

# Common Cobalt Administrative Tasks

## Startup/Shutdown

Due to the use of `setuid` and `setgid` within Cobalt, it is currently a requirement to run the Cobalt daemons as root.

### Startup

Cobalt may be started up by running `/etc/init.d/cobalt start` as root on the Service Node (`mirasn1`) and the Script Login (`miralac1`). When switching to root to bring up Cobalt, make sure that you get a clean root login environment. Ensure that the control system is up and running and that the additional `runjob_mux` is running on the Script Login before bringing up Cobalt.

### Single-Component Startup

A single component of Cobalt may be restarted by running `/etc/init.d/cobalt start-component <component-name>`.

### Cobalt Shutdown

To shutdown Cobalt, run `/etc/init.d/cobalt stop`. If restarting Cobalt, make sure that all components have fully shut down prior to running the startup command. Additionally, it is highly recommended that you suspend scheduling, place a reservation, and ensure that no jobs are currently running. This operation is fatal to all currently running jobs managed by Cobalt. Jobs that are being run outside of Cobalt are unaffected.

### Single-Component Shutdown

A single component of Cobalt may be restarted by running `/etc/init.d/cobalt stop-component <component-name>`. Depending on the component, this operation may be fatal to running jobs.

### A note on currently running jobs

If a forker is restarted, then any running process beneath that forker is considered lost. This operation is fatal to any running job or auxiliary script beneath that forker. Additionally, restarting the system component is fatal to all jobs running through Cobalt on the BlueGene/Q.

## Components for the BlueGene/Q Configuration

### Service Node Components

- slp - Service location component
- cqm - Queue-Manager
- bgsched - scheduler
- bgqsystem - system
- system\_script\_forker - forker for auxiliary internal Cobalt scripts
- bg\_runjob\_forker - forker for non-script mode jobs (invokes runjob directly)  
cdbwriter - database writer (may run without)

### Script Login Components

- user\_script\_forker - forker that runs user-specified scripts (script-mode jobs)

---

## Reservations

For further information, please consult the the setres manpage

### Setting Reservations

*setres -n <name>-s <start-time>-d <duration><block-id1><block-id2>...* Reservations may be set with the setres command. Notes on options:

- the start-time is in the format of YYYY-MM-DD-HH:mm
- the duration is in HH:mm:ss format
- if a block is included in the list of locations, then all blocks that have compute nodes that intersect the locations will be included in the reservation. This does not include passthrough-only blocks, however.

All times are in UTC, unless you set your TZ variable. It is recommended that hardware replacement reservations be named *hw.<location-name>*

### Modifying Reservations

A reservation may be modified by using *setres -m -n <name>[options-and-new-values]* You may pass in any option that you can specify with *setres*. These new values will replace the old values. Keep in mind that if you want to extend a reservation, you need to specify the time from the reservation start (i.e. -d 1:15:00 to add 15 minutes to a 1 hour reservation)

## Releasing Reservations

Running *releaseres* <reservation-name> with administrator permissions will release any defined reservation. This command will take a list of reservation names. A cyclic reservation may be deferred via *setres -D -n* <name>. *releaseres* will delete a cyclic reservation, not defer it to its next cycle.

---

## Scheduling

Scheduling may be halted and resumed via *schedctl*.

- *schedctl --start* resumes scheduling
- *schedctl --stop* halts scheduling immediately
- *schedctl --status* reports if scheduling has already been disabled

This will immediately halt all scheduling from Cobalt, and will prevent any new jobs from running. This also prevents reservations from running.

---

## Jobs

The *cqadm* command is the queue-manager administration command.

### Administrative Hold

*cqadm --hold jobid1 jobid2 ...*

This command will place a job into a hold state, which prevents the job from running. This command will accept a list of jobids. It may not be run with any other *cqadm* flag.

### Administrative Hold Release

*cqadm --release jobid1 jobid2 ...*

This will release an admin hold on all listed jobs. If a job has any other type of hold on it, such as a user-hold placed on the job by the user via *qhold*.

## Administrative Job Kill

*cqadm --kill jobid*

This is equivalent to running `qdel` on a job as that user. This allows an administrator to remove any job from the queue

## Force Delete

**CAUTION: This command may cause resources to become unavailable due to bypassing the normal resource cleanup!**

*cqadm --kill --force jobid*

This forces cobalt to delete a job from the queue manager. It will make one last attempt to free the resources from the system component. However, it will not guarantee the resources are free. Use this only as a last resort.

## Changing Job Score

This is handled through `schedctl`:

*schedctl --score new-score jobid*

This will set a job's score to a new value. Cobalt will always try to run the highest scored job first, so long as there is hardware available that can run the job.

---

# Block Administration

## Block Details

To obtain a detailed description of a block and its associated hardware from Cobalt's perspective: *partadm -b blockid1 blockid2 ....*

## Listing Blocks

A list of all blocks that Cobalt is tracking, including ones marked non-functional, can be obtained by using *partadm -l*.

## Disabling Single Blocks

A single block may be disabled, and removed from scheduling decisions using *partadm --disable blockid1 blockid2 ....* This removes the block from the display in `partlist`.

## Enabling Single Blocks

A single block may be enabled, and returned from scheduling decisions using `partadm --enable blockid1 blockid2 ....`. This restores the block in partlist.

## Boot Control

Block booting can be controled/queried by the following commands:

- `partadm --boot-status`
- `partadm --boot-start`
- `partadm --boot-stop`

Halting booting without first halting scheduling is strongly discouraged and could result in allocated jobs being prematurely cleaned.

## Force-Cleaning

If scheduling is stopped and booting is halted a block may be force-cleared by issuing `partadm -c blockid`. Check the logs to ensure that the block has cleared prior to resuming booting and scheduling.

---

## Cluster Systems

Most of the above applies to cluster systems, as all of the components are identical between clusters and BlueGenes. The exception is the system component. Cluster systems have their own system component, as well as administrative commands, and stat command. It should be noted that `partadm` and `partlist` do not work on cluster systems. `nodeadm` and `nodelist` are used instead.

## Nodeadm

### Adding/Removing Nodes

Nodes may be added by adding them to the hostfile (by default: `/etc/cobalt.hostfile`). Likewise, you may remove nodes by removing their entries from the hostfile.

### Adding/Removing Queues

Queues may be added or removed by using `nodeadm` to assign queues to nodes. Nodes may be members of many queues. To change the queues that are assigned in a ':'-delimited list:

```
nodeadm --queue [list-of-queues] [list-of-target-nodes]
```

## Recovering From Errors

Nodes may be marked as being in error by being put into the *down* state. This is triggered by a failure in the setup or cleanup scripts (prologues and epilogues). Once the condition causing the failure has been dealt with, a node may be returned to scheduling by issuing the following command

```
nodeadm --up [list of nodes to restore]
```

## Marking Nodes Offline

Should a situation arise where a node has an intermittent failure, for instance, a node with marginal hardware, a node may be removed from scheduling using the following

```
nodeadm --down [affected nodes]
```

## Clearing "Stuck" Nodes

Certain failure modes may result in a node being stuck in the allocated state. While efforts have been made to avoid this, should this occur, you can clear the *allocated* status from the node by running:

```
nodeadm --down [affected nodes] nodeadm --up [affected nodes]
```

Be careful doing this. It is likely that cleanup actions were unable to complete, particularly in the event of a partial allocation. The administrator should make sure that the node is free of user processes before performing this operation. Afterwards, the node will be in the *idle* state.

## Nodelist

Nodelist is a straight-across replacement for partlist. Effectively it's the same as running *nodeadm -l* at this point in time.