See the csh warning under Extended description.
`afternotok:jobid[:jobid...]`	This job may be scheduled for execution only after jobs jobid have terminated with errors. See the csh warning under Extended description.
`afterany:jobid[:jobid...]`	This job may be scheduled for execution after jobs jobid have terminated, with or without errors.
`on:count`	This job may be scheduled for execution after count dependencies on other jobs have been satisfied. This form is used in conjunction with one of the "before" forms (see below).
`before:jobid[:jobid...]`	When this job has begun execution, then jobs jobid... may begin.
`beforeok:jobid[:jobid...]`	If this job terminates execution without errors, then jobs jobid... may begin. See the csh warning under Extended description.
`beforenotok:jobid[:jobid...]`	If this job terminates execution with errors, then jobs jobid... may begin. See the csh warning under Extended description.

Dependency	Description
`beforeany:jobid[:jobid...]`	When this job terminates execution, jobs jobid... may begin. If any of the before forms are used, the jobs referenced by jobid must have been submitted with a dependency type of on. If any of the before forms are used, the jobs referenced by jobid must have the same owner as the job being submitted. Otherwise, the dependency is ignored.
Array dependencies make a job depend on an array or part of an array. If no count is given, then the entire array is assumed. For examples, see Dependency examples.	
`afterstartarray:arrayid[count]`	After this many jobs have started from arrayid, this job may start.
`afterokarray:arrayid[count]`	This job may be scheduled for execution only after jobs in arrayid have terminated with no errors.
`afternotokarray:arrayid[count]`	This job may be scheduled for execution only after jobs in arrayid have terminated with errors.
`afteranyarray:arrayid[count]`	This job may be scheduled for execution after jobs in arrayid have terminated, with or without errors.
`beforestartarray:arrayid[count]`	Before this many jobs have started from arrayid, this job may start.
`beforeokarray:arrayid[count]`	If this job terminates execution without errors, then jobs in arrayid may begin.
`beforenotokarray:arrayid[count]`	If this job terminates execution with errors, then jobs in arrayid may begin.
`beforeanyarray:arrayid[count]`	When this job terminates execution, jobs in arrayid may begin. If any of the before forms are used, the jobs referenced by arrayid must have been submitted with a dependency type of on. If any of the before forms are used, the jobs referenced by arrayid must have the same owner as the job being submitted. Otherwise, the dependency is ignored.

Dependency	Description
Error processing of the existence, state, or condition of the job on which the newly submitted job is a deferred service, i.e. the check is performed after the job is queued. If an error is detected, the new job will be deleted by the server. Mail will be sent to the job submitter stating the error.	
Jobs can depend on single job dependencies and array dependencies at the same time.	
`afterok:jobid [:jobid...],afterokarray:arrayid [count]`	This job may be scheduled for execution only after jobs jobid and jobs in arrayid have terminated with no errors.

Dependency examples

```
qsub -W depend=afterok:123.big.iron.com /tmp/script
```

```
qsub -W depend=before:234.hunk1.com:235.hunk1.com
```

```
/tmp/script
```

```
qsub script.sh -W depend=afterokarray:427[]
```

(This assumes every job in array 427 has to finish successfully for the dependency to be satisfied.)

```
qsub script.sh -W depend=afterokarray:427[][5]
```

(This means that 5 of the jobs in array 427 have to successfully finish in order for the dependency to be satisfied.)

```
qsub script.sh -W depend=afterok:360976,afterokarray:360977[]
```

(Job 360976 and all jobs in array 360977 have to successfully finish for the dependency to be satisfied.)

Operands

The qsub command accepts a script operand that is the path to the script of the job. If the path is relative, it will be expanded relative to the working directory of the qsub command.

If the script operand is not provided or the operand is the single character "-", the qsub command reads the script from standard input. When the script is being read from Standard Input, qsub will copy the file to a temporary file. This temporary file is passed to the library interface routine pbs_submit. The temporary file is removed by qsub after pbs_submit returns or upon the receipt of a signal which would cause qsub to terminate.

Standard input

The qsub command reads the script for the job from standard input if the script operand is missing or is the single character "-".

Input files

The script file is read by the qsub command. qsub acts upon any directives found in the script.

When the job is created, a copy of the script file is made and that copy cannot be modified.

Standard output

Unless the **-z** option is set, the job identifier assigned to the job will be written to standard output if the job is successfully created.

Standard error

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

Environment variables

The values of some or all of the variables in the qsub commands environment are exported with the job (see the **qsub** and **qsub** options).

The environment variable PBS_DEFAULT defines the name of the default server. Typically, it corresponds to the system name of the host on which the server is running. If PBS_DEFAULT is not set, the default is defined by an administrator established file.

The environment variable PBS_DPREFIX determines the prefix string which identifies directives in the script.

The environment variable PBS_CLIENTRETRY defines the maximum number of seconds qsub will block (see the **-b** option). Despite the name, currently qsub is the only client that supports this option.

torque.cfg

The `torque.cfg` file, located in PBS_SERVER_HOME (`/var/spool/torque` by default) controls the behavior of the qsub command. This file contains a list of parameters and values separated by whitespace. See "torque.cfg" Configuration File on page 403 for more information on these parameters.

Extended description

Script Processing:

A job script may consist of PBS directives, comments and executable statements. A PBS directive provides a way of specifying job attributes in addition to the command line options. For example:

```
:
#PBS -N Job_name
#PBS -l walltime=10:30,mem=320kb
#PBS -m be
#
step1 arg1 arg2
step2 arg3 arg4
```

The qsub command scans the lines of the script file for directives. An initial line in the script that begins with the characters "*#!*" or the character "*:*" will be ignored and scanning will start with the next line. Scanning will continue until the first executable line, that is a line that is not blank, not a directive line, nor a line whose first nonwhite space character is "#". If directives occur on subsequent lines, they will be ignored.

A line in the script file will be processed as a directive to qsub if and only if the string of characters starting with the first nonwhite space character on the line and of the same length as the directive prefix matches the directive prefix.

The remainder of the directive line consists of the options to qsub in the same syntax as they appear on the command line. The option character is to be preceded with the "-" character.

If an option is present in both a directive and on the command line, that option and its argument, if any, will be ignored in the directive. The command line takes precedence.

If an option is present in a directive and not on the command line, that option and its argument, if any, will be processed as if it had occurred on the command line.

The directive prefix string will be determined in order of preference from:

- The value of the **-C** option argument if the option is specified on the command line.
- The value of the environment variable PBS_DPREFIX if it is defined.
- The four character string #PBS.

If the **-C** option is found in a directive in the script file, it will be ignored.

C-Shell .logout File:

The following warning applies for users of the c-shell, csh. If the job is executed under the 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. To preserve the job exit status, either remove the `.logout` file or place the following line as the first line in the `.logout` file:

```
set EXITVAL = $status
```

and the following line as the last executable line in `.logout`:

```
exit $EXITVAL
```

Interactive Jobs:

If the **qsub** option is specified on the command line or in a script directive, or if the "interactive" job attribute declared true via the **-W** option, `-W interactive=true`, either on the command line or in a script directive, the job is an interactive job. The script will be processed for directives, but will not be included with the job. When the job begins execution, all input to the job is from the terminal session in which qsub is running.

When an interactive job is submitted, the qsub command will not terminate when the job is submitted. qsub will remain running until the job terminates, is aborted, or the user interrupts qsub with an SIGINT (the control-C key). If qsub is interrupted prior to job start, it will query if the user wishes to exit. If the user response "yes", qsub exits and the job is aborted.

One the interactive job has started execution, input to and output from the job pass through qsub. Keyboard generated interrupts are passed to the job. Lines entered that begin with the tilde (~) character and contain special sequences are escaped by qsub. The recognized escape sequences are:

Sequence	Description
~.	qsubterminates execution. The batch job is also terminated.
~susp	Suspend the qsubprogram if running under the C shell. "susp" is the suspend character (usually CNTL-Z).
~asusp	Suspend the input half of qsub(terminal to job), but allow output to continue to be displayed. Only works under the C shell. "asusp" is the auxiliary suspend character, usually CNTL-Y.

Exit status

Upon successful processing, the qsub exit status will be a value of zero.

If the qsub command fails, the command exits with a value greater than zero.

Related Topics

qalter(1B)

qdel(1B)

qhold(1B)

qrls(1B)
qsig(1B)
qstat(1B)
pbs_server(8B)

Non-Adaptive Computing topics

- pbs_connect(3B)
- pbs_job_attributes(7B)
- pbs_queue_attributes(7B)
- pbs_resources_irix5(7B)
- pbs_resources_sp2(7B)
- pbs_resources_sunos4(7B)
- pbs_resources_unicos8(7B)
- pbs_server_attributes(7B)
- qselect(1B)
- qmove(1B)
- qmsg(1B)
- qrerun(1B)

qterm

Terminate processing by a PBS batch server.

Synopsis

```
qterm [-l] [-t type] [server...]
```

Description

The qterm command terminates a batch server. When a server receives a terminate command, the server will go into the "Terminating" state. No new jobs will be allowed to be started into execution or enqueued into the server. The impact on jobs currently being run by the server depends

In order to execute qterm, the user must have PBS Operation or Manager privileges.

Options

Option	Name	Description
-l	local	Terminate processing only if the active server is local to where qterm is being executed.

Option	Name	Description
-t	type	Specifies the type of shut down. The types are: • *quick* – This is the default action if the `-t` option is not specified. This option is used when you wish that running jobs be left running when the server shuts down. The server will cleanly shutdown and can be restarted when desired. Upon restart of the server, jobs that continue to run are shown as running; jobs that terminated during the server's absence will be placed into the exiting state. The immediate and delay types are deprecated.

Operands

The server operand specifies which servers are to shut down. If no servers are given, then the default server will be terminated.

Standard error

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

Exit status

Upon successful processing of all the operands presented to the qterm command, the exit status will be a value of zero.

If the qterm command fails to process any operand, the command exits with a value greater than zero.

Related Topics(non-Adaptive Computing topics)

pbs_server(8B)

qmgr(8B)

pbs_resources_aix4(7B)

pbs_resources_irix5(7B)

pbs_resources_sp2(7B)

pbs_resources_sunos4(7B)

pbs_resources_unicos8(7B)

trqauthd

(Torque authorization daemon)

Synopsis

```
trqauthd -D
trqauthd -d
trqauthd -F
trqauthd --logfile_dir
trqauthd -n
```

Description

The trqauthd daemon, introduced in Torque 4.0.0, replaced the pbs_iff authentication process. When users connect to pbs_server by calling one of the Torque utilities or by using the Torque APIs, the new user connection must be authorized by a trusted entity which runs as root. The advantage of trqauthd's doing this rather than pbs_iff is that trqauthd is resident, meaning you do not need to be loaded every time a connection is made; multi-threaded; scalable; and more easily adapted to new functionality than pbs_iff.

Beginning in Torque 4.2.6, trqauthd can remember the currently active pbs_server host, enhancing high availability functionality. Previously, trqauthd tried to connect to each host in the `TORQUE_HOME/` *<server_name>* file until it could successfully connect. Because it now remembers the active server, it tries to connect to that server first. If it fails to connect, it will go through the *<server_name>* file and try to connect to a host where an active pbs_server is running.

Beginning in Torque 6.1.0, you have the option when starting trqauthd to disable trqauthd from logging anything. In addition, the -F (don't fork) option is available when running under systemd.

If you run trqauthd before starting pbs_server, you will receive a warning that no servers are available. To avoid this message, start pbs_server before running trqauthd.

Options

-D — Debug	
Format	---
Default	---
Description	Run trqauthd in debug mode.
Example	`trqauthd -D`

-d — Terminate

Format	---
Default	---
Description	Terminate trqauthd.
Example	`trqauthd -d`

-F — Fork

Format	---
Default	---
Description	Prevents the system from forking. Useful when running under systemd (Red Hat 7-based or SUSE 12-based systems).
Example	`trqauthd -F`

--logfile_dir — Specify log file directory

Format	=<path>
Default	---
Description	Specifies custom directory for trqauthd log file.
Example	`trqauthd --logfile_dir=/logs`

-n — No Logging

Format	---
Default	---
Description	Disables trqauthd from logging anything.

-n — No Logging

Example	`trqauthd -n`