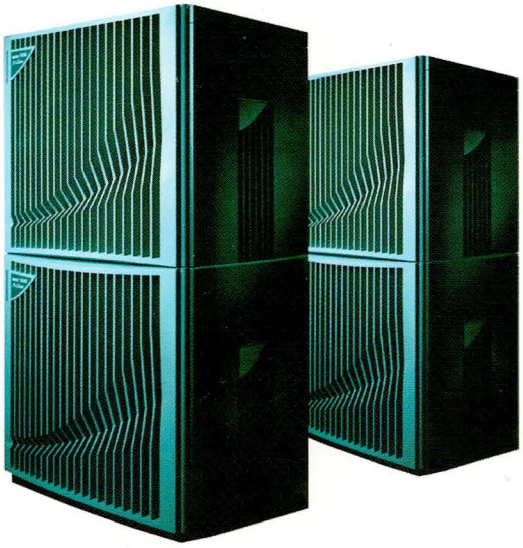
HEWLETT® PACKARD

S-Class and
X-Class Servers



NQS System Administration Guide

Fourth Edition

NQS System Administration Guide

Exemplar S-Class and X-Class Servers

B5589-90008

Fourth Edition

June 1997

Hewlett-Packard Company
Convex Division
Richardson, Texas
United States of America

NQS System Administration Guide

Exemplar S-Class and X-Class Servers

B5589-90008

Notice

The information contained in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

This entire book is recyclable.

Printed in the United States of America

Revision information for

NQS System Administration Guide

Exemplar S-Class and X-Class Servers

Edition	Document No.	Description
Fourth	B5589-90008	Released June 1997. Updated platform references to Exemplar Technical Servers.
Third	B5589-90001	Released January 1997 with NQS V2.3. Includes general updates.
Second	770-007430-001	Released September 1995.
First	770-007430-000	Initial release August 1994.

Contents

Figures

Tables

Preface

Purpose and audience

The *NQS System Administration Guide* is written for Network Queuing System (NQS) managers on Exemplar S-Class Technical Servers, and it covers the following topics:

- Basic NQS concepts
- How to configure your NQS network
- How to maintain queues and queue requests on a regular basis

Conventions

This section discusses notational conventions used in this book.

Command syntax

Consider the following example:

COMMAND *input_file* [...] {a | b} *[output_file]*

COMMAND

Must be typed as it appears.

input_file

Indicates a file name that must be supplied by the user.

[...]

The horizontal ellipsis in brackets indicates that additional input file names may be supplied.

{a | b}

Either a or b must be supplied.

[output_file]

Enclosed in brackets indicates an optional file name supplied by the user.

General conventions

In general, the following conventions are used in this guide:

- *Italics:*
  - Designate user-supplied variables in a command-line example
  - Introduce new and important terms
  - Indicate document titles
- `Constant-width font` designates:
  - System output in screens and examples
  - Command names and options
  - Directives, program statements, display examples, printout examples, and error messages returned
- **Bold, constant-width font** designates user input in screens and examples, and must be typed exactly as it appears.
- Vertical ellipsis shows that lines of code have been left out of an example.
- Words and abbreviations that indicate keyboard keys you press are identified in a distinctive bold type. For example, **RETURN** refers to the carriage return key. Words separated by a hyphen indicate two keys that you must press simultaneously. For example, **CTRL-X** indicates you must press and hold down the **CTRL** key and then press the **X** key.
- The word "enter" in a phrase such as "enter **`ls`**" means you type the command and then press **RETURN**.
- References to man pages appear in the form adb(1), where the name of the man page is followed by its section number enclosed in parentheses.

Caution

A Caution highlights information necessary to avoid damage to the system.

Associated documentation

Using this software may require information not specific to the tasks described in this document.

For more information you can order the following books from Hewlett-Packard:

NQS User's Guide (B5589-90009)

Describes how to use NQS to submit, track, and control jobs for batch execution in queues.

SPP-UX System Administration Guide (B5655-90002)

Describes how to configure, control access to, and manage the file system of the SPP-UX operating system. Also describes the subcomplex manager, system accounting, and printer management.

FlexLM End User Manual (DSW-901)

Provides a licensing overview for system administrators and end users. Topics covered include the license file and license administration tools.

Ordering documentation

To order additional copies of this document or other documents listed in "Associated documentation," send requests to:

Hewlett-Packard Company
Convex Division
Customer Service
P.O. Box 833851
Richardson TX 75083-3851 USA

Please include the order number or the exact title of the document.

Technical assistance

If you have questions that are not answered in this book, contact the Hewlett-Packard Convex Technical Assistance Center (TAC) at the following locations:

Within the continental U.S., call 1 (800) 952-0379.

From Canada, call 1 (800) 345-2384.

All other locations, contact your local Hewlett-Packard office.

You can also use the `contact` utility, if you would like to report any problems you may have with NQS or its associated documentation.

Overview 1

This chapter introduces basic Network Queuing System (NQS) concepts and features, including:

- NQS basic processing
- Utilities available for configuring and controlling NQS
- Levels of authorization required to take advantage of these utilities
- Differences between NQS for Exemplar S-Class and X-Class Technical Servers and other NQS systems

Queue processing

NQS lets users submit jobs to a queue for batch execution.

A queue is a list of requests that are ready and waiting to run.

A request is one or more commands submitted by a user or a user program to a queue. These commands are usually run after a certain time or event has passed; they do not require further interaction with the user. A typical request is a program containing commands that perform large-scale, detailed computations on a static data file.

Users can submit requests to queues residing on either a local or remote machine configured with NQS for Exemplar Technical Servers or another version of NQS.

NQS queues

There are two types of NQS queues:

Batch

Batch queues run requests. They hold requests for scheduled, perhaps delayed, processing by subsystems within NQS. Demand queues are special batch queues that accept requests only if they can place the requests into immediate execution.

Pipe

Pipe queues are routing queues that do not directly run requests, but instead transmit requests to other queues. Each pipe queue has a pipe client that does the actual routing and a set of destination queues that are possible recipient queues. A destination queue can be either a batch queue, a demand queue, or another pipe queue on either a local or remote machine.

NQS pipe clients

There are three types of NQS pipe clients:

`pipeclient`

Routes a request to the first queue in its destination set that is able to accept the job. Destinations may reject the request due to queue limit violations or lack of account authorization, to name two possibilities.

`pipedemand`

Routes a request to the first queue in its destination set that can immediately run the request. A destination cannot immediately run a request if the number of jobs currently running at the destination matches its run limit. If a request cannot be routed to a destination, the request stays queued in the pipe queue until a destination becomes available. `pipedemand` minimizes the time a request spends in a queued state and maximizes job throughput in a cluster environment.

`pipeldav`

Sorts its destination set by load factor and routes a request to the destination with the lowest load factor able to accept the job. `pipeldav` places requests into queues quickly. It does not wait until a queue is empty or spend unnecessary time polling queues. Neither does `pipeldav` give all the jobs to a processor with a light load, because each job placed in a queue increases the load factor.

To route a request based on load factor, `pipeldav`:

- Scans the set of destination queues to determine associated host machines.
- Determines the host load average for each host machine using rstatd software loaded on the host machine. rstatd is an optional NFS product; contact your local sales representative for additional information.

- Determines the queue length for each destination queue by querying the netdaemon on the host machine.
- Calculates the load factor for each destination queue using the formula in Figure 1.

$$\text{load factor} = \frac{\text{host load average} + (\text{queue length} \times \text{weight})}{\text{processor speed}}$$

Figure 1 Load factor calculation

The NQS manager sets the weighting factor as an option of the `pipeldav` program; the weighting factor defaults to 1.0.

- Routes the request to the destination queue—with the lowest load factor—able to accept the job.

NQS configuration and control utilities

The following two NQS utilities configure and operate NQS:

- `qmapmgr`
- `qmgr`

These utilities are discussed in the following sections.

qmgr utility

`qmgr` is the NQS utility that controls queues and requests on the local machine.

You can use `qmgr` utility at any time to:

- Abort queues
- Create queues
- Configure queues
- Delete queues
- Enable and disable queues
- Move requests from one queue to another
- Reorder requests inside a queue
- Set queue attributes
- Show information about attributes, managers, and queues
- Start and stop queues
- Start and stop NQS

See "qmgr commands" on page 71 for detailed information on `qmgr`.

qmapmgr utility

`qmapmgr` is the NQS utility that builds and maintains a network database used to establish a mapping between NQS and each machine in the NQS configuration.

Without this mapping, NQS cannot recognize local and remote destinations and queues.

You can use `qmapmgr` to make changes to the network database on a local machine at any time; however, make sure all machines have the same network database.

See "qmapmgr commands" on page 143 for detailed information on `qmapmgr`.

Levels of authorization

`qmgr` commands available to each user depend on user type. NQS distinguishes the following user types:

General users

Can track and control their own requests.

Managers

Have access to all `qmgr` commands. See "qmgr commands" on page 71 for a complete list. By default, root is an NQS manager and can assign this privilege to other users.

Licensing

NQS is a per CPU licensed product. There are two types of licenses for Exemplar software products:

- Per CPU: License for use on a system with a predetermined number of CPUs.
- Per User: License for a specific number of simultaneous users. A "user" is a nonsystem, unique user id number (UID.) The largest license allows unlimited system access.

A paper with your license password accompanies the shipment of NQS software. In order to enable this software, you must add the license password to the following file:

/usr/local/flexlm/licenses/convex.dat

See the *FLEXlm End User Manual* that accompanies this distribution for more information.

Differences

NQS for Exemplar S-Class and X-Class Technical Servers differs from other versions of NQS in five major ways. Understanding the differences is important if you combine several systems at one site:

- The `move my_request` command does not exist in other NQS systems.
- NQS for Exemplar Technical Servers can mount remote files using NFS, other systems cannot.
- The `maximum_request_priority` command, if used when submitting a request between NQS for Exemplar Technical Servers and another NQS system, decreases the priority of previously submitted requests. It does not delete previously submitted requests.
- Direct submission (for example, `qsub -q long@host`) between machines running NQS for Exemplar Technical Servers and other machines is not possible.
- Several NQS for Exemplar Technical Servers packet numbers are not recognized by other NQS systems.

  Table 1 lists packet numbers recognized only by NQS.

Table 1 Packet numbers recognized only by NQS

Packet Number	Type	Use
200	nqs	Set queue import attribute
201	nqs	Set queue description
203	nqs	Set queue accounting on or off
204	nqs	Set accounting log file
206	nqs	Queue request from remote qsub
207	nqs	Get sequence number
208	nqs	Hold request
209	nqs	Release request
210	nqs	Add queue alias
211	nqs	Delete queue alias
212	nqs	Ping with ack (used for debugging)
213	nqs	Ping without ack (used for debugging)
214	nqs	Set maximum request priority
218	nqs	Force request to run
219	nqs	Suspend request
220	nqs	Resume request
223	nqs	Force run request
224	nqs	Set global per-user-run-limit
225	nqs	Set queue per-user-run-limit
226	nqs	Restart request
228	nqs	Set copy-open-files on checkpoint
205	net	Get remote queue and request information

((Configuring NQS) 2

[[NQS is suitable for most single-machine system configurations as it is shipped and installed; however, every site is different. You may want to:]

- [[[[[Create an optimal, efficient configuration that matches your work loads to available resources]]
- [[Configure NQS to handle site-specific situations]]
- [[Install NQS on multiple machines]]]]]]

((Summary of steps)()

[This chapter describes tasks you must perform to install and configure NQS. These tasks, in recommended order of performance, are:]

Step 1 [[[[[Install NQS.]]

Step 2 [[Take a `qmapmgr` snapshot.]]

Step 3 [[Assign managers using the `add manager` command.]]

Step 4 [[Determine queue structure.]]

Step 5 [[Create and configure batch queues.]]

Step 6 [[Create pipe queues.]]

Step 7 [[Delete queues.]]

Step 8 [[Configure and activate NQS accounting.]]

Step 9 [[Establish a shell strategy.]]

Step 10 [[Take a `qmgr` snapshot.]]

Step 11 [[Configure error logging.]]

Step 12 [[Automate queue operations.]]

Step 13 [[Notify users of changes.]]]]]]

((Getting started)()

[Several procedures in this chapter require viewing the attributes of queues. Use the `show long queue` command in the `[qmgr]` utility to accomplish this task. You must start the `qmgr` utility before you can use this command. To start `qmgr`, enter:]

`[[/opt/nqs/bin/qmgr]]`

[The command format is:]

`[[show long queue]` *[queuename]]*

[where *[queuename]* is the name of the queue.]

[The attributes for a pipe queue are a subset of the batch queue attributes. That is, a pipe queue has fewer attributes. The following example illustrates the output for this command when viewing the attributes for a batch queue:
]

```
[Mgr: show long queue s]

[s@hostC; type=BATCH; [ENABLED, INACTIVE]; pri=48
0 exit; 0 run; 0 stage; 0 queued; 0 wait; 0 hold; 0 arrive;
Run_limit = 2; Per-user run-limit = NONE
Accounting: Off
Activity ID offset: 1
Maximum request priority: 63
Cumulative system space time = 0.000000 seconds
Cumulative user space time = 0.000000 seconds
Unrestricted access
Import directory: Yes
Shell execution as login: No
Per-process permanent file size limit = UNLIMITED
Per-process execution nice value = 0 <DEFAULT> ]
```

[The first line of the output describes information about the queue.]

`[[s@hostC; type=BATCH; [ENABLED, INACTIVE]; pri=48]]`

`[[[s@hostC]]`

[[Is the name of this queue and the machine where it is configured. In this example, the attributes shown apply to the *s* queue on *hostC*.]]

`[[type=BATCH]]`

[[Is the type of queue where]

`[[[BATCH]]`

[[Runs NQS requests.]]

[[PIPE]]

[[Routes NQS requests to queues that can run them.]]]]

[[ENABLED]]

[[Indicates if this queue can accept requests. Where:]

[[[ENABLED]]

[[NQS is running on the local machine, and the queue is accepting requests.]]

[[DISABLED]]

[[NQS is running on the local machine, but the queue is not accepting requests.]]

[[CLOSED]]

[[NQS is not running on the local machine.]]]]

[[INACTIVE]]

[[Indicates if this queue can run requests. Where:]

[[[INACTIVE]]

[[Requests in the queue can run; none are running.]]

[[RUNNING]]

[[Requests in the queue can run; some are running.]]

[[STOPPING]]

[[New requests sent to the queue cannot run, but requests that are currently running can complete.]]

[[STOPPED]]

[[Requests in the queue cannot run; none are running.]]

[[SHUTDOWN]]

[[NQS is not running on the local machine.]]]]

[[pri=48]]

[[Indicates interqueue priority. This priority affects queue selection for running the next job. Priority can be any number from 0 to 63; 0 is the lowest priority and 63 the highest.]]]

[The second line of the output tallies the number of requests in specific states:]

```
[[0 exit; 0 run; 0 stage; 0 queued; 0 wait; 0 hold; 0 arrive;]]
```

[[[0 exit]]

[[Number of requests in this queue in an exiting state.]]

[[0 run]]

[[Number of requests in this queue in a running state.]]

[[0 stage]]

[[Number of requests in this queue in a staged state, which means the request has completed running and NQS is moving the stdout and stderr files to the appropriate destination directory.]]

[[0 queued]]

[[Number of requests in this queue in a state of being queued.]]

[[0 wait]]

[[Number of requests in this queue in a waiting state.]]

[[0 hold]]

[[Number of requests in this queue in a holding state.]]

[[0 arrive]]

[[Number of requests in this queue in an arriving state.]]]

[The following lines show miscellaneous information about the queue:]

```
[Run_limit = 2; Per-user run-limit = NONE
Accounting: Off
Maximum request priority: 63
Cumulative system space time = 0.000000 seconds
Cumulative user space time = 0.000000 seconds
Unrestricted access
Import directory: Yes]
```

[[[Run_limit]]

[[Maximum number of requests that can run in this queue at any one time. When the limit is exceeded, NQS queues requests until the number of jobs running is less than the limit (applies to batch queues only).]]

[[Per-user run-limit]]

[[Number of requests a user may run in this queue at any one time. Per-user run limits apply after per-queue run limits (applies to batch queues only).]]

[[Accounting]]

[[Indicates if batch accounting is activated (applies to batch queues only).]]

[[Maximum request priority]]

[[Maximum priority of a request that can be submitted to the queue.]]

[[Cumulative system space time]]

[[[Batch queues]

[[Total amount of system time used to process requests in this batch queue, since the queue was created.]]

[Pipe queues]

[[Total amount of system time used to route requests in this pipe queue, since the queue was created.]]]]

[[Cumulative user space time]]

[[Total amount of user time used to process requests in this queue, since the queue was created.]]

[[Unrestricted access]]

[[Indicates the access restrictions placed on this queue. These restrictions do not apply to a request submitted by the superuser; superuser requests are always queued. Access restrictions for a queue can be:]

[[[Unrestricted]]

[[Accepts any request from any submitter.]]

[[Restricted]]

[[Accepts only those requests submitted by a specified group or user.]]

[[Pipeonly]]

[[Accepts requests only from a pipe queue.]]]]

[[Import directory]]

[[Indicates if this queue imports the current working directory for a request (mounts the directory on the machine processing the

request) before running the request (applies to batch queues only). This can be:]

[[`[Yes]`]

[[This queue automatically imports the current working directory for any request running in the queue. A user can override this setting for individual requests using the `-ni` option of `qsub`.]]

[`[Available]`]

[[Lets a user specify importation of the current working directory for any request submitted to this queue using the -i option of `qsub`.]]

`[No]`

[[This queue does not import the current working directory for requests submitted to the queue. Furthermore, it rejects requests that require imported directories.]]]]

`[[Per-process permanent file size limit]]`

[[The maximum permanent file size for any process in a running request. If any process tries to create a permanent file larger than the limit set, NQS sends a SIGXFSZ signal to the offending process. If the process does not catch the signal or ignores the signal, the process exits.]]

`[[Per-process execution nice value]]`

[[The minimum nice value for any process in a running request. The nice value determines the proportion of CPU time allocated to a process, relative to all other processes in the system. The lower the nice value assigned to a process, the higher the proportion of CPU time allocated to it. A practical range is from -20 to 20.]]]

[The rest of the output displays the per-process resource limits (if any) configured for this queue. SPP-UX enforces only the per-process permanent file size limit and the per-process execution nice value.]

[If a user submits a request with limit specifications to a queue, NQS checks the request limits to ensure they do not exceed the enforceable limits set for the queue. If the request limits exceed the queue limits, NQS rejects the request.]

[If a user submits a request without limit specifications to a queue, NQS uses the enforceable limits set for the queue as default limits.]

[NQS assigns limits to a request when the request is queued. If a queue limit is changed after a request is queued, the queued

request is not affected. However, if the previously-queued request exceeds the new queue limit, NQS displays a warning message.]]

((Installing NQS)

[Perform the following steps to[] install NQS:]

Step 1 [[[[[Determine which machines will run NQS.]]

Step 2 [[Install NQS on the desired machines. To install NQS on each machine, follow the steps listed in the *[Distribution Notice]*that accompanies this release.]]

Step 3 [[Log in to the machine you are configuring as the superuser.]]

Step 4 [[Check to see if nameserver is running.]]

Step 5 [[Enter the following command at the system prompt:]

`[/opt/nqs/bin/[qmapmgr]]`

[The `[qmapmgr]` prompt appears:]

`[[Mapmgr:]]]`

Step 6 [[Create the nmap database on the local machine by entering:]

`[[create]]]`

Step 7 [[Add the MID for each host to the nmap database using the `add mid` command. Enter a separate command for each machine that will run NQS. The command format is:]

`[[add mid]` *[n machine_name]]*

[where]

[[n]]

[[Is a unique number identifying the machine that will run NQS.]]

[[machine_name]]

[[Is the name of the machine that will run NQS. This name should correspond to an entry in the /etc/hosts file and should not be an alias.]]]

[The following example illustrates the use of the `create` and `add` commands to create the nmap database and assign MIDs to *[hostA]*, *[hostB]*, and *[hostC].*]

```
[# qmapmgr
Mapmgr: create
nmap_success: successful completion.
Mapmgr: add mid 1 hostA
nmap_success: successful completion.
Mapmgr: add mid 2 hostB
nmap_success: successful completion.
Mapmgr: add mid 3 hostC
nmap_success: successful completion.
]]
```

Step 8 [[If nameserver is not running, skip this step.]

[If nameserver is running, supply the fully qualified host name as an alias for each machine you are adding using the `[add name]` command. For example, if the fully qualified name for *[hostA]* is hostA.berkley.edu, enter:]

```
[[add name hostA.berkley.edu 1]]
```

[where 1 is the MID assigned to *[hostA].*]

[You can check to see if your system is running named by using one of the following methods:]

- [[[[[Check for the file /etc/resolv.conf]]
- [[Enter:]

  ```
  [[% ps -eadf | grep named]]]]]]
  ```

Step 9 [[Exit `[qmapmgr.]` Enter:]

```
[[exit]]]
```

Step 10 [[Repeat these steps on each machine that will run NQS.]

[Make sure you enter the same MID/machine name combinations at each machine. That is, *[hostA]* in the sample configuration must be MID 1 on all machines.]]]]]]

((Taking a qmapmgr snapshot)()

[After installing NQS on each machine, save the current NQS configuration as a series of `qmapmgr` commands using the `qsnapshot [-m]` command.]

[This command saves the configuration to screen by default. However, you can save the configuration to a file and later use the contents of this file to restore the NQS `[qmapmgr]` configuration. For example, to save the configuration to a file named batch_nconfig, enter:]

```
[[qsnapshot -m > batch_nconfig]]
```

[The following example illustrates output from the `qsnapshot [-m]` command:]

```
[# qsnapshot -m
CREATE
ADD MID 1 hostA
ADD MID 2 hostB
ADD MID 3 hostC]
```

[For more information, refer to the qsnapshot(8), qmgr(8), and qmapmgr(8) man pages]]

((Assigning managers)()

[After taking a snapshot of the NQS `[qmapmgr]` configuration on each machine, set up NQS managers.]

[An NQS manager has access to all NQS commands. `[ qmgr]`'s `[add manager]` command grants users access to all commands defined by NQS as manager commands.]

[[Perform the following steps to grant NQS manager privileges using the `add manager` command:]

Step 1 [[[[[Log in to the machine you are configuring as the superuser.]]

Step 2 [[Start the `qmgr` utility. Enter:]

`[[/opt/nqs/bin/qmgr]]`

[The `[qmgr]` prompt appears:]

`[Mgr:]]`

Step 3 [[Add NQS managers using the `[add manager]` command. The command format is:]

`[` **`[add manager]`** *[user_name]* **`[:m]`** `]`

[where]

[[*[user_name]*]

[[Is the name of the user who is to have manager access.]]

`[[m]]`

[[Indicates manager access.]]]

[The following example illustrates use of this command to grant manager privileges to the user batchman:]

```
[# qmgr
Mgr: show managers
root:m
Mgr: add manager batchman:m
ConvexNQS+ manager[TCML_COMPLETE]:transaction
complete at local host]]
```

Step 4 [[Repeat these steps on each machine running NQS.]]

Step 5 [[Exit `[qmapmgr]`. Enter:]

`[[exit]]]]]]]`

((Determining queue structure)()

[After assigning NQS manager privileges, you must determine the queue structure needed to satisfy user requirements. Do this for each machine configured with NQS.]

[There are two types of NQS queues:]

[[Batch]

[[Batch queues run requests. They hold requests for scheduled, perhaps delayed, processing by subsystems within NQS. Demand queues are special batch queues that accept requests only if they can place the requests into immediate execution.]]

[Pipe]

[[Pipe queues are routing queues that do not directly run requests, but instead transmit requests to other queues. Each pipe queue has a pipe client that does the actual routing and set of destination queues that are possible recipient queues. A destination queue can be a batch queue, a demand queue, or another pipe queue on either a local or remote machine.]]]

[Assume a sample configuration with four machines: *[hostA]*, *[hostB]*, *[hostC]*, and *[hostD]*. Your intent:]

- [[[[[Use *[hostA]* primarily for electronic mail, word processing, spreadsheet, and other applications requiring reasonable interactive response time.]]
- [[Use *[hostB]*, *[hostC]*, and *[hostD]* primarily for number-crunching, resource-intensive applications.]]]]]

[Also assume two user groups in this sample configuration: a development group and a design group. Some users may belong to both groups. Each user in each group has an account on each machine, and the login names and UIDs are the same across machines (that is, no account mapping is required).]

[Recommendation:]

- [[[[[Create three demand queues—one on *[hostB]*, one on *[hostC,]* and one on *[hostD].*]]
- [[Create a pipe queue on *[hostA]*, from which the demand queues can *pull* resource-intensive jobs.]]]]]

[[Figure 2] illustrates this configuration.]

[[[]]

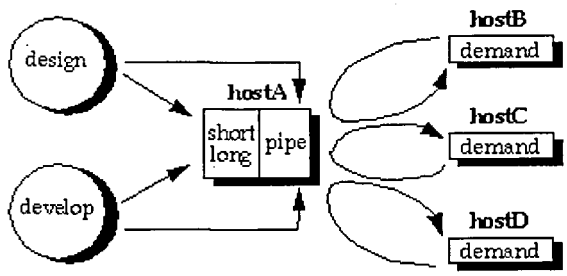


Figure 2 [Sample queue configuration]]

[All users can submit jobs requiring little or moderate resource usage to the default batch queues on *[hostA].*]

[Design users can submit long-running jobs to the pipe queue on *[hostA].* Whenever a demand queue on *[hostB], [hostC,]* or *[hostD]* becomes idle, it pulls a queued job from the pipe queue and runs it.]

[Development users can also submit long-running jobs to the pipe queue on *[hostA].* Whenever a demand queue on *[hostB], [hostC],* or *[hostD]* becomes idle, it pulls a queued job from the pipe queue and runs it.]

[This sample configuration minimizes the time a job spends in a queued state, maximize job throughput in a cluster environment, and prevents one group from monopolizing the other group's resources.]

[While this sample configuration is not designed to cover every situation you may encounter, it covers the main issues you must consider when establishing your system.]]

((Creating batch queues)()

[Once you determine the queue structure needed to satisfy user requirements, you may need to add batch queues to all or some of the machines configured with NQS.]

[Perform the following steps to add batch queues:]

Step 1 [[[[[Log in to the machine you are configuring as an NQS manager.]]

Step 2 [[Start the `[qmgr]` utility. Enter:]

`[[/opt/nqs/bin/qmgr]]`

[The `[qmgr]` prompt appears:]

`[[Mgr:]]]`

Step 3 [[Add a batch queue using the `[create] [batch_queue]` command.]

[For example, to add a new batch queue named *b* with an interqueue priority of 48, enter:]

`[[create batch_queue b pr=48]]`

[The command format is:)

`[[create batch_queue ]`*[queuename]*`[ priority=]`*[priority]*`[ [pipeonly] [import_dir=]`*[import-option]*`[] [run_limit=]`*[run-limit]*`[] [demand] [sender=]`*[sender]*`[] [SUbcomplex = ]`*[subcomplex name]*`]]`

[where]

[[[queuename]]

[[Is the name of the queue. The name can consist of any printable nonblank character except for the at sign (@), a comma (,), an equal sign (=), and a left or right parenthesis ( ). The name cannot start with a digit.]]

[[priority]]

[[Is the interqueue priority for the queue. This priority affects queue selection for running the next job. It can be any number from 0 to 63; 0 is the lowest priority and 63 the highest.]]

`[[pipeonly]]`

[[Indicates the queue can only accept requests submitted from a pipe queue. Otherwise, the queue can accept requests from any source (such as a program or script file).]]

[[import_option]]

[[Indicates if NQS imports the current working directory for a request (mounts the directory on the machine processing the request) before running the request. This can be:]

`[[[Yes]]`

[[NQS automatically imports the current working directory for any request running in the queue. You can override this setting for individual requests using the `-ni` option of `qsub`.]]

`[[Available]]`

[[Lets a user specify importation of the current working directory for requests submitted to the queue using the `[-i]` option of `[qsub]`.]]

`[[No]]`

[[NQS does not import the current working directory for requests. Furthermore, the queue rejects requests requiring imported directories.]]]

[If you specify `[ Yes]` or `[Available]`, NQS imports directories with NFS by making temporary mount points in the /tmp directory of the originating machine. Be aware of local, automatic clean-up facilities of /tmp that could change these NFS mount points.]]

[*[run_limit]*]

[[Is the maximum number of requests that can run in the queue at one time. For example, if you submit four requests to a queue whose run limit is set to two, two of the requests begin running and two are queued. If you do not supply a run limit, the value defaults to one.]]

`[[demand]]`

[[Indicates that requests may enter this queue only if they can execute immediately (which is determined by *[run-limit]*). `[Demand]` implies `[pipeonly].`]]

[*[sender]*]

[[Is the name of the pipe queue(s) that can send requests to this batch queue, and applies only if *`[demand]`* is specified. If you supply more than one sender, separate items in the list with commas and surround the list with parentheses. For example, to designate three destinations, enter:]

`[[(sendq1, sendq2, sendq3)]]`

[The syntax of the sender queue name must be in one of the following forms:]

[[local_queue_name]]

[[local_queue_name@local_machine_name]]

[[remote_queue_name@remote_machine_name]]

[[remote_queue_name@[remote_machine_mid]]]

[Square brackets [ ] are required where shown.]

Caution(()

((Do not define any sender without a `[server=(/opt/nqs/lib/pipedemand])` attribute.))))

`[[SUbcomplex] =` *[subcomplex name]]*

[[Causes all requests running from this queue to execute on the named subcomplex. The subcomplex must be a valid and currently loaded subcomplex when the queue is created and started. A subcomplex can be configured using `[scm]` or by using the following command:]

[[`qmgr COnfig_subcomplex`]]

`[[COnfig_subcomplex]` executes the `[scm]` utility.]

[Subcomplex permissions should be set to reflect the user and group set intended for use on a subcomplex batch queue.]]]]

Step 4 [[Repeat [Step 3] for each added batch queue on this machine.]]

Step 5 [[Exit `[qmgr]`. Enter:]

`[[exit]]]`

Step 6 [[Repeat these steps on each machine running NQS.]]]]]]

((Configuring batch queues)()

[Once you add any needed batch queues, you must configure them. You must also configure all default batch queues on all machines configured with NQS.]

[Perform the following steps to configure batch queues:]

Step 1 [[[[[Log in to the machine you are configuring as an NQS manager.]]

Step 2 [[Start the `[qmgr]` utility. Enter:]

```
[[/opt/nqs/bin/qmgr]]
```

[The `[qmgr]` prompt appears:]

```
[Mgr:]]
```

Step 3 [[Set the per-user run limit for the queue using the `[set] [per_user] [run_limit]` command. The command format is:]

`[[set per_user run_limit]` = *[run-limit queuename]]*

[where]

[[[run-limit]]

[[Is the number of requests a user can run in a queue at one time. To turn off per-user run limit, set this value to 0. NQS applies per-user run limits after applying per-queue run limits.]]

[[queuename]]

[[Is the name of the queue for which you are setting the limit.]]])

Step 4 [[Set the global per-user run limit for all queues using the `[set] [global per_user run_limit]` command. The command format is:)

`[[set global per_user run_limit]` = *[run-limit]]*

*[*where *[run-limit]* is the number of requests a user can run in all NQS queues at one time.*]*

[[Table 2] shows how per-user, per-queue, and global run limits work together. Requests with an asterisk begin running; requests without an asterisk are queued.]

[[]]

Table 2 [Per-user, per-queue, and global run limits]

(Limits and requests)	(S queue)	(L queue)	(All queues)
[Per-user run limit]	[2]	[2]	[]
[Per-queue run limit]	[4]	[4]	[]
[Global per-user run limit]	[]	[]	[3]
[Requests]	[johndoe*]	[root*]	
	[johndoe*]	[janedoe*]	
	[johndoe]	[janedoe*]	
	[root*]	[janedoe]	
	[root*]	[root]	
	[janedoe]	[johndoe*]	
	[janedoe]	[]	

Step 5 [[The maximum permanent file size controls the file size for any process in a running request. When a request is submitted to a queue, NQS checks the request limits to ensure the maximum permanent file size limit for the request does not exceed the maximum permanent file size limit for the queue. If it does exceed the maximum limit, NQS rejects the request.]

[You must set the maximum permanent file size for any batch queue residing on an Exemplar S-Class and X-Class Technical Servers. The recommended setting is 4194303 b.]

[Set the maximum permanent file size limit using the `[set per_process permfile_limit ]` command. The command format is:]

`[[set per_process permfile_limit]` = *[(limit) queuename]]*

[where]

[[[limit[units]]]]

[[Is the maximum size a permanent file can be for any process in the running request in *[limit [units]]* format. The recommended setting is 4194303 b. *[limit]* can be any integer up to 8 digits. You can specify that no limit be applied by

entering *[*`unlimited`*]*. If you omit *[units]*, bytes is assumed. *[units]* can be one of the following:]

[[`[b]`]

`[[Bytes]]`

`[w]`

[[Words]]

`[kb]`

[[Kilobytes ($2^{[10]}$ bytes)]]

`[kw]`

[[Kilowords ($2^{[10]}$ words)]]

`[mb]`

[[Megabytes ($2^{[20]}$ bytes)]]

`[mw]`

[[Megawords ($2^{[20]}$ words)]]

`[gb]`

[[Gigabytes ($2^{[30]}$ bytes)]]

`[gw]`

[[Gigawords ($2^{[30]}$ words)]]]]

[[queuename]]

[[Is the name of the queue for which you are setting the limit.]]]]

Step 6 [[NQS generates an error if the command in [Step 7] is applied to a batch queue residing on your Exemplar Technical Server.]]

Step 7 [[If you want to restrict queue access, use the `[set no_access]` command. The command format is:]

`[[set no_access]` *[queuename]]*

[where *[queuename]* is the name of the queue for which you are restricting access.]]

Step 8 [[If you set the access restriction parameter to no access in [Step 7], supply a list of users who may access the queue. Use the `[add user]` command. The command format is:]

`[[add users] =` *[user queuename]]*

[where]

[[[user]]

[[Is one or more users who may access the queue. If you supply more than one user, separate items in the list with commas and surround the list with parentheses.]

[[user] can be either the user name or UID. Enclose a UID in square brackets [ ].]]

[[queuename]]

[[Is the name of the queue for which you are restricting access.]]])

Step 9 [[Verify that all attributes for all queues have been correctly set. Use the `[show long queue]` command. The command format is:)

`[[show long queue]` *[queuename]]*

[For example, to show the attributes for the queue named *b*, enter:]

`[[show long queue b]]`

[The following example illustrates the output for this command:]

```
[Mgr: show long queue b
b@hostA; type=BATCH; [ENABLED, INACTIVE]; pri=48
 0 exit;   0 run;   0 stage;   0 queued;   0 wait; 0 hold;  0 arrive;
Run_limit = 1; Per-user run-limit = NONE
Accounting: Off
Activity ID offset: 0
Maximum request priority: 63
Cumulative system space time = 0.000000 seconds
Cumulative user space time = 0.000000 seconds
Unrestricted access
Import directory: Not available
Checkpoint: Not available
Copy open files on checkpoint: No
Share policy fixed = short
Per-process core file size limit = UNLIMITED
Per-process data size limit = UNLIMITED
Per-process permanent file size limit = UNLIMITED
Per-process execution nice value = 0 <DEFAULT> ]
```

[Correct any incorrect attributes using the set commands. See ["qmgr commands" on page 71] for a list of these commands and their formats.])

Step 10 [[Exit `[qmgr]`. Enter:]

`[[exit]]]`

Step 11 [[If you configured any queues to import the current working directory for a request, edit the /etc/exports file on each machine to include entries for all file systems eligible for remote mounting.]

[For example, to specify exportation of the /usr and /mnt file systems on *[hostA]* to *hostB* and *[hostC]*, include the following lines in the /etc/exports file on *hostA*:]

```
[[/usr -access=hostB:hostC]]

[[/mnt -access=hostB:hostC]]
```

[If you are using a nameserver, you must supply the fully qualified machine name. Refer to the exports(5) man page for more information on the format of this file.]]

Step 12 [[After editing the exports file, initialize the list of exportable file systems using `[exportfs]`. From the shell prompt, enter:]

```
[[exportfs -a]]]
```

Step 13 [[Edit the /etc/hosts file on the local machine to include any remote machine names you are exporting from.]

[If you are using TCP/IP protocol, the following example illustrates an /etc/hosts file entry on *[hostB]*:]

```
[130.168.71.160    hostA        # any comment
130.168.71.162    hostB        # any comment]
```

[Each line in this file represents one entry; each entry represents one machine. The format of the /etc/hosts file is:]

[[internet_address official_name [aliases...] [#comment]]]

[where]

[[[internet_address]]

[[Is the official internet address for this machine.]]

[[official_name]]

[[Is the official name for this machine, as specified with the `hostname` program.]]

[[aliases]]

[[Is an unofficial name or list of unofficial names for this machine.]]

[[#comment]]

[[Is any comment you want about this machine.]]]

[If you are not using TCP/IP protocol, you must include an entry in this file for the local machine. The following example illustrates an /etc/hosts file entry on *[hostA]*:]

```
[130.168.71.160    hostA    localhost
130.168.71.162    hostB    #any comment]]
```

Step 14 [[If you want to restrict access to file systems on a remote machine, skip to [Step 15].]

[Edit the /etc/hosts.equiv file on the remote machine to include the names of the machines that import the file systems. This makes it possible for the user to log in without further password validation, as long as the user has an account on the remote machine.]

[Each line in this file represents one entry. Each entry represents one remote machine. The following example illustrates an /etc/hosts.equiv file on *[hostC]*:]

```
[hostA
hostB]]
```

Step 15 [[If you want to restrict access to file systems on a remote machine, tell those users to edit their .rhosts file on the remote machine to include the names of the hosts that import the file systems.]

[The .rhosts file has the same format as the /etc/hosts.equiv file described in [Step 14]. Refer to the rhost(5) man page for more details.]]

Step 16 [[During processing, NQS holds job output and transaction scripts in several directories in /var/spool/nqs before sending job output to the submitter's output file. If /var/spool/nqs becomes full, the jobs will fail.]

[Consider creating a separate disk partition for /var/spool/nqs. Refer to the *[SPP-UX System Administration Guide: S-Class Servers]* for details on setting up partitions.]]

Step 17 [[Start the `[qmgr]` utility. Enter:]

```
[[/opt/nqs/bin/qmgr]]]
```

Step 18 [[Enable all batch queues using the `[enable queue]` command. For example, to enable the batch queue named *[b]*, enter:]

```
[[enable queue b]]]
```

Step 19 [[Start all batch queues using the `[start queue]` command. For example, to start the batch queue named *[b]*, enter:]

```
[[start queue b]]]
```

Step 20 [[Exit `[qmgr]`. Enter:]

```
[[exit]]]
```

Step 21 [[Repeat these steps on each machine running NQS.]]]]]]

((Creating pipe queues)()

[Once you determine the queue structure needed to satisfy user requirements, you must add pipe queues to machines configured with NQS.]

[Perform the following steps to add pipe queues:]

Step 1 [[[[[Log in to machine you are configuring as an NQS manager.]]

Step 2 [[Decide if the new pipe queue routes jobs to the first destination that accepts the request, to the first destination that can run the job immediately, or to the destination with the lowest load factor that is able to accept the request.]]

Step 3 [[Start the `qmgr` utility. Enter:]

```
[[/opt/nqs/bin/qmgr]]
```

[The `[qmgr]` prompt appears:]

`[[Mgr]:]]`

Step 4 [[Create a pipe queue using the `[create pipe_queue]` command in `[qmgr]`. The command format is:)

`[[create pipe_queue]` *[queuename]*
`[priority]=`*[priority]* `[server]=(`*[server]*`)`
`[[destination]=`*[destination]*`]` `[[pipeonly]]`
`[[run_limit]=`*[run-limit]*`]]`

[where]

[[[queuename]]

[[Is the name of the queue. The name can consist of any printable nonblank character except for the at sign (@), a comma (,), an equal sign (=), and a left or right parenthesis (). The name cannot start with a digit.]]

[[priority]]

[[Is the interqueue priority for the queue. This priority affects queue selection for running the next job. This can be any number from 0 to 63; 0 is the lowest priority and 63 the highest.]]

[[server]]

[[Is the name of the pipe client that transports submitted requests to one of the destination queues. This can be:]

`[[[pipeclient]]`

[[Routes the request to the first destination that will accept it. Destinations may reject a request due to queue limit violations or lack of account authorization. The full path name for this pipe client is /opt/nqs/lib/pipeclient.]]

[[pipedemand]]

[[Routes the request to the first destination that can run the job immediately. The full path name for this pipe client is /opt/nqs/lib/pipedemand.]]

[[pipeldav]]

[[Sorts the destination list by load factor and tries destinations with low load factors first. The full path name for this pipe client is /opt/nqs/lib/pipeldav.]]]

[Refer to the pipeclient(8) man page for more details.]]

[[destination]]

[[Is the name of destination queue(s) to which this pipe queue can route requests. If you supply more than one destination, separate items in the list with commas and surround the list with parentheses. For example, to designate three destinations, enter:]

[[(destq1, destq2, destq3)]]

[The syntax of the destination queue name must be in one of the following forms:]

[[local_queue_name]]

[[local_queue_name@local_machine_name]]

[[remote_queue_name@remote_machine_name]]

[[remote_queue_name@[remote_machine_mid]]]

*[*Square brackets **[ ]** are required where shown.*]*

Caution(()

((Do not specify a pipedemand pipe client without making sure all destination queues are defined with a demand attribute.))))

[[pipeonly]]

[[Indicates the queue can only accept requests from a pipe queue. Otherwise, the queue can accept requests from any source.]]

[[run_limit]]

[[Is the maximum number of pipe clients that may run simultaneously to deliver requests to their destination.]]]]

Step 5 [[Set the maximum permanent file size limit using the set `per_process permfile_limit` command. **)**

[The maximum permanent file size controls the file size for any process in a running request. When a request is submitted to a queue, NQS checks the request limits to ensure the maximum permanent file size limit for the request does not exceed the maximum permanent file size limit for the queue. If it does exceed the maximum limit, NQS rejects the request.]

[You must set the []maximum permanent file size for any batch queue residing on an Exemplar Technical Server. The recommended setting is 4194303 b. The command format is:]

`[[set per_process permfile_limit] =` *[(limit) queuename]]*

[where]

[[[limit]]

[[Is the maximum size of a permanent file for any process in the running request in*[limit [units]]* format. The recommended setting is 4194303 b. *limit* can be any integer up to 8 digits. You can specify that no limit be applied by entering **`[unlimited]`**. If you omit *[units]*, bytes is assumed. *[units]* can be any one of the following:]

[[b]

`[[Bytes]]`

`[w]`

[[Words]]

`[kb]`

[[Kilobytes ($2^{[10]}$ bytes)]]

`[kw]`

[[Kilowords ($2^{[10]}$ words)]]

`[mb]`

[[Megabytes ($2^{[20]}$ bytes)]]

`[mw]`

[[Megawords ($2^{[20]}$ words)]]

`[gb]`

[[Gigabytes ($2^{[30]}$ bytes)]]

`[gw]`

[[Gigawords ($2^{[30]}$ words)]]]]

[[queuename]]

[[Is the name of the queue for which you are setting the limit.]]]]

Step 6 [[If you want to restrict queue access, use the `[set no_access]` command. The command format is:]

`[[[set no_access]` *[queuename]]*

[[where *[queuename]* is the name of the queue for which you are restricting access.]]]]

Step 7 [[If you set the access restriction parameter to no access in [Step 6], supply a list of users who may access the queue. Use the `[add user]` command. The command format is:]]

`[[add users] =` *[user queuename]]*

[where]

[[[user]]

[[Is one or more users who may access the queue. If you supply more than one user, separate items in the list with commas and surround the list with parentheses. *user* can be either the user name or UID. Enclose a UID in square brackets [].]]

[[queuename]]

[[Is the name of the queue for which you are restricting access.]]]]

Step 8 [[If you set the access restriction parameter to no access in [Step 6], supply a list of groups who may access the queue. Use the `add group` command. The command format is:]]

[`[add group] =` *[group queuename]]*

[where]

[[[group]]

[[Is one or more groups who may access the queue. If you supply more than one group, separate items in the list with commas and surround the list with parentheses. *[group]* can be either the group name or GID. Enclose a GID in square brackets [].]]

[[queuename]]

[[Is the name of the queue for which you are restricting access.]]]]

Step 9 [[Verify that all attributes for the queue have been correctly set. Use the `show long queue` command. The command format is:]]

`[[show long queue]` *[queuename]]*

[For example, to show the attributes for the queue named *best*, enter:]

```
[[show long queue best]]
```

[The following example illustrates the output for this command:]

```
[Mgr: show long queue best

best@hostA; type=PIPE; [ENABLED, INACTIVE]; pri=48
0 depart;   0 route;   0 queued;   0 wait;   0 hold;   0 arrive;
Run_limit = 1;   Per-user run limit = NONE
Cumulative system space time = 0.000000 seconds
Cumulative user space time = 0.000000 seconds
Unrestricted access
Queue server: /opt/nqs/lib/pipeldav -w 1.0 hostC 2.0
Destset = [v@hostA,v@hostB,v@hostC]]
```

[Correct any incorrect attributes using the `[set]` commands. Refer to ["qmgr commands" on page 71] for a complete list of these commands and their formats.]]

Step 10 [[Enable the queue using the `enable queue` command. For example, to enable the pipe queue named *[best]*, enter:]

```
[[enable queue best]]]
```

Step 11 [[Start the queue using the `start queue` command. For example, to start the pipe queue named *[best]*, enter:]

```
[[start queue best]]]
```

Step 12 [[Repeat [Step 4] through [Step 11] for each pipe queue on this machine.]]

Step 13 [[Exit `[qmgr]`. Enter:]

```
[[exit]]]
```

Step 14 [[Repeat these steps on each machine running NQS.]]]]]]

((Deleting queues)()

[If you create a queue you no longer need, perform the following steps to delete it. The queue must be empty before you can delete it.]

Step 1 [[[[[Log in to machine you are configuring as an NQS manager.]]

Step 2 [[Start the `qmgr` utility. Enter:]

```
[[/opt/nqs/bin/qmgr]]
```

[The `[qmgr]` prompt appears:]

```
[[Mgr]:]]
```

Step 3 [[You must disable the queue before deleting it. The command format is:)

`[[disable q]` *[queuename]]*

[where *[queuename]* is the name of the queue you want to disable.]]

Step 4 [[You must stop the queue before deleting it. The command format is:)

`[[[stop q]` *[queuename]]*

[[where *[queuename]* is the name of the queue you want to stop.]]])

Step 5 [[Delete the queue using the `[delete] queue` command. The command format is:)

`[[[delete q]` *[queuename]]*

[[where *[queuename]* is the name of the queue you want to delete.]]])

Step 6 [[Exit `[qmgr]`. Enter:]

```
[[exit]]]
```

Step 7 [[Repeat these steps on each machine running NQS.]]]]]]

((Configuring and activating NQS accounting))()

[The NQS batch accounting system tracks the system resources used by an individual user or group by request.]

[It collects information according to the structure defined in the /usr/include/batch-acct.h file. The following example illustrates the NQS accounting structure:]

```
[struct batch_acct {
     char quename[MAX_QUEUENAME+1];  /* the name of the batch queue*/
     char host[MAXHOSTNAMELEN+1];    /* the host where the job ran */
     time_t submit_time;             /* when the job was submitted */
     time_t complete_time;           /* when the job completed */
     uid_t uid;                      /* user's uid (see getuid(2)) */
     gid_t gid;                      /* user's gid (see getgid(2)) */
     long aid;                       /* activity id (see getaid(2))*/
     long seqno;                     /* job sequence number */
     char rhost[MAXHOSTNAMELEN+1];   /* originating host name */
     short rpriority;                /* request (intra-queue) priority*/
     short qpriority;                /* queue (inter-queue) priority */
     short nice;                     /* nice value */
     struct rusage rusage;           /* resource usage */
     time_t start_time;              /* when the job started */
     int reserved[7];                /* reserved for future use */
};]
```

[Once the accounting information is collected, the system manager can use the `qsa` utility to interpret it. See ["Generating accounting reports" on page 67] for details on using `[qsa].`]

[Perform the following steps to configure and activate the NQS accounting system:]

Step 1 [[[[[Log in to machine you are configuring as an NQS manager.]]

Step 2 [[Start the `[qmgr]` utility. Enter:]

`[[/opt/nqs/bin/qmgr]]`

[The *[qmgr]* prompt appears:]

`[[Mgr:]]]`

Step 3 [[Identify the log file where accounting data is collected on this machine using the `[set acc_logfile]` command. For example, to identify a log file named /usr/adm/batchacct, enter:]

`[[set acc_logfile /usr/adm/batchacct]]]`

Step 4 [[Using the `[show parameters]` command, display queue parameters to verify the change. The following example illustrates use of this command to check the accounting log file:)

`[Mgr:` **`show parameters`**

Accounting log file = /usr/adm/batchacct]]

Step 5 [[Activate NQS accounting for this queue using the [set accounting] command.]

[When accounting is activated for a queue, NQS saves batch job information in an accounting log file (if one is specified.) If requested with the [qsub -me] command, NQS sends mail to the user detailing what resources were used. For example, to activate accounting for the [s] queue, enter:]

[[set accounting=on s]]]

Step 6 [[Determine the per-queue shell execution resource using the set [exec_shell_login] command.]

[If your jobs require customized resources, you may want to set this attribute to [Yes]. This causes any request submitted to this queue to be executed under your login shell, by sourcing your .cshrc and .login files. If you do not want the jobs to use your login shell, submit the job with [qsub -nl].]

[If your jobs normally do not require customized resources, but you may need them, set this attribute to [Available]. This causes any request sent to this queue to *not* be executed under your login shell, unless submitted with [qsub -l].]

[If your jobs normally do not require customized resources and you do not wish to allow this, set this attribute to [No]. None of the requests will be executed under a login shell.]

[The command format is:]

[[set exec_shell_login]=*[Answer queuename]]*

[where]

[[[Answer]]

[[Can be one of the following:]

[[Yes]]

[[No]]

[[Available]]]

[[queuename]]

[[Is the name of the queue you are configuring.]]]

[For example, to set the attribute to allow per-queue shell execution of the user's .login and .cshrc files for the *[s]* queue, enter:]

[[set exec_shell_login=Yes s]]]

Step 7 [[Using the `show long queue` command, display queue attributes to verify the change. The following example illustrates the use of this command to check if accounting is activated.]

```
[Mgr: show long queue s

s@hostC; type=BATCH; [ENABLED, INACTIVE]; pri=48
0 exit; 0 run; 0 stage; 0 queued; 0 wait; 0 hold; 0 arrive;

Accounting: On]]
```

Step 8 [[Repeat [Step 6] and [Step 7] for each queue that requires NQS accounting on this machine.]]

Step 9 [[Exit `qmgr`. Enter:]

```
[[exit]]]
```

Step 10 [[Repeat thse steps on each machine running NQS.]]]]]]

((Establishing a shell strategy)()

[A shell strategy determines the command interpreter used to interpret request script commands if a request is submitted to a queue without an identified []shell interpreter. Perform the following steps to set the shell strategy.]

Step 1 [[[[[Log in to machine you are configuring as an NQS manager.]]

Step 2 [[Start the `[qmgr]` utility. Enter:]

```
[[/opt/nqs/bin/qmgr]]
```

[The `[qmgr]` prompt appears:]

```
[[Mgr]:]]
```

Step 3 [[Set the shell strategy using the `[set shell_strategy]` command. The format for this command is:)

`[[set shell_strategy]` *[option]]*

[where *[option]* can be one of the following:]

`[[[fixed=]`*[shell]]*

[[Indicates the shell that interprets script file commands, where *[shell]* can be csh, ksh, or sh. Supply the absolute path name. The shell must exist and be executable or this command fails.]]

`[[free]]`

[[Indicates the user's login shell (as defined in the /etc/passwd file) is used initially to interpret commands. NQS then supplies the name of the script file to the login shell as standard input. The user's login shell reads the first line of the script file. If the first line specifies a shell, NQS uses that shell to interpret the script file commands. Otherwise NQS uses sh.]]

`[[login]]`

[[Indicates the user's default login shell (as defined in the /etc/password file) is used to interpret the script file commands.]]]

[For example, to set the shell strategy to free, enter:]

```
[[set shell_strategy free ]]]
```

Step 4 [[Exit `qmgr`. Enter:]

```
[[exit]]]
```

Step 5 [[Repeat these steps on each machine running NQS.]]]]]]

((Taking a qmgr snapshot)()

[Save the current NQS configuration on each machine as a series of `[qmgr]` commands using the `qsnapshot` command.]

[This command saves the configuration to screen by default; however, you can save the configuration to a file and later use the contents of that file to restore the NQS `[qmgr]` configuration. For example, to save the new configuration to a file named batch_qconfig, enter:]

`[[qsnapshot > batch_qconfig]]`

[The following example illustrates output from the `qsnapshot` command:]

`[# qsnapshot`

```
SET ACC_LOGFILE /dev/null
SET AID_MASK = 1
SET DEFAULT BATCH_REQUEST PRIORITY 31
SET DEFAULT DESTINATION_RETRY TIME 72
SET DEFAULT DESTINATION_RETRY WAIT 5
SET MAIL 0
SET GLOBAL PER_USER RUN_LIMIT = 4
SET MANAGERS root:m test:m janedoe:m johndoe:m
SET SHELL_STRATEGY LOGIN
CREATE PIPE_QUEUE best PRIORITY =48 SERVER = (opt/nqs/lib/pipeldav)
RUN_LIMIT = 1
CREATE PIPE_QUEUE v PRIORITY = 48 SERVER = (opt/nqs/lib/pipeclient)
RUN_LIMIT = 1
CREATE BATCH_QUEUE short PRIORITY = 48 RUN_LIMIT = 1 IMPORT_DIR = No
SHARE_POLICY FIXED = short
SET ACCOUNTING = Off short
SET COPY_OPEN_FILES = No short
SET NICE_VALUE_LIMIT = ( 0 ) short
SET PER_PROCESS PERMFILE_LIMIT = ( UNLIMITED ) short
SET MAXIMUM REQUEST_PRIORITY 63 short
SET PER_USER_RUN_LIMIT = 1 short
ADD ALIAS s short]
```

[For more information, refer to the qsnapshot(8), qmgr(8), and qmapmgr(8) man pages.]]

((Configuring error logging)()

[When an NQS utility encounters an error, it writes a message directly to the user's terminal. A user can usually interpret these messages.]

[When daemons, such as `[nqsdaemon]` and `[netdaemon]`, encounter an error, they communicate with `[logdaemon]`. `[logdaemon]` handles error logging for daemons according to level of severity, and sends the error message to the `[syslog]` daemon at the `[syslog]` level shown in [Table 3].]
`[[]]`

Table 3 [Error severity levels]

(Error level)	(Logdaemon action)	(Syslog level)
[Fatal]	[Log error, write to stdout, mail managers]	[LOG_CRIT]
[Error]	[Log error, write to stdout]	[LOG_ERR]
[Warn]	[Log error, write to stdout]	[LOG_WARNING]
[Info]	[Log error, write to stdout]	[LOG_INFO]
[Log]	[Log error]	[LOG_INFO]
[Debug]	[Log error]	[LOG_DEBUG]

`[[syslog]` handles these messages as defined in the /etc/syslog.conf file. Perform the following steps to configure the /etc/syslog.conf file to your needs:]

Step 1 [[[[[Log in on the system console for the machine you are configuring as the superuser.]]

Step 2 [[Change the syslog.conf file. Each line in the syslog.conf file represents a message group. The format for this file is:]

[[facility.level] `[send_message_here]` *]*

`[where ]`

[[[facility]]

[[Is the part of the system that generates the message.]]

[[level]]

[[Describes the error level of the message. This can be any error level listed in [Table 3]. When you choose an entry, the

system records errors for that entry level and all preceding entry levels in the chart. For example, if you choose `Info`, you receive error messages from `[Info]`, `[Warn]`, `[Error]`, and `[Fatal].`]]

[[send_message_here]]

[[Can be one of the following:]

- [[[[[Absolute path name of a file—writes messages to the named file. The file named here must exist before messages can be logged to it. Be sure to complete [Step 3].]]
- [[Hostname preceded with an at sign (@)—forwards messages to the named site.]]
- [[A list of users separated by commas. These users receive the messages if they are logged in.]]
- [[An asterisk—sends messages to all users logged in.]]]]]]]

[For example, to log all batch errors of levels preceding and including debug to /usr/adm/batchlog on an Exemplar Technical Server, enter the following line in /etc/syslog.conf:]

```
[[local0.debug /usr/adm/batchlog]]
```

[Refer to the syslogd(8) man page for more details on how to set up syslog.conf.]]

Step 3 [[If you supply a path name in the *[send_message_here]* field of the syslog.conf file, create those files using the `[touch]` command. For example, create /usr/adm/batchlog by entering:]

```
[[touch /usr/adm/batchlog]]]
```

Step 4 [[Reinitialize the syslog daemon by entering:]

```
[[kill -HUP 'cat /etc/syslog.pid']]]
```

Step 5 [[Check for the following files to make sure your Exemplar Technical Server starts the `syslogd` daemon each time the system boots:]

[/sbin/init.d/syslogd]

[/sbin/rc2.d/S220syslogd]]

Step 6 [[If you included debug messages for logging, decide on the debug level. The system sends significant debug messages to `[logdaemon]` only if the debug level is greater than 0, the following example illustrates the level of logging:]

```
[04/21/90 12:12 ConvexNQS+(DEBUG): main(): Configuration loaded.
04/21/90 12:12 ConvexNQS+(DEBUG): main(): Rebuild queue state.
04/21/90 12:12 ConvexNQS+(DEBUG): main(): Queue state rebuilt.
```

```
04/21/90 12:12 ConvexNQS+(DEBUG): main(): Enabling virtual timers.
04/21/90 12:12 ConvexNQS+(DEBUG):main():Looping to read request
packets.]]
```

Step 7 [[If you want to change the debug level, log in as an NQS manager.]]

Step 8 [[Start the `[qmgr]` utility. Enter:]

`[[/opt/nqs/bin/qmgr]]`

[The *`[qmgr]`* prompt appears:]

`[[Mgr:]]]`

Step 9 [[Set the debug level using the `[set debug]` command. The command format is:]

`[[set debug ]`*[level]]*

[where]

[[*[level]*]

[[Can be one of the following:]

[[`[0]`]

[[No debugging information is displayed.]]

[`[1]`]

[[Minimum debugging information is displayed.]]

[`[2]`]

[[Maximum debugging information is displayed.]]]]]]

Step 10 [[Repeat thse steps on each machine running NQS.]]]]]]

((Automating queue operations)()

[You can automatically control starting, stopping, and aborting queues using the `[cron]` utility.]

`[[cron]` runs commands at specified dates and times according to the instructions in the /.crontab file.]

[Refer to the crontab(5) man page for detailed information on using the `cron` utility and setting up the crontab file.]]

((Notifying users of changes)()

[After you have configured NQS, notify your users of changes and recommendations for the new NQS system.]

((Remote access capabilities)

[When NQS is configured on more than one machine, users can submit jobs to a remote machine if they have access privileges on the remote machine.]

[There are two ways to provide access to a remote machine:]

- [[[[[Edit the /etc/hosts.equiv file on the user's machine to include the remote machine.]]
- [[Create a .rhosts file in the user's home directory that includes the remote machine.]]]]]

[Tell your users if they should create a .rhosts file in their home directory.]]

((Miscellaneous information)

[Also, furnish users with the following information:]

- [[[[[Names and aliases of any NQS queue]]
- [[Default shell strategy established for the NQS system]]
- [[Import attribute setting for each NQS queue]]
- [[Manager privileges assignments]]
- [[Default limits for each NQS queue]]
- [[Hours that each NQS queue accepts and runs requests]]]]]]]

Controlling queue operations 3

Several commands are available for controlling queue operation, including commands for:

- Removing requests from a queue
- Enabling a queue
- Disabling a queue
- Starting a queue
- Stopping a queue

This chapter describes how to perform each of these tasks.

Getting started

Start the `qmgr` utility by logging in as an NQS manager. Then enter:

```
/opt/nqs/bin/qmgr
```

The following prompt appears:

```
Mgr:
```

Removing requests

You can remove requests from a queue by:

- Aborting all running requests
- Purging all nonrunning requests
- Moving requests to another queue

Each of these actions is described in the following sections.

Aborting requests

Use the `abort queue` command to abort all running requests in a queue.

NQS sends a SIGTERM signal to each process running in the queue, then sends a SIGKILL signal to any process that continues to run after the SIGTERM signal.

NQS removes all aborted requests from the queue and returns all output files associated with the requests to the appropriate destination.

The command format is:

`abort queue` *queuename [seconds]*

where

queuename

Is the name of the queue to be aborted.

seconds

Is the number of seconds to wait before executing the SIGKILL signal after the SIGTERM signal is sent. If you don't supply a seconds value, NQS delays 60 seconds.

Purging requests

Use the `purge queue` command to drop all nonrunning requests from a queue. NQS completes all running requests. The purged requests are irretrievable.

The command format is:

`purge queue` *queuename*

where *queuename* is the name of the queue to be purged.

Moving requests

Use the `move queue` command to move all nonrunning requests to another queue.

NQS moves requests regardless of queue limit violations, access restrictions, or attribute violations.

The command format is:

`move queue` *queuename des_queue*

where

queuename

Is the name of the queue whose requests are to be moved.

des_queue

Is the destination queue for the moved requests.

Enabling and disabling queues

You must enable a queue before the queue can accept requests for processing. You must disable it to prevent it from accepting requests.

This section describes how to enable and disable queues.

Enabling queues

Use the `enable queue` command to allow a queue to accept requests for processing.

The command format is:

`enable queue` *queuename*

where *queuename* is the name of the queue to be enabled.

Disabling queues

Use the `disable queue` command to prevent a queue from accepting requests.

The command format is:

`disable queue` *queuename*

where *queuename* is the name of the queue to be disabled.

Starting and stopping queues

You must start a queue before the queue can process accepted requests. You must stop it to prevent it from processing accepted requests.

Starting queues

Use the `start queue` command to allow a queue to process requests.

The command format is:

`start queue` *queuename*

where *queuename* is the name of the queue to be started.

Stopping queues

Use the `stop queue` command to prevent it from processing requests.

NQS completes currently running requests, but queues all nonrunning requests. Users may still submit new requests; however, they are also queued.

The command format is:

`stop queue` *queuename*

where *queuename* is the name of the queue to be stopped.

Controlling requests

4

Several commands are available for controlling requests. This chapter describes how to perform the following tasks:

- Deleting a specific request
- Delaying a queued request
- Delaying a running request
- Moving a specific nonrunning request
- Changing the priority of a nonrunning request
- Forcing a queued request to run

In addition, a request flow overview describes the flow of a request through each queue type.

Getting started

To perform most of the commands described in this chapter, you must know:

- A request's unique identifier
- How to start the `qmgr` utility

Determining a request ID

You can display a request's identifier and other request information using the `qstat` command. The command format is:

`qstat` *[option ...] [queuename[@hostname]...]*

where

option

Controls the type and amount of information displayed. If no options are specified, `qstat` shows only those requests belonging to the user issuing the command. *option* can be one or more of the following:

`-a`

Displays status for all requests in the queue.

`-l`

Displays additional information about the queue and requests.

`-m`

Displays the date and time requests are scheduled to run.

`-u` *username*

Displays only those requests belonging to the specified *username.*

`-x`

Displays additional information about the queue.

queuename

Is the name of the queue for which you want status information. If you do not supply a queue, NQS displays information for all queues on the requested machine.

hostname

Is the name of the machine that receives the request. If *hostname* is omitted, NQS assumes the local machine.

Refer to the `qstat`(1) man page or *NQS User's Guide* for more details.

For example, to display standard output for requests in the *v* queue on the local host, enter:

```
qstat v
```

Figure 3 illustrates the output for this command.

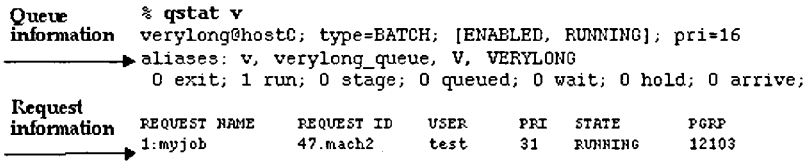


Figure 3 qstat standard output

`qstat` displays two types of information:

- Information on the queue
- Information on each request in the queue

Information important to the actions described in this chapter is request information. This information is described below.

```
 REQUEST NAME     REQUEST ID     USER           PRI          STATE        PGRP
 1:my
job          47.mach2       test           31           RUNNING     12103
```

`REQUEST NAME`

Name assigned to the request.

`REQUEST ID`

Unique identifier assigned to request when it is submitted to the queue.

`USER`

User submitting the request.

`PRI`

Intraqueue priority assigned to request. This priority affects which request runs next in a queue. Priority can be any number from 0 to 63; 0 is the lowest priority and 63 the highest.

`STATE`

State of the request. This can be one of the following options:

`ARRIVING`

Request is arriving at the queue.

`DEPARTING`

Request is departing from the queue but has not yet been received by the destination queue.

`EXITING`

Request has completed running and will exit from the system after NQS returns the required output files to their intended destinations.

`HOLDING`

A hold has been placed on the request, preventing it from entering any other state.

`QUEUED`

Request is queued and eligible for running or routing. This is the most common state.

ROUTING

Request has reached the head of a pipe queue and is being routed to another queue.

RUNNING

Request has reached its final destination batch queue and is running.

WAITING

Request is waiting for a specified amount of time to pass before trying to run. This could be because it was submitted with a future date and time specified for running, or because a pipe queue could not route the request but will try later.

SUSPENDED

Request is suspended from running. It remains suspended until manually resumed.

PGRP

Process group of the request, if available to the local NQS daemon. This information is displayed only for processes that are running.

For more information on queue or request properties, refer to the `qstat`(1) man page.

Starting qmgr

Start the `qmgr` utility by logging in as an NQS manager. Then enter:

```
/opt/nqs/bin/qmgr
```

The following prompt appears:

```
Mgr:
```

Deleting specific queue requests

The following two commands delete a specific request(s) from a queue:

- The `qmgr` utility `delete request` command
- The NQS `qdel` command

Each command is described in the following sections.

Using the delete request command

The `delete request` command is a `qmgr` utility command, which means you must start `qmgr` before you can use this command.

The command format is:

`delete request` *request_id* [*request_id ...*]

where *request_id* is the number assigned to the request when it is submitted to NQS. Use the `qstat` command to find the *request_id*.

You can specify more than one request.

You can specify a running or nonrunning request. If a request is running, NQS sends a SIGKILL signal to all processes in the request.

NQS removes deleted requests and discards them.

Using the qdel command

The `qdel` command is an NQS command that runs from a shell command line. The command format is:

`qdel` [*option*] *request_id [@ hostname] [request_id [@ hostname]...]*

where *option* can be one of the following:

`-u` *username*

Lets you delete requests other than your own, where *username* is the name of the user who owns the request. By default, only the user who submitted the request can delete it from a queue. This option lets the superuser or NQS manager delete someone else's request from a queue.

`-k`

Sends a SIGKILL (-9) signal to a running request. When the request exits, NQS deletes it.

sig

Sends a specified signal to a running request. *sig* can either be the signal number or signal name in the /usr/include/signal.h file.

request_id

The number assigned to the request when it is submitted to NQS. Use the `qstat` command to find the *request_id*.

You can specify more than one request.

You can specify a running or nonrunning request.

If you are using the `-u` option, the *request_id* must belong to the user defined in *username*.

hostname

The name of the machine where the queue containing the request resides. If you omit *hostname*, NQS assumes the local host.

For example, a user with manager privileges enters the following command to delete the request identified as *291* on the local machine submitted by user *smith*:

```
qdel -u smith 291
```

Delaying queued requests

Use the `hold request` and `release request` commands to delay the running of a queued request.

If an NQS manager places the request on hold, only an NQS manager can release the hold.

The `hold request` and `release request` commands are `qmgr` utility commands, which means you must start `qmgr` before you can use these commands.

Each command is described in the following sections.

Placing a queued request on hold

Use the `hold request` command to place a request on hold, thereby preventing it from running. The command format is:

`hold request` *request_id [request_id...]*

where *request_id* is the number assigned to the request when it is submitted to NQS. Use the `qstat` command to find the *request_id*.

You can specify more than one request.

You must specify a request in the queued state.

Releasing the hold on a queued request

Use the `release request` command to release the hold on a queued request, making it eligible for execution. The command format is:

`release request` *request_id [request_id ...]*

where *request_id* is the number assigned to the request when it is submitted to NQS. Use the `qstat` command to find the *request_id*.

You can specify more than one request.

You must specify a request in the holding state.

Moving specific nonrunning requests to another queue

Use the `move request` command to move a specific nonrunning request from one queue to another.

The `move request` command is a `qmgr` utility command, which means you must start `qmgr` before you can use this command.

The command format is:

`move request` *request_id [request_id ...] queue*

where

request_id

Is the number assigned to the request when it is submitted to NQS. Use the `qstat` command to find the *request_id*.

You can specify more than one request.

You must specify a nonrunning request.

queue

Is the name of the queue to which you want to move the request.

If the request violates any queue limits, access restrictions, or attributes in the receiving queue, NQS does not move the request.

To move a running request, first suspend the request, move it, then resume the suspended request.

Changing the priority of nonrunning requests

Use the `modify request` command to change the priority of a nonrunning request. The `modify request` command is a `qmgr` utility command, which means you must start `qmgr` before you can use this command.

The command format is:

`modify request priority =` *value request_id*

where

value

Is the new priority of the queue request. This is a number between 0 and 63; 0 is the lowest priority and 63 the highest. A user can only decrease the priority of a request. An NQS manager can raise a request's priority.

request_id

Is the number assigned to the request when it is submitted to NQS. Use the `qstat` command to find the *request_id*.

You can specify more than one request.

You must specify a nonrunning request.

Forcing requests to run

There are two commands to force a queued request to begin running immediately:

- The NQS `qrun` command
- The `qmgr` utility `run request` command

Each command is described in the following sections.

Using the qrun command

The `qrun` command is an NQS command that runs from a shell command line. The command format is:

`qrun request` *request_id* [*request_id ...*]

where *request_id* is the number assigned to the request when it is submitted to NQS. Use the `qstat` command to find the *request_id*.

You can specify more than one request.

If running the request exceeds the current run limit of the queue, NQS increases the queue's run limit by one, until the request finishes running.

Using the run request command

The `run request` command is a `qmgr` utility command, which means you must start `qmgr` before you can use this command.

The command format is:

`run request` *request_id [request_id ...]*

where *request_id* is the number assigned to the request when it is submitted to NQS. Use the `qstat` command to find the *request_id*.

You can specify more than one request.

If running the request exceeds the current run limit of the queue, NQS increases the queue's run limit by one, until the request finishes running.

Request flow overview

The remaining sections in this chapter describe the flow of a request through each of the following queue types:

- Local batch queue
- Remote batch queue
- Local pipe queue
- Remote pipe queue

Local batch queue processing

You submit a request to a local batch queue using the `qsub` command.

`qsub` sends the request to the `nqsdaemon`. The `nqsdaemon`:

- Checks any qualifiers included with the `qsub` command against defined queue limits and attributes
- Checks the ability of the queue to import files
- Determines whether the recipient queue is a pipe-only queue
- Accepts or rejects the request based on these checks

When the `nqsdaemon` determines it is time to run an accepted request, it spawns a shepherd process that:

- Sets up the environment for the request
- Runs the job
- Sends mail to the user if it cannot run the shell script submitted
- Waits for the job to complete
- Logs accounting information
- Undoes anything it set up to run the job (such as unmounting remote file systems mounted to import files)
- Returns output files to the user's directory

Remote batch queue processing

You submit a request to a remote batch queue using the `qsub` command.

`qsub` sends the request to the `netdaemon` on the remote machine.

The `netdaemon` spawns a netserver that tries to queue the request with the `nqsdaemon`.

The nqsdaemon:

- Checks to make sure the user has an account on the remote machine
- Checks to make sure the user name on the local machine matches the user name on the remote machine
- Checks any qualifiers included with the qsub command line against the defined queue limits and attributes
- Checks the ability of the queue to import files
- Determines if the recipient queue is a pipe-only queue
- Accepts or rejects the request based on these checks

When the nqsdaemon determines it is time to run an accepted request, it spawns a shepherd process that:

- Sets up the environment for the request
- Runs the job
- Sends mail to the user if it cannot run the shell script submitted
- Waits for the job to complete
- Logs accounting information
- Undoes anything it set up to run the job (such as unmounting remote file systems mounted to import files)
- Returns output files to the user's directory

Local pipe queue processing

You submit a request to a local pipe queue using the qsub command.

qsub sends the request to the nqsdaemon.

The nqsdaemon:

- Checks the qsub qualifiers against defined queue limits and attributes
- Determines whether the recipient queue is a pipe-only queue
- Accepts or rejects the request based on these checks
- Routes an accepted request when it reaches the top of the queue by spawning the pipe client

The pipe client selects a destination queue for the request based on characteristics of:

- The request
- Each queue in the destination set defined for the pipe queue

If the pipe client finds a suitable destination on a remote machine, it contacts the netdaemon on the destination machine and sends

the request to it. The netdaemon spawns the netserver, which in turn tries to queue the request with the local nqsdaemon.

If the pipe client does not find a suitable destination, it returns an appropriate transaction code to nqsdaemon.

Remote pipe queue processing

You submit a request to a remote pipe queue using the qsub command.

qsub sends the request to the netdaemon on the remote machine.

The netdaemon spawns a netserver that tries to queue the request with the nqsdaemon.

The nqsdaemon:

- Checks to make sure the user name on the local machine matches the user name on the remote machine
- Accepts or rejects the request based on this check
- Routes an accepted request when it reaches the top of the queue by spawning the pipe client

The pipe client selects a destination queue for the request based on characteristics of:

- The request
- Each queue in the destination set defined for the pipe queue

If the pipe client finds a suitable destination on a remote machine, it contacts the netdaemon on the destination machine and sends the request to it. The netdaemon spawns the netserver, which in turn tries to queue the request with the local nqsdaemon.

If the pipe client does not find a suitable destination, it returns an appropriate transaction code to nqsdaemon.

Generating accounting reports

5

The NQS batch accounting systems tracks the system resources used by an individual user or group by request.

Implementing NQS batch accounting involves identifying a log file to collect NQS-specific accounting data, and then enabling batch accounting on a per-batch-queue basis. See "Configuring NQS" on page 9 for more information.

Once you implement NQS batch accounting, you can generate accounting reports for:

- An entire batch system
- Specific users
- Specific queues
- Specific users in specific queues

This chapter describes how to generate NQS accounting reports.

Perform the following steps to generate NQS reports:

Step 1 Log in. You may need to log in as root, depending on the permissions set on the accounting log file.

Step 2 Generate a report using the `qsa` command. The command format is:

`qsa` *mode [constraints][acct-file]*

where

mode

Is the desired output. This can be one of the following:

`-r`

Raw mode
Formats each record that matches the specified constraints and writes it to the user's `stdout`. The following example illustrates one record from raw mode output:

```
% qsa -r -q l
queue long host mach1
sub time 643564104 com time 643564107
uid 16 gid 49 aid 0
seqno 224 rhost mach1 rprio -1 qprio 4 nice 0
usertime 0.076545 systime 0.151084
io 15
```

`-x`

Extended mode
Formats each record that matches the specified constraints and writes it to user's `stdout`. It differs from raw mode in that it adds time spent waiting in queues, time spent executing, and time between submission and completion; and it converts all time values to ASCII. The following example illustrates one record from extended mode output:

```
% qsa -x -q l
queue long host mach1
sub time Thu May 24 10:28:24 1990
com time Thu May 24 10:28:27 1990
sta time Thu May 24 10:28:24 1990
user test group dev aid 0
seqno 224 rhost mach1 rprio -1 qprio 4 nice 0
turnaround 3 secs
waited
ran 3 secs
user time 0.076545 system time 0.151084
io 15
```

`-s`

Summing mode

Processes each record that matches the specified constraints and appends to stdout totals for CPU time consumed, user CPU time consumed, system CPU time consumed, and I/O operations performed. The following example illustrates one record from summing mode output:

```
% qsa -s
in 17 records from file /var/adm/batchacct
total turnaround 2242 secs
total execution 2221 secs
total user time 1076.766425 secs
total system time 386.580745 secs
total io 19877 operations
```

`-a`

Averaging mode

Processes each record that matches the specified constraints and appends to `stdout` averages for CPU time consumed, user CPU time consumed, system CPU time consumed, I/O operations performed, time spent waiting in queue, time spent executing, and time between submission and completion. Without the `-q` option, gives averages for every queue on your system in which accounting is activated. The following example illustrates one record from averaging mode output:

```
% qsa -a
in 17 records from file /var/adm/batchacct
average turnaround 131.882355 secs
average execution 130.647064
average user time 63.306437
average system time 22.757830
average io 1169.235352 operations
```

constraints

Controls the records selected for processing. You can specify records by queue, user, or both. If you omit this option, `qsa` processes all records. Constraints can be one or more of the following:

`-Q`

Processes records in all queues. NQS groups records by each queue that appears in the accounting file. You cannot use this option with the `-q` flag.

`-q` *queuename*

Processes records in a specified queue where *queuename* can be one or more queues. NQS groups records by queues specified in one or more occurrences of this option. You cannot use this option with a `-Q` flag.

If you do not supply a queue name, `qsa` displays accounting information for all queues listed in the accounting file.

`-U`

Processes records for all users where *username* can be one or more names of users. NQS groups records by each user that appears in the accounting file. You cannot use this option with the `-u` flag.

`-u` *username*

Processes records for specific users. NQS groups records by users specified in one or more occurrences of this option. You cannot use this option with the `-U` flag.

If you do not supply a *username*, `qsa` displays accounting information for queues enabled for NQS batch accounting.

acct_file

Is the name of the log file where the NQS accounting data is stored. If you do not supply a file name, NQS uses /var/adm/batchacct.

Refer to the qsa(8) man page for more information.

qmgr commands 6

qmgr reference pages

This chapter contains a description of each `qmgr` command. Use these commands to configure and operate NQS on the local machine.

You can enter commands in any combination of uppercase and lowercase characters.

The first two or three characters of each word in each command are unique, and you need to enter only those characters for NQS to recognize the command. These accepted abbreviations are indicated in the "Syntax" section of each command description as uppercase letters.

NQS man pages

In addition, the following online man pages are available for NQS:

- batch-acct(5)
- nqsdaemon(8)
- pipeclient(8)
- qdel(1)
- qjlist(1)
- qlimit(1)
- qmapmgr(8)
- qmgr(8)
- qps(1)
- qrun(8)
- qsa(8)
- qsnapshot(8)
- qstat(1)
- qsub(1)

abort queue

Authorization

NQS manager privileges are required to use this command.

Description

Removes all running requests in a queue.

NQS sends a SIGTERM signal to each process of each request running in the queue. After the time indicated in *seconds* has passed, NQS also sends a SIGKILL signal to all remaining processes.

All running requests are lost when the queue is aborted; therefore, you must suspend and then resume any running requests you want to save before you abort the queue.

You can automatically abort queues using the `cron` utility. Refer to the crontab(5) man page for detailed information on using the cron utility and setting up the crontab file.

Syntax

`ABort Queue` *queuename* `[`*seconds*`]`

where

queuename

Name of the queue to abort.

seconds

Amount of real time NQS waits after sending a SIGTERM signal to send a SIGKILL signal. If you omit *seconds*, NQS assumes a default value of 60 seconds.

add alias

Authorization

NQS manager privileges are required to use this command.

Description

Adds an alternate name to a queue.

You can use the queue name or any alias assigned to the queue to reference the queue.

Syntax

`ADd Alias` *alias queuename*

where

alias

Unique, alternate name to add.

queuename

Name of the queue to which the alias is assigned.

add destination

Authorization

NQS manager privileges are required to use this command.

Description

Adds one or more destination queues to a pipe queue destination set.

If you supply more than one destination, separate items in the list with commas and surround the list with parentheses. For example, to designate three destinations, enter:

```
(destq1, destq2, destq3).
```

Syntax

`ADd DEStination =` *destination queuename*

where

destination

Name of the destination queue to add. The syntax of the destination queue name must be in one of the following forms. Where shown, square brackets [ ] are required.

local_queue_name

local_queue_name@local_machine_name

remote_queue_name@remote_machine_name

remote_queue_name@[remote_machine_mid]

You must define all remote machine names and machine IDs in the local system database. See "Installing NQS" on page 16 for details on how to do this.

queuename

Name of the queue to which you are adding destinations.

add groups

Authorization

NQS manager privileges are required to use this command.

Description

Adds one or more groups to the list of groups allowed access to a queue.

If you supply more than one group, separate items in the list with commas and surround the list with parentheses. For example, to designate three groups, enter:

```
(group1, group2, group3).
```

You must set the queue for no access before adding groups. See the `set no_access` command for details on how to do this.

Syntax

`ADd Groups =` *group queuename*

where

group

Name or GID of the group to add. Enclose a GID in square brackets [ ].

queuename

Name of the queue to which access is granted.

add managers

Authorization

NQS manager privileges are required to use this command.

Description

Grants one or more users access to `qmgr` commands.

Syntax

`ADd Managers` *username:option [username:option ...]*

where

username

Name of the user to receive access. The syntax of *username* must be in one of the following forms. Where shown, square brackets [ ] are required.

local_account_name

[local_account_id]

[remote_user_id]@ remote_machine_name

[remote_user_id]@[remote_machine_mid]

You must define all remote machine names and IDs in the local system database. See "Installing NQS" on page 16 for details on how to do this.

option

Indicates if the user receives manager privileges. `m` grants manager access.

add sender

Authorization

NQS manager privileges are required to use this command.

Description

Adds one or more pipe queues to the list of queues that can send requests to a demand queue (batch queue created with the demand attribute).

If you supply more than one pipe queue, separate items in the list with commas and surround the list with parentheses. For example, to designate three pipe queues, enter:

(sendq1,sendq2,sendq3).

Syntax

`ADd Sender =` *sender queuename*

where

sender

Name of the pipe queue to add. The syntax of the pipe queue name must be in one of the following forms. Where shown, square brackets [ ] are required.

local_queue_name

local_queue_name@local_machine_name

remote_queue_name@remote_machine_name

remote_queue_name@[remote_machine_mid]

You must define all remote machine names and machine IDs in the local system database. See "Installing NQS" on page 16 for details on how to do this.

queuename

Name of the demand queue to which you are adding senders.

Restriction

When the demand batch queue notifies senders, it accepts jobs on a first-come, first-served basis, which is generally the first sender in the list.

add users

Authorization

NQS manager privileges are required to use this command.

Description

Adds one or more users to the list of users allowed access to a queue.

If you supply more than one user, separate items in the list with commas and surround the list with parentheses. For example, to designate three users, enter:

```
(user1, user2, user3).
```

You must set the queue for no access before adding users. See the `set no_access` command for details on how to do this.

Syntax

`ADd Users =` *user queuename*

where

user

Name or UID of the user to add. Enclose a UID in square brackets [ ].

queuename

Name of the queue to which access is granted.

create batch_queue

Authorization

Manager privileges are required to use this command.

Description

Creates a new NQS batch queue.

Syntax

```
CReate Batch_queue queuename PRiority=priority
[PIpeonly] [Import_dir=import-option]
[Run_limit=run-limit] [Demand] [SEnder=sender]
[SUbcomplex = subcomplex name]
```

where

queuename

Name of the new batch queue. The name can consist of any printable nonblank character except for the at sign (@), a comma (,), an equal sign (=), and a left or right parenthesis ( ). The name cannot start with a digit.

priority

Interqueue priority for the queue. This priority affects queue selection for running the next job. Can be any number from 0 to 63; 0 is the lowest priority and 63 the highest.

`PIpeonly`

Indicates the queue can only accept requests submitted from a pipe queue. Otherwise, the queue can accept requests from any source (such as a program or script file).

import_option

Indicates if NQS imports the current working directory for a request (mounts the directory on the machine processing the request) before running the request. *import-option* can be:

`Yes`

NQS automatically imports the current working directory for any request running in the queue. A user can override this setting for individual requests using the `-ni` option of `qsub`.

`Available`

Lets a user specify importation of the current working directory for requests submitted to the queue using the `-i` option of `qsub`.

No

NQS does not import the current working directory for requests submitted to the queue. Furthermore, the queue rejects requests that require imported directories.

run-limit

Maximum number of requests that can run in the queue at any given time. If you do not supply a run limit, the value defaults to 1.

`Demand`

Indicates that requests may enter this queue only if they can execute immediately (which is determined by *run-limit*). `Demand` implies `PIpeonly`.

`sender`

Is the name of the pipe queue(s) that can send requests to this batch queue, and applies only if `Demand` is specified. If you supply more than one sender, separate items in the list with commas and surround the list with parentheses. For example, to designate three destinations, enter:

```
(sendq1, sendq2, sendq3)
```

The syntax of the sender queue name must be in one of the following forms. Square brackets [ ] are required where shown.

local_queue_name

local_queue_name@local_machine_name

remote_queue_name@remote_machine_name

remote_queue_name@[remote_machine_mid]

Note

Make sure all senders are defined with a server=(/opt/nqs/lib/pipedemand) attribute.

`SUbcomplex =` *subcomplex name*

Causes all requests run from this queue to execute on the named subcomplex. The subcomplex must be a valid and currently loaded subcomplex when the queue is created and started. A subcomplex can be configured using the `scm` utility or by using `qmgr COnfig_subcomplex`. `COnfig_subcomplex` executes the `scm` utility.

Note

The subcomplex permissions should be set to reflect the user and group set intended for use on a subcomplex batch queue.

create pipe_queue

Authorization

NQS manager privileges are required to use this command.

Description

Creates a new NQS pipe queue.

Syntax

`CReate Pipe_queue` *queuename* `PRiority=`*priority* `Server=`*server* `Destination=`*destination* `PIpeonly Run_limit=`*run-limit*

where

queuename

Name of the new pipe queue. The name can consist of any printable nonblank character except for the at sign (@), a comma (,), an equal sign (=), and a left or right parenthesis ( ). The name cannot start with a digit.

priority

Interqueue priority for the queue. This priority affects queue selection for running the next job. This can be any number from 0 to 63; 0 is the lowest priority and 63 the highest.

server

Name of the pipe client that transports submitted requests to one of the destination queues. *(server)* can be:

`pipeclient`

Routes the request to the first destination that will accept it. Destinations may reject a request due to queue limit violations or lack of account authorization. The full path name for this pipe client is /opt/nqs/lib/pipeclient.

`pipedemand`

Routes the request to the first destination that can run the job immediately. The full path name for this pipe client is /opt/nqs/lib/pipedemand.

`pipeldav`

Sorts the destination list by load factor and tries destinations with low load factors first. The full path name for this pipe client is /opt/nqs/lib/pipeldav.

Refer to the pipeclient(8) man page for more details.

destination

Name of one or more destination queues to which this pipe queue can route requests. If you supply more than one destination, separate items in the list with commas and surround the list with parentheses. For example, to designate three destinations, enter `(destq1, destq2, destq3)`.

The syntax for the destination queue name must match one of the following forms. Where shown, square brackets [ ] are required.

local_queue_name

local_queue_name@local_machine_name

remote_queuename@remote_machine_name

remote_queuename@[remote_machine_mid]

You must define all remote machine names and IDs in the local system database. See "Installing NQS" on page 16 for details on how to do this.

Note

If you specify a pipedemand pipe client, make sure all destination queues are defined with a demand attribute.

`PIpeonly`

Indicates the queue can only accept requests submitted from a pipe queue. Otherwise, the queue can accept requests from any source.

run-limit

Maximum number of pipe clients that may run simultaneously to deliver requests to their destination. If you do not supply a run limit, NQS defaults to 1.

delete alias

Authorization

NQS manager privileges are required to use this command.

Description

Deletes an alternate name for a queue.

Syntax

`DElete Alias` *alias*

where

alias

Alternate name to delete.

delete destination

Authorization

NQS manager privileges are required to use this command.

Description

Deletes one or more destination queues from a pipe queue destination set.

If you supply more than one destination, separate items in the list with commas and surround the list with parentheses. For example, to designate three destinations, enter:

```
(destq1, destq2, destq3).
```

NQS successfully completes any request being transferred to the deleted destination before deleting the destination.

If you delete all destinations for the pipe queue, the pipe queue is effectively stopped, although its actual status remains unchanged. Adding a new destination immediately starts the queue running again.

Syntax

`DElete DEStination =` *destination queuename*

where

destination

Name of the destination queue to delete. The syntax for the name must be in one of the following forms. Where shown, square brackets [ ] are required.

local_queue_name

local_queue_name@local_machine_name

remote_queue_name@ remote_machine_name

remote_queue_name@[remote_machine_mid]

You must define all remote machine names and IDs in the local system database. See "Installing NQS" on page 16 for details on how to do this.

queuename

Name of the pipe queue from which you are deleting destinations.

delete groups

Authorization

NQS manager privileges are required to use this command.

Description

Deletes one or more groups from the list of groups allowed access to a queue.

If you supply more than one group, separate items in the list with commas and surround the list with parentheses. For example, to designate three groups, enter:

```
(group1, group2, group3).
```

You can only use this command to delete groups added with the add groups command.

You must set the queue for no access before deleting groups. See the `set no_access` command for details on how to do this.

Syntax

`DElete Groups =` *group queuename*

where

group

Name or GID of the group to delete. Enclose a GID in square brackets [ ].

queuename

Name of the queue to which access is denied.

delete managers

Authorization

NQS manager privileges are required to use this command.

Description

Deletes one or more managers from the NQS manager access list.

You cannot delete the superuser account (root).

Syntax

`DElete Managers` *username:option* [*username:option* ...]

where

username

Name of the user to delete. The syntax of *username* must be in one of the following forms. Where shown, square brackets [ ] are required.

local_account_name

[local_account_id]

[remote_user_id]@ remote_machine_name

[remote_user_id]@[remote_machine_mid]

You must define all remote machine names and IDs in the local system database. See "Installing NQS" on page 16 for details on how to do this.

option

Indicates if the user was granted manager privileges. `m` removes manager access.

delete queue

Authorization

NQS manager privileges are required to use this command.

Description

Deletes a queue.

You cannot delete the queue unless it is empty and disabled. See the `disable queue` command in this chapter for details on how to do this.

Syntax

`DElete Queue` *queuename*

where

queuename

Name of the queue to delete.

delete request

Authorization

NQS manager privileges are required to delete a request you do not own.

Description

Deletes one or more requests from a machine.

You can delete any running or nonrunning request. If a request is running, NQS sends a SIGKILL signal to all processes in the request. NQS removes and discards deleted requests.

Syntax

`DElete Request` *request_id* [*request_id* ...]

where

request_id

Number assigned to the request when it is submitted to NQS. Use the `qstat` command to find the *request_id*. Refer to the qstat(1) man page for details on using the `qstat` command.

delete sender

Authorization

NQS manager privileges are required to use this command.

Description

Deletes one or more pipe queues from a demand queue sender set. A demand queue is a batch queue created with the demand attribute.

If you supply more than one pipe queue, separate items in the list with commas and surround the list with parentheses. For example, to designate three destinations, enter:

```
(sendq1, sendq2, sendq3).
```

NQS successfully completes any request being transferred from the deleted sender before deleting the pipe queue from the sender set. If you delete all senders for a demand queue, the demand queue is effectively stopped, although its actual status remains unchanged. Adding a new sender immediately starts the queue running again.

Syntax

`DElete Sender =` *sender queuename*

where

sender

Name of the pipe queue to delete. The syntax of the pipe queue name must be in one of the following forms. Where shown, square brackets [ ] are required.

local_queue_name

local_queue_name@local_machine_name

remote_queue_name@remote_machine_name

remote_queue_name@[remote_machine_mid]

You must define all remote machine names and machine IDs in the local system database. See "Installing NQS" on page 16 for details on how to do this.

queuename

Name of the demand queue from which you are deleting senders.

delete users

Authorization

NQS manager privileges are required to use this command.

Description

Deletes one or more users from the list of users allowed access to a queue.

If you supply more than one user, separate items in the list with commas and surround the list with parentheses. For example, to designate three users, enter:

```
(user1, user2, user3).
```

You can only use this command to delete users added with the add users command.

You must set the queue for no access before deleting users. See the `set no_access` command for details on how to do this.

Syntax

`DElete Users =` *user queuename*

where

user

Name or UID of the user to delete. Enclose a UID in square brackets [ ].

queuename

Name of the queue to which access is denied.

disable queue

Authorization

NQS manager privileges are required to use this command.

Description

Prevents a queue from accepting requests for processing.

Syntax

`DIsable Queue` *queuename*

where

queuename

Name of the queue to disable.

enable queue

Authorization

NQS manager privileges are required to use this command.

Description

Lets a queue accept requests for processing.

If the queue is already enabled, NQS ignores the command.

Syntax

`ENable Queue` *queuename*

where

queuename

Name of the queue to enable.

exit

Description

Exits from the NQS `qmgr` utility program.

You can also end the `qmgr` utility by sending an end-of-file character. The end-of-file character is typically **CTRL-d**.

Syntax

```
EXit
```

help

Description

Invokes the `qmgr` help facility and displays information about a `qmgr` command or topic.

Syntax

`HElp [`*command*`]`

where

command

Command for which you need information. If you do not supply a command, `qmgr` displays information describing available commands.

hold request

Authorization

NQS manager privileges are required to hold a request you do not own.

Description

Places one or more queued requests on hold, preventing them from running.

A request remains on hold in the queue until you release it with the release request command.

You cannot place a running request on hold.

Syntax

`HOld Request` *request_id* `[`*request_id* `...]`

where

request_id

Number assigned to the request when it is submitted to NQS. Use the `qstat` command to find the *request_id*. Refer to the qstat(1) man page for details on using the `qstat` command.

modify request

Authorization

NQS manager privileges are required to raise a request's priority. Users can lower the priority of jobs they own.

Description

Changes the intraqueue priority of one or more queued requests so that they run in a different order.

You cannot change the priority of a running request.

Syntax

`MODify Request priority=`*value request_id* `[`*request_id ...*`]`

where

value

New priority. This can be a number between 0 and 63; 0 is the lowest priority and 63 the highest.

request_id

Number assigned to the request when it is submitted to NQS. Use the `qstat` command to find the *request_id*. Refer to the qstat(1) man page for details on using the `qstat` command.

move my_request

Authorization

NQS manager privileges are required to move a request you do not own.

Description

Moves one or more of your queued requests to another queue.

NQS does not move the request if any queue limits, attributes, or access restrictions are violated.

To move a request that is running, first suspend the request, move it, then resume the suspended request.

Syntax

`MOVe My_request` *request_id* `[`*request_id ...*`]` *queuename*

where

request_id

Number assigned to the request when it is submitted to NQS. Use the `qstat` command to find the *request_id*. Refer to the qstat(1) man page for details on using the `qstat` command.

queuename

Name of the destination queue.

move queue

Authorization

NQS manager privileges are required to use this command.

Description

Moves all nonrunning requests in a queue to another queue.

NQS moves the requests regardless of any queue limit, access restrictions, or attribute violations.

You cannot move running requests.

Syntax

`MOVe Queue` *queuename dest_queue*

where

queuename

Name of the current queue.

dest_queue

Name of the destination queue.

move request

Authorization

NQS manager privileges are required to use this command.

Description

Moves one or more nonrunning requests to another queue.

NQS does not check for queue limit violations, access restrictions, or attribute violations at the destination queue before moving the requests.

To move a request that is running, first suspend the request, move it, then resume the suspended request.

Syntax

`MOVe Request` *request_id* `[`*request_id ...*`]` *queuename*

where

request_id

Number assigned to the request when it is submitted to NQS. Use the `qstat` command to find the *request_id*. Refer to the qstat(1) man page for details on using the `qstat` command.

queuename

Name of the destination queue.

purge queue

Authorization

NQS manager privileges are required to use this command.

Description

Removes all nonrunning requests from a queue.

NQS completes all running requests. The purged requests are irretrievable.

Syntax

`Purge Queue` *queuename*

where

queuename

Name of the queue to purge.

release request

Authorization

NQS manager privileges are required to release a request you do not own. If an NQS manager puts a request on hold, only an NQS manager can release it.

Description

Releases the hold on one or more queue requests, making them eligible to run.

A request must be in a holding state in order to be released.

Syntax

`RELease Request` *request_id* [*request_id ...*]

where

request_id

Number assigned to the request when it is submitted to NQS. Use the `qstat` command to find the *request_id*. Refer to the qstat(1) man page for details on using the `qstat` command.

run request

Authorization

NQS manager privileges are required to use this command.

Description

Forces one or more requests to begin running immediately.

If running the request exceeds the run limit of the queue, NQS increases the queue run limit until the request finishes running.

Syntax

`RUn Request` *request_id* [*request_id ...*]

where

request_id

Number assigned to the request when it is submitted to NQS. Use the `qstat` command to find the *request_id*. Refer to the qstat(1) man page for details on using the `qstat` command.

set acc_logfile

Authorization

NQS manager privileges are required to use this command.

Description

Defines a log file that collects NQS batch accounting information for an NQS machine.

Syntax

`SEt ACC_logfile` *logfile_name*

where

logfile_name

Name of the log file.

set accounting

Authorization

NQS manager privileges are required to use this command.

Description

Enables NQS batch accounting for a batch queue.

Syntax

`SEt ACCOunting = {ON|OFF}` *queuename*

where

queuename

Name of the batch queue to set.

set debug

Authorization

NQS manager privileges are required to use this command.

Description

Defines the level of debug output logged when an NQS utility encounters an error.

Syntax

`SEt DEBug`*level*

where

level

The amount of debug information logged:

`0`

Displays no debugging information.

`1`

Displays minimum debugging information.

`2`

Displays maximum debugging information.

set default batch_request priority

Authorization

NQS manager privileges are required to use this command.

Description

Defines the default intraqueue priority for any batch request submitted without a priority assignment to a queue.

This priority determines the relative ordering of requests within the queue.

Syntax

`SEt DEFault Batch_request Priority` *priority*

where

priority

Default priority. This can be an integer ranging between 0 and 63; 0 is the lowest priority and 63 the highest.

set default batch_request queue

Authorization

NQS manager privileges are required to use this command.

Description

Defines the default queue for any batch request submitted without a queue assignment to an NQS machine.

Syntax

`SEt DEFault Batch_request Queue` *queuename*

where

queuename

Name of the default queue.

set default destination_retry time

Authorization

NQS manager privileges are required to use this command.

Description

Sets the default amount of time that can elapse while NQS tries to resubmit a request to a destination queue through a pipe queue.

If this time elapses, the request fails.

Syntax

`SEt DEFault DEStination_retry Time` *retry-time*

where

retry-time

Amount of time in hours.

set default destination_retry wait

Authorization

NQS manager privileges are required to use this command.

Description

Defines the default time to wait before NQS resubmits a request to a destination queue through a pipe queue.

If a pipe queue destination fails to accept a request because of a network failure or remote server failure, NQS disables the destination for the number of minutes set in *wait-time*. At the end of this time, NQS enables the destination and resubmits the request.

Use the `set default destination_retry time` command to prevent an infinite number of retries.

Syntax

`SEt DEFault DEStination_retry Wait` *wait-time*

where

wait-time

Amount of time in minutes.

set description

Authorization

NQS manager privileges are required to use this command.

Description

Defines the description associated with a queue.

Syntax

`SEt DESCription =` *(description) queuename*

where

description

A character string that describes the queue. This string can be up to 79 characters long. Enter **NULL** to delete a description.

queuename

Name of the queue to set.

set destination

Authorization

NQS manager privileges are required to use this command.

Description

Creates a new destination set for a pipe queue.

If you supply more than one destination, separate items in the list with commas and surround the list with parentheses. For example, to designate three destinations, enter:

```
(destq1, destq2, destq3).
```

NQS removes all previously-defined destinations.

Syntax

`SEt Destination =` *destination queuename*

where

destination

Name of the destination queue to add. The syntax of the destination queue name must be in one of the following forms:

local_queue_name

local_queue_name@local_machine_name

remote_queue_name@remote_machine_name

remote_queue_name@[remote_machine_mid]

Where shown, square brackets [ ] are required.

You must define all remote machine names and IDs in the local system database. See "Installing NQS" on page 16 for details on how to do this.

queuename

Name of the pipe queue for which you are changing destinations.

set global per_user run_limit

Authorization

NQS manager privileges are required to use this command.

Description

Controls the number of requests a single user can run in all NQS queues at one time.

Syntax

`SEt Global Per_user Run_limit =` *run-limit*

where

run-limit

Total number of requests.

set exec_shell_login

Authorization

NQS manager privileges are required to use this command.

Description

Sets the status of the per-queue shell execution resource.

Syntax

`SEt_exec_shell_login` = *Answer queue*

Answer

can be one of the following:

`Yes`
Requests submitted to this queue will be executed under a login shell, unless `qsub -nl` is used to submit the request.

`No`
Requests cannot be executed under a login shell.

`Available`
Requests will be executed under a login shell only if specifically asked for using `qsub -l`.

queue

Queue name

set import_dir

Authorization

NQS manager privileges are required to use this command.

Description

Establishes whether a batch queue imports the current working directory for a request (mounts the directory on the machine processing the request) before running the request.

NQS imports directories with NFS by making temporary mount points in /tmp of the originating machine. Be aware of local automatic clean-up facilities of /tmp that could change these NFS mount points.

To use the import option, the system manager must first edit the /etc/exports file to add entries for all file systems eligible for remote mounting. Refer to the exports(5) and exportfs(8) man pages for details on how to do this.

Syntax

`SEt Import_dir =` *import-option queuename*

where

import-option

Can be one of the following:

`Yes`

Queue automatically imports the current working directory for requests submitted to the queue. A user can override this setting for individual requests using the `-ni` option of `qsub`.

`Available`

Lets the user specify importation of the current working directory for requests submitted to the queue using the `-i` option of `qsub`.

`No`

Queue does not import the current working directory. Furthermore, the queue rejects requests that require imported directories.

queuename

Name of the batch queue to set.

set mail

Authorization

NQS manager privileges are required to use this command.

Description

Defines the user account name appearing as sender for NQS mail. The default account name is the superuser account.

This mail notifies users (when appropriate) of events concerning their NQS requests.

Syntax

`SEt MAIl` *accountname*

where

accountname

Name of the user account. The name can consist of any printable nonblank character except for the at sign (@), a comma (,), an equal sign (=), and a left or right parenthesis ().

set managers

Authorization

NQS manager privileges are required to use this command.

Description

Defines a list of authorized `qmgr` managers.

NQS removes all previously-defined authorizations.

NQS managers have full privileges for all `qmgr` commands.

Syntax

`SEt MANagers` *username:option* `[`*username:option ...*`]`

where

username

Name of the user to receive access. The syntax of *username* must be in one of the following forms. Where shown, square brackets [ ] are required.

local_account_name

[local_account_id]

[remote_user_id]@ remote_machine_name

[remote_user_id]@[remote_machine_mid]

You must define all remote machine names and IDs in the local system database. See "Installing NQS" on page 16 for details on how to do this.

option

Indicates if the user receives manager privileges. `m` grants manager access.

set maximum request_priority

Authorization

NQS manager privileges are required to use this command.

Description

Defines the maximum intraqueue priority a queue can accept for any request submitted to the queue.

NQS lowers the priority for any request submitted with a higher priority than the queue priority.

Syntax

`SEt MAXimum Request_priority=`*priority queuename*

where

priority

Maximum priority. This can be an integer between 0 and 63; 0 is the lowest priority and 63 the highest.

queuename

Name of the queue to set.

set nice_value_limit

Authorization

NQS manager privileges are required to use this command.

Description

Defines the minimum nice value for any process in a running request in a queue.

The nice value determines the proportion of CPU time allocated to a process relative to all other processes in the system. The lower the nice value assigned to a process, the higher the proportion of CPU time allocated to that process. A practical range is from -20 to 20.

If a user submits a request with a nice value limit, NQS checks the request limit to ensure it does not exceed the queue limit. If the request limit exceeds the queue limit, NQS rejects the request.

If a user submits a request without a nice value limit, NQS uses the queue limit as the default limit.

NQS assigns limits to a request when the request is queued. If a queue limit is changed after a request is queued, the queued request is not affected; however, if the previously-queued request exceeds the new queue limit, NQS displays a warning message.

Syntax

`SEt NIce_value_limit =` *nice-value queuename*

where

nice-value

Minimum nice value. This can be an integer between -64 and 64.

queuename

Name of the queue to set.

set no_access

Authorization

NQS manager privileges are required to use this command.

Description

Deletes all groups and users from the access list for a queue, rendering it inaccessible.

This command does not affect requests in the queue.

This command does not affect superuser privileges; the superuser always has access to all enabled queues, regardless of the contents of the access list.

To restrict access to a queue, use the `set no_access` command to remove all access, then use the add users and add groups commands to specify the users and groups permitted access.

Syntax

`SEt NO_Access` *queuename*

where

queuename

Name of the queue to restrict access.

set no_default batch_request queue

Authorization

NQS manager privileges are required to use this command.

Description

Specifies there is no default batch queue for the NQS network, which means all jobs submitted to NQS must have a queue specification.

Syntax

```
SEt NO_Default Batch_request Queue
```

set per_process permfile_limit

Authorization

NQS manager privileges are required to use this command.

Description

Defines the maximum permanent file size for any process in a running request in a queue.

If a user submits a request with a permanent file size limit, NQS checks the request limit to ensure it does not exceed the queue limit. If the request limit exceeds the queue limit, NQS rejects the request.

If a user submits a request without a permanent file size limit, NQS uses the queue limit as the default limit.

NQS assigns limits to a request when the request is queued. If a queue limit is changed after a request is queued, the queued request is not affected; however, if the previously-queued request exceeds the new queue limit, NQS displays a warning message.

If any process tries to create a permanent file larger than the limit set, NQS sends a SIGXFSZ signal to the offending process. If the process does not catch the signal or ignores the signal, the process exits.

Syntax

`SEt PER_Process Permfile_limit =` *(limit) queuename*

where

limit

Maximum size a permanent file can be for any process in the running request. This can be any integer representing less than 100,000,000 bytes with an optional fractional part. The syntax for *limit* is

integer [.fraction] [units]

where *integer* and *fraction* are strings of up to eight decimal digits and *units* is one of the following:

`b`

Bytes

`w`

Words

`kb`

Kilobytes (2^{10} bytes)

`kw`

Kilowords (2^{10} words)

`mb`

Megabytes (2^{20} bytes)

`mw`

Megawords (2^{20} words)

`gb`

Gigabytes (2^{30} bytes)

`gw`

Gigawords (2^{30} words)

queuename

Name of the queue to set.

Notes

You must set the maximum permanent file size for all batch queues. Recommended setting is 4194303 b.

set per_user run_limit

Authorization

NQS manager privileges are required to use this command.

Description

Controls the number of requests a user can run in a queue at any one time.

NQS applies per-user run limits after applying per-queue run limits.

Syntax

`SEt Per_user Run_limit =` *run-limit queuename*

where

run-limit

Maximum number of requests. A *run-limit* of 0 disables enforcement of this limit.

queuename

Name of the queue to set.

set pipe_client

Authorization

NQS manager privileges are required to use this command.

Description

Defines the pipe client for a pipe queue.

Syntax

`SEt PIpe_client =` *(client) queuename*

where

client

Name of the pipe client.

You must define the full path name of the pipe client, followed by any arguments required by the program.

NQS supplies three pipe clients:

- /opt/nqs/lib/pipeclient for standard pipe clients
- /opt/nqs/lib/pipedemand for demand queue pipe clients
- /opt/nqs/lib/pipeldav for load-balancing pipe clients

Refer to the pipeclient(8) man page for more information.

queuename

Name of the pipe queue to set.

set priority

Authorization

NQS manager privileges are required to use this command.

Description

Defines the interqueue priority for a queue.

This priority affects queue selection for running the next job.

Syntax

`SEt PRiority =` *priority queuename*

where

priority

Interqueue priority. This can be any number between 0 and 63; 0 is the lowest priority and 63 the highest.

queuename

Name of the queue to set.

set run_limit

Authorization

NQS manager privileges are required to use this command.

Description

Controls the maximum number of requests that can run in a queue at any one time.

NQS applies per-queue run limits before applying per-user run limits.

Syntax

`SEt Run_limit =` *run-limit queuename*

where

run-limit

Maximum number of requests. If you omit *run-limit*, the value defaults to one.

queuename

Name of the queue to set.

set sender

Authorization

NQS manager privileges are required to use this command.

Description

Defines a list of pipe queues that can send requests to a demand queue (batch queue created with the demand attribute).

If you supply more than one pipe queue, separate items in the list with commas and surround the list with parentheses. For example, to designate three pipe queues, enter:

```
(sendq1,sendq2,sendq3).
```

Syntax

`SEt SENder =` *sender queuename*

where

sender

Name of the pipe queue. The syntax of the pipe queue name must be in one of the following forms:

local_queue_name

local_queue_name@local_machine_name

remote_queue_name@remote_machine_name

remote_queue_name@[remote_machine_mid]

Where shown, square brackets [ ] are required.

You must define all remote machine names and machine IDs in the local system database. See "Installing NQS" on page 16 for details on how to do this.

queuename

Name of the demand queue for which you are defining senders.

Restriction

When the demand batch queue notifies senders, it accepts jobs on a first-come, first-served basis, which is generally the first sender in the list.

set shell_strategy fixed

Authorization

NQS manager privileges are required to use this command.

Description

Defines the shell used to interpret batch request script file commands submitted without a specified interpreter to NQS.

Syntax

`SEt SHell_strategy FIxed =` *(shell)*

where

shell

This can be `csh`, `ksh`, or `sh`. Supply the absolute path names. The shell must exist and be executable or this command fails.

set shell_strategy free

Authorization

NQS manager privileges are required to use this command.

Description

Indicates the user's login shell (as defined in the /etc/passwd file) as the initial shell used to interpret batch request script file commands submitted without a specified interpreter to NQS.

NQS supplies the name of the script file to the login shell as standard input. The user's login shell reads the first line of the script file. If the first line specifies a shell, NQS uses that shell to interpret script file commands. Otherwise, NQS uses sh. In other words, the request is interpreted as though it were run interactively.

Syntax

```
SEt SHell_strategy FRee
```

set shell_strategy login

Authorization

NQS manager privileges are required to use this command.

Description

Indicates the user's login shell (as defined in the /etc/passwd file) as the shell used to interpret batch request script file commands submitted without a specified interpreter to NQS.

Syntax

```
SEt SHell_strategy Login
```

set unrestricted_access

Authorization

NQS manager privileges are required to use this command.

Description

Adds all groups and all users to the access list for a queue, making it accessible to any user or group.

Syntax

`SEt Unrestricted_access` *queuename*

where

queuename

Name of the queue to access.

show

Description

Provides status information about NQS on the local machine.

Syntax

`SHOW` *option*

where

option

Can be one of the following:

`ALL`

Displays information about NQS queues, authorized managers, operating parameters and supported resource limits.

`LIMITS_SUPPORTED`

Displays supported resource limits.

`LONG_QUEUE`

Displays NQS queue status in an expanded format.

`MANAGERS`

Displays the list of authorized NQS managers.

`PARAMETERS`

Displays NQS operating parameters.

`QUEUE`

Displays NQS queue status in a short format.

The following pages describe each option in detail.

show all

Description

Displays information about NQS queues, authorized managers, operating parameters and supported resource limits.

Syntax

```
SHOw All
```

Output

The following sample shows output from this command:

```
Mgr: show all
Queues:
Queue for short jobs.
short@mach1; type=BATCH;  [ENABLED, INACTIVE];  pri=48
aliases: s, short_queue, S, SHORT
subcomplex: nqs
 0 exit; 0 run; 0 stage; 0 queued; 0 wait; 0 hold; 0 arrive;

Queue for long jobs.
long@mach1;  type=BATCH;  [ENABLED, INACTIVE];  pri=32
aliases: l, long_queue, L, LONG
subcomplex: nqs
 0 exit; 0 run; 0 stage; 0 queued; 0 wait; 0 hold; 0 arrive;

Managers:
 root:m
 batchop:m

ConvexNQS+ operating parameters:
 Debug level =  0
 Default batch_request priority = 31
 Default batch_request queue = long
 Default destination_retry time = 72 hours
 Default destination_retry wait = 5 minutes
 Accounting log file = /dev/null
 Activity ID mask = None
\x11Mail account = root
\x11Next available sequence number = 26
 Batch request shell choice strategy = FREE
```

show limits_supported

Description

Displays the resource limits the operating system on the local machine can enforce. If you submit a request with a limit that is not enforced, NQS ignores the limit.

Syntax

```
SHOw LImits_supported
```

Output

The following example shows sample output from this command:

```
Mgr: show limits_supported
Per-process permanent file size limit (-lf)
Nice value (-ln)
```

show long queue

Description

Displays NQS queue status in an expanded format.

Syntax

`SHOw LOng Queue` *queuename username*

where

queuename

Name of the queue to display. If you omit *queuename*, NQS displays status information for all NQS queues.

username

Name of the user requests to display. If you omit *username*, NQS displays the status of each request in the queue.

Output

The following sample shows output from this command:

```
Mgr: show long queue v
Queue for very long jobs.
verylong@mach1;  type=BATCH;  [ENABLED, INACTIVE];  pri=16 aliases: v,
verylong_queue, V, VERYLONG
 0 exit; 0 run; 0 stage; 0 queued; 0 wait; 0 hold; 0 arrive;
 Run_limit = 1;
 Accounting: Off
 Activity ID offset: 0
 Add: maximum request priority: 63
 Cumulative system space time = 1703.580000 seconds
 Cumulative user space time = 2020.910000 seconds
 Unrestricted access
 Import directory: Yes
\x11Checkpoint: Not available
\x11Shell execution as login: No
 Per-process core file size limit = UNLIMITED
 Per-process data size limit = UNLIMITED
 Per-process permanent file size limit = UNLIMITED
 Per-process execution nice value = 4
```

show managers

Description

Displays the list of authorized NQS managers.

Syntax

```
SHOw Managers
```

Output

The following example shows sample output from this command:

```
Mgr: show managers

root:m
leader:m
```

show parameters

Description

Displays the current values for the general NQS operating parameters.

Syntax

```
SHOw Parameters
```

Output

The following sample shows output from this command:

```
Mgr: show parameters
Debug level = 2
Default batch_request priority = 31
Default batch_request queue = long
Default destination_retry time = 72 hours
Default destination_retry wait = 5 minutes
Global per-user runlimit = NONE
Accounting log file = /dev/null
Activity ID mask = None
Mail account = root
Next available sequence number = 201
Batch request shell choice strategy = FREE
```

show queue

Description

Displays NQS queue status in a short format.

Syntax

`SHOw Queue` *[queuename] [username]*

where

queuename

Name of the queue to display.

If you omit *queuename*, NQS displays status information for all NQS queues.

username

Name of the user requests to display.

If you omit *username*, NQS displays the status of each request in the queue.

Output

The following sample shows output from this command:

```
Mgr: show queue v
Queue for very long jobs.
verylong@mach1; type=BATCH; [ENABLED, INACTIVE]; pri=16
aliases: v, verylong_queue, V, VERYLONG
 0 exit; 0 run; 0 stage; 0 queued; 0 wait; 0 hold; 0 arrive;
```

shutdown

Authorization

NQS manager privileges are required to use this command.

Description

Stops requests where appropriate, and then shuts down NQS on the local machine.

NQS sends a SIGTERM signal to each process of each running request. After the time indicated in *seconds* has passed, NQS also sends a SIGKILL signal to those processes that have not changed process groups.

All requests terminated as a result of the `shutdown` command are requeued for later execution.

Syntax

`SHUtdown` *[seconds] [force]*

where

seconds

Amount of real time that NQS waits after sending a SIGTERM signal to send a SIGKILL signal. If you omit *seconds*, NQS assumes a default value of 60 seconds.

`force`

Forces NQS to shut down.

start nqs+

Authorization

NQS manager privileges are required to use this command.

Description

Starts NQS on the local machine.

You cannot use this command to start NQS the first time you install NQS.

Syntax

```
STArt Nqs+
```

start queue

Authorization

NQS manager privileges are required to use this command.

Description

Allows a queue to process requests.

If the queue is already started, NQS sends a transaction completion message; no jobs are aborted.

You can automatically start queues using the `cron` utility. Refer to the crontab(5) man page for detailed information on using the cron utility and setting up the crontab file.

Syntax

`STArt Queue` *queuename*

where

queuename

Name of the queue to start.

stop queue

Authorization

NQS manager privileges are required to use this command.

Description

Prevents a queue from processing requests.

NQS completes currently running requests but freezes all nonrunning requests. These frozen requests remain in the queue and run when you restart the queue using the `start queue` command.

You may still submit new requests; however, they are also frozen.

You can automatically stop queues using the `cron` utility. Refer to the crontab(5) man page for detailed information on using the cron utility and setting up the crontab file.

Syntax

`STOp Queue` *queuename*

where

queuename

Name of the queue to stop.

qmapmgr commands 7

qmapmgr reference pages

This chapter contains a description of each `qmapmgr` command. Use these commands to build and maintain the NQS database on the local machine.

You can enter commands in any combination of uppercase and lowercase characters.

The first two characters of each word in each command are unique, and you need to enter only those characters for NQS to recognize the command. These accepted abbreviations are indicated in the "Syntax" section of each command description as uppercase characters.

NQS man pages

Online man pages available for NQS are listed below:

- batch-acct(5)
- nqsdaemon(8)
- pipeclient(8)
- qdel(1)
- qjlist(1)
- qlimit(1)
- qmapmgr(8)
- qmgr(8)
- qps(1)
- qrun(8)
- qsa(8)
- qsnapshot(8)
- qstat(1)
- qsub(1)

add mid

Authorization

Superuser privileges are required to use this command.

Description

Adds a new machine identification to the NQS database.

Syntax

`Add Mid` *mid principal-name*

where

mid

Unique number identifying the machine to add.

principle-name

Name of the corresponding NQS machine. The name must be unique and correspond to an entry in the /etc/hosts file; it should not be an alias.

add name

Authorization

Superuser privileges are required to use this command.

Description

Adds an alternate name for a machine to the NQS database.

You can use the machine name or any alias assigned to the machine to reference a machine.

Syntax

`Add Name` *alias mid*

where

alias

Unique, alternate name. This alias is known only to the local NQS machine and is not recognized across machines.

mid

MID of the machine receiving the alias. The MID must already exist in the NQS database.

change name

Authorization

Superuser privileges are required to use this command.

Description

Changes the name of a machine in the NQS database.

Syntax

`CHange Name` *mid new-name*

where

mid

MID of the machine to change. The MID must already exist in the NQS database.

new-name

New machine name. The new name must be unique and correspond to an entry in the /etc/hosts file.

create

Authorization

Superuser privileges are required to use this command.

Description

Creates a new NQS database.

Syntax

```
CReate
```

delete mid

Authorization

Superuser privileges are required to use this command.

Description

Deletes a machine from the NQS database.

Syntax

`Delete Mid` *mid*

where

mid

MID of the machine to delete.

delete name

Authorization

Superuser privileges are required to use this command.

Description

Deletes an alternate name used to reference a machine.

Syntax

`Delete Name` *alias*

where

alias

Alternate name to delete.

exit

Description

Exits from the NQS `qmapmgr` utility.

You can also exit `qmapmgr` by entering the `quit` command or by pressing **CTRL-d**.

Syntax

```
Exit
```

get mid

Description

Displays the MID for a machine.

Syntax

`Get Mid` *machine-name*

where

machine-name

Name or alias of the machine to display.

get name

Description

Displays the machine name matching a MID.

Syntax

`Get Name` *mid*

where

mid

MID of the machine to display.

help

Description

Invokes the `qmapmgr` help facility and displays all available `qmapmgr` commands.

Syntax

```
Help
```

Output

The following example shows sample output from this command:

```
Mapmgr: help
Commands:
Add Name <name> <to_mid>
Add Mid <mid> <principal-name>
CHange Name <mid> <new-name>
CReate
Delete Name <name>
Delete Mid <mid>
Exit
Get Mid <name>
Get Name <mid>
Help
Quit
SHow
^D (to exit qmapmgr)
```

quit

Description

Exits from the NQS `qmapmgr` utility.

You can also exit `qmapmgr` by entering the `exit` command or by pressing **CTRL-d**.

Syntax

```
Quit
```

show

Description

Displays all entries in the NQS database in MID order.

Each entry includes the MID, machine name, and alias if assigned.

Syntax

```
Show
```

Output

The following example shows sample output from this command.

```
Mapmgr: show
Mid 1 = machine2, s
Mid 2 = machine1
Mid 3 = pluto, p
Mid 4 = test
Mid 5 = user1, u
```

NQS run-time directory hierarchy

A

This appendix describes the NQS run-time directory hierarchy. Table 4 provides information on NQS directories, file content, and usage.

Caution

Do not change access permissions on any of these directories.

Table 4 NQS run-time directory hierarchy

Directory	Description	Access mode
/var/spool/nqs	NQS directory tree	711
/var/spool/nqs/new	Parent of new request database hierarchy (mode 770 must be maintained to prevent unauthorized linking to and deleting of new request files)	770
/var/spool/nqs/new/requests	New request files	777
/var/spool/nqs/private	NQS private directories	771
/var/spool/nqs/private/root	NQS root directory	771
/var/spool/nqs/private/root/control	Parent of the request control file directories	771
/var/spool/nqs/private/root/control/*	Control subdirectories	1771
/var/spool/nqs/private/root/data	Parent of the request data files, such as print files and commands for batch requests	771
/var/spool/nqs/private/root/data/*	Data subdirectories	771

Table 4 NQS run-time directory hierarchy **(continued)**

Directory	Description	Access mode
/var/spool/nqs/private/root/database	NQS database files	771
/var/spool/nqs/private/root/database_qa	Queue access files	771
/var/spool/nqs/private/root/database_qo	Queue ordering files	771
/var/spool/nqs/private/root/failed	Saved copies of requests that caused problems	770
/var/spool/nqs/private/root/interproc	Interprocess communication files for completion code results from the NQS daemon to local client processes	773
/var/spool/nqs/private/root/output	Spooled output files and mail log files for NQS batch requests	771
/var/spool/nqs/private/root/output/*	Output subdirectories	1771
/var/spool/nqs/private/root/transact	Transaction files	771
/var/spool/nqs/private/root/transact/*	Transaction subdirectories	1777
/var/spool/nqs/scripts	Links to active shell scripts when the NQS shell strategy is set to free mode	771

Access permissions

NQS sets access permissions automatically and uses them to limit access to files. If access permissions are changed erroneously, please call the Hewlett-Packard Convex Technical Assistance Center (TAC) for instructions on restoring them.

For more information on the contents or access mode of any of these files, please contact the TAC.

/var/spool directory

Because it holds script files and job output until output is sent to the user, the /var/spool directory can become full. Consider creating a separate partition for /var/spool/nqs.

/database directory

The directory named database contains information for the `set` commands. If this or any other NQS database becomes corrupted, contact the TAC.

Output from failed jobs

While system managers and others logged in as root may use `cd` to move to these directories, many of the files are binary and cannot be read. You do not need to access them or remove anything from them, with the exception of the /failed directory, which contains output from failed jobs. Users whose output is sent to the /failed directory receive mail informing them of the failure and location of their output. The system manager may remove files from the /failed directory.

Look in the failed directory for output from jobs that fail or from jobs whose resource limits prevent correct output placement. You must be logged in as root to remove files from this directory.

NQS daemons

B

This appendix describes the daemons used by NQS.

A daemon is a process that provides a particular service when needed, but is not connected to a terminal or a particular user. NQS uses the following daemons:

`logdaemon`

Is contacted by `nqsdaemon` and `netdaemon` when they need to print an error message. `logdaemon` sends a message to `syslogd` (if defined) and notifies the NQS manager if the error is fatal.

`netclient`

Transfers jobs to remote machines and transfers `stderr` and `stdout` to appropriate destinations.

`netdaemon`

Handles all remote transactions involving queues, including submitting jobs and copying `stdout` and `stderr` files.

`netserver`

Is created by `netdaemon` to handle operations that require a substantial amount of time, such as queuing a request.

`nqsdaemon`

Handles all local transactions, including job submission and deletion, job scheduling, and system configuration.

`pipeclient`

Is created by `nqsdaemon` to route requests to the first queue in a pipe queue destination set that is able to accept the job.

`pipedemand`

Is created by `nqsdaemon` to route requests to the first queue in a pipe queue destination set that can immediately run the request.

`pipeldav`

Is created by `nqsdaemon` to route requests based on load factor.

`shepherd`

A child of the `nqsdaemon` that watches over batch jobs, `shepherd` is responsible for setting up the job environment and returning output files.

When you use `qps` to print information about NQS daemon processes, `shepherd` appears as `nqsdaemon` under COMMAND. The corresponding information under QUEUE and REQ distinguishes the `shepherd` process from the true `nqsdaemon` process.

Index

N

O

P

Q

R

S

U

B5589-90008