CONVEX CXbatch System Manager's Utilities Reference

Order No. DSW-184

Second Edition
September 1990

CONVEX Computer Corporation
Richardson, Texas USA

CONVEX
CXbatch System Manager's Utilities Reference

Order No. DSW-184
Document Number 710-006830-003

Printed in the United States of America.

Revision Information for

CONVEX CXbatch System Manager's Utilities Reference

Edition	Document No.	Description
Second	710-006830-003	Released with CONVEX CXbatch, V2.0, September 1990, contains new `qmgr` and `qmapmgr` reference pages.
First Edition, Rev. 1	710-006830-001	Released with CONVEX CXbatch, V1.1 April 1990.
First	710-000950-200	Released with CONVEX CXbatch, V1.0 February 1989.

Contents

Using this Document

Purpose and Audience

This document describes how to use the `qmgr` and `qmapmgr` utilities to establish and operate CXbatch.

This document is part of a four-volume set consisting of *CONVEX CXbatch Concepts, CONVEX CXbatch User's Guide, CONVEX CXbatch System Manager's Guide,* and *CONVEX CXbatch System Manager Utilities Reference.*

This document addresses:

- System managers who install, configure, and maintain CXbatch.
- CXbatch users who need to manage their own requests.

Organization

This manual is organized as follows:

- Chapter 1 provides an overview of `qmgr` and `qmapmgr`, their functionality, user interface, command format, and access levels.
- Chapter 2 is an alphabetical reference that describes each `qmgr` command within the `qmgr` utility and lists its format and parameters.
- Chapter 3 is an alphabetical reference that describes each `qmapmgr` command within the `qmapmgr` utility and lists its format and parameters.
- Appendix A provides instructions for reporting software or documentation problems to the CONVEX Technical Assistance Center (TAC).

Technical Assistance

If you have questions that are not answered in this book, contact the CONVEX Technical Assistance Center (TAC).

- Within the continental U.S., call 1(800)952-0379.
- From Canada, call 1(800)345-2384.
- All other locations, contact the local CONVEX office.

Reader Response

If you have comments or questions about the contents of this book, please notify the CONVEX documentation department by using the Reader's Forum at the end of this document.

Associated Documents

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

You can order the following documents from CONVEX Computer Corporation for more information on the ConvexOS operating system:

- *CONVEX UNIX Primer* (DSW-133) - introduces new users to the ConvexOS operating system.
- *ConvexOS Programmer's Reference* (DSW-332) - standard reference for the ConvexOS operating system.
- *Managing ConvexOS: Configuration Guide* (DSW- 030) - contains information needed to configure a CONVEX supercomputer.
- *Managing ConvexOS: Operations Guide* (DSW- 031) - contains information needed to operate a CONVEX supercomputer.
- *CONVEX Share Scheduler Concepts* (DSW- 261) - contains conceptual information on the CONVEX Share Scheduler.
- *CONVEX Share Scheduler System Mnager's Guide* (DSW- 265) - contains information needed to manage and maintain the Share Scheduler.
- *CONVEX Checkpoint Restart Guide* (DSW-350) - contains detailed information and instructions for using Checkpoint Restart.
- *ConvexOS Utilities User's Guide* (DSW- 005) - provides detailed information on the CONVEX Assembler, Loader, text editors, and `adb` (assembly language debugger.

Ordering Documentation

To order the current edition of this or any other CONVEX document, send requests to:
CONVEX Computer Corporation
Customer Service
PO Box 833851
Richardson, TX 75083-3851
USA

Include the order number or the exact title as listed on the front cover.

Notational Conventions

This section discusses notational conventions used in this guide.

Command Syntax

Consider this example:

COMMAND *input_file* `[...]` `{a | b}` [*output_file*]
① ② ③ ④ ⑤

1. **COMMAND** must be typed as it appears.
2. *input_file* indicates a file name that must be supplied by the user.
3. The horizontal ellipsis in brackets indicates that additional input file names may be supplied.
4. Either `a` or `b` must be supplied.
5. *output_file* indicates an optional file name.

General Conventions

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

- **`Bold constant-width font`** identifies user input in examples.
- *Italics*
  - Designate user-supplied variables in a command-line example.
  - Introduce new and important terms.
  - Identify variables in mathematical equations.
  - Indicate titles of documents.
- `Constant-width font` is used to designate input and output, including:
  - Command names and options.
  - System calls.
  - Data structures and types.
  - Directives, program statements, display examples, printout examples, and error messages returned.
- Horizontal ellipsis (...) shows repetition of the preceding item(s).
- Vertical ellipses show 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 that 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 that you type the command and then press **RETURN**.
- References to the *ConvexOS Programmer's Reference* appear in the form adb(1), where the name of the man page is followed by its section number enclosed in parentheses.

qmgr and qmapmgr

1

Introduction

Two utilities exist for establishing and operating CXbatch:

- The `qmgr` utility allows you to define and control the CXbatch queues.
- The `qmapmgr` utility allows you to configure the network database used by CXbatch to map between CXbatch and each machine in the configuration.

Each of these utilities is described in the remaining sections of this chapter. In addition, Chapter 2 describes each `qmgr` command in detail, and Chapter 3 describes each `qmapmgr` command in detail. This chapter describes:

- The `qmapmgr` and `qmgr` utilities.
- User interfaces to the utilities.
- The command formats within each utility.
- Access levels to the utilities.

qmgr

The `qmgr` utility is the utility system managers use to configure CXbatch. It is a program that controls CXbatch requests, queues, and the general CXbatch configuration on the local machine.

You may use the `qmgr` utility to:

- Abort queues.
- Add queues, pipe queue destinations, group permissions, managers, and user permissions.
- Create queues.
- Delete queues, pipe queue destinations, group permissions, managers, requests, and user permissions.
- Give queue access to Checkpoint Restart.
- Assign Share Scheduler shares to 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.

Detailed information on each of these functions can be found in the qmgr(8) man page, the on-line help facility within `qmgr`, and Chapter 2 of this manual.

User Interface

To access the `qmgr` utility, enter

```
% qmgr
```

You can also execute a `qmgr` command by entering

```
% qmgr command
```

where *command* is the specific `qmgr` command you want to execute.

A user defined as a CXbatch manager has access to all `qmgr` commands and can change any CXbatch characteristic on the local machine. For a list of all `qmgr` commands, see the appropriate man page, the on-line help, and Chapter 2 of this manual.

A user defined as a CXbatch operator has access to the following subset of `qmgr` commands.

Queue-related commands:

- `Abort queue` — Aborts all currently running requests in the queue.

❑ `Disable queue`	Prevents the queue from accepting requests.
❑ `Enable queue`	Allows queue to accept requests.
❑ `Move queue`	Moves all requests in one queue to another queue.
❑ `Purge queue`	Deletes all queued requests from the queue.
❑ `Start queue`	Puts queue into operation to allow jobs to run in it.
❑ `Stop queue`	Prevents queue from executing any other requests; the currently executing request is allowed to complete.

Request-related commands:

❑ `Delete request`	Deletes a specific request from a queue.
❑ `Hold request`	Puts request on hold, preventing it from running.
❑ `Modify request`	Modifies attributes of a request.
❑ `Move request`	Moves a request from one queue to another queue.
❑ `Release request`	Releases request on hold, making it eligible for running.
❑ `Resume request`	Resumes execution of suspended requests.
❑ `Suspend request`	Suspends execution by checkpointing, then kills request, which stays suspended in the queue.

Miscellaneous commands:

❑ `Exit`	Exits `qmgr`.
❑ `Help`	Displays help screens.
❑ `Set global per-user limit`	Establishes global run limits per user.
❑ `Set per-user run limit`	Establishes the per user run limit on specified queue.
❑ `Set run limit`	Establishes the run limit of an existing queue.
❑ `Show`	Displays information about CXbatch.
❑ `Shutdown`	Shuts down CXbatch on the local machine.

All other users of CXbatch have access to the following subset of `qmgr` commands:

❑ `Chkpnt request`	For their requests only.
❑ `Delete request`	For their own requests only.
❑ `Exit`	For all users.
❑ `Help`	For all users.

❑ `Hold request`	For their own requests only.
❑ `Modify request`	For their own requests only.
❑ `Move my_request`	For their own requests only.
❑ `Release request`	For their own requests only.
❑ `Show`	All show commands.

Command Format

To execute a command from within `qmgr` or `qmapmgr`, enter

```
Mgr: keyword parameter
```

where `keyword` is the specific `qmgr` or `qmapmgr` command and *parameter* is the additional information required by the specific command. An example command for `qmgr` is

```
Mgr: delete queue Queue1
```

where **`delete queue`** is the `keyword` and *Queue1* is the *parameter.*

Keywords may be entered in either uppercase or lowercase characters, but any queue names created in lowercase must be entered in lowercase. You must enter each command on a single command line.

For a full list and description of `qmgr` and `qmapmgr` commands, see the appropriate man pages, on-line help, and Chapters 2 and 3 of this manual.

Access Privileges

A user with the proper privileges may access `qmgr`. The system manager can limit access to the separate `qmgr` commands by using the CONVEX operating system `op` utility or by using the `add manager` command from within `qmgr`. The following sections describe access to each utility.

op Utility

You can set access privileges for `qmgr` commands with the ConvexOS `op` utility. This utiltiy is a binary executable that allows you to specify classes of users and their access to certain commands. The `op` utility reads the /etc/op.access file to determine whether a user can execute a specified command. An entry in the /etc/op.access file might look like the following:

```
enableq /usr/convex/qmgr enable queue $1; users=batchman, batchop
```

This entry grants access to the `enable queue` command to users `batchman` and `batchop`.

You can set access to each `qmgr` function separately, or you can group functions together. You can grant permission to any set of trusted users to execute certain `qmgr` commands without giving them full superuser privileges. (For more information, see the op(8) and op.access(5) man pages.)

If you use the `op` utility, make sure that the `qmgr` commands are secure before granting `qmgr` privileges to groups. Use this utility if you want to limit access to specific commands within `qmgr`.

add managers Command

The second method of granting access is the `add manager` command within `qmgr`. It defines CXbatch managers and operators and may be used instead of the `op` utility. This command is described in detail in the qmgr(8) man page and in Chapter 2 of this manual.

If the `add managers` command defines a user as a manager, it grants access to all `qmgr` commands to that user. If it defines a user as an operator, it grants access to only those commands appropriate for an operator. See the section titled "User Interface," for a list of operator commands.

qmapmgr

The qmapmgr utility builds and maintains the network database used by CXbatch. This database contains information on machine identification numbers (MIDs) and machine names; its existence is mandatory for the proper operation of CXbatch. qmapmgr establishes a mapping betweenMIDs and host names. Without this mapping, CXbatch does not recognize local or remote queues.

You can add to, delete from, and change the database using qmapmgr once the database has been established.

Detailed information on qmapmgr can be found in the qmapmgr(8) man page and Chapter 3 of this manual.

User Interface

To gain access to the qmapmgr utility, enter

```
% qmapmgr
```

You can also execute a qmapmgr command by entering

```
% qmapmgr command
```

where *command* is the specific qmapmgr command you want to execute.

Command Format

To execute a command from within qmapmgr, enter

```
Mapmgr: keyword parameter
```

where keyword is the specific qmapmgr command and *parameter* is the additional information required by the specific command. An example command for qmapmgr is

```
Mapmpgr: get name 99
```

where get name is the keyword and *99* is the *machine id*.

Keywords may be entered in either uppercase or lowercase characters, but any queue names created in lowercase must be entered in lowercase. You must enter each command on a single command line.

For a full list and description of the qmapmgr commands, see the appropriate man page and Chapter 3.

Access Privileges

As system manager, you must have write access to the files in the /etc/nmap directory to be able to change any of the values in the database. When CXbatch is installed, this access is limited to the superuser. Anyone with read access to the files can use the commands that display information in the database. When CXbatch is installed, all users have read access to these files.

You can also use the op utility to limit access to qmapmgr commands.

qmgr Commands 2

Basic Information

This chapter contains a description of each `qmgr` command. Each command description includes

- Definition of command syntax.
- Description of command parameters.
- Examples.

The first two or three characters of each word in each command are unique, and you need to enter only those characters for CXbatch to recognize the command. These accepted abbreviations are indicated in the "Format" section of each command description as uppercase letters. Commands may be entered in any combination of uppercase and lowercase characters.

ABORT QUEUE

abort all requests in the specified queue

Format:

ABort Queue *queuename* [*seconds*]

Description:

The ABORT QUEUE command aborts all running requests in the specified queue. CXbatch sends a SIGTERM signal to each process of each request presently running in the named queue. After the time indicated in *seconds* has passed, CXbatch also sends a SIGKILL signal to all remaining processes. If you omit *seconds*, CXbatch assumes a default value of 60 seconds.

Operator privileges are required to use this command.

Parameters:

queuename

This is the name of the queue that is aborted.

seconds

This is the amount of real time CXbatch waits before sending a SIGKILL signal.

ADD ALIAS

add an alias to a queue

Format: **ADd Alias** *alias queuename*

Description: The ADD ALIAS command adds an alias tothe specified queue. A queue alias is an alternate name for a queue; any CXbatch command that requires a queue name also works with an alias. Full CXbatch manager privileges are required to use this command.

Parameters:

alias

This is the alias that is assigned to the queue.

queuename

This is the name of the queue that is assigned an alias.

ADD DESTINATION

add destination queue(s) to pipe queue destination set

Format:

ADd DEStination = (*destination* [,...]) *queuename*

ADd DEStination = *destination* *queuename*

Description:

The ADD DESTINATION command adds one or more destination queues to an existing destination set for a local pipe queue. Any machine name included in the destination queue name must be defined in the local system's network host table. Full CXbatch manager privileges are required to use this command.

Parameters:

destination

This is the name of the destination queue that is added. 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 *remote_queue_name* is the name of the destination queue and *remote_machine_mid* is the machine ID (entered as an integer) of the remote machine. If the group ID is used, it must be enclosed in square brackets.

queuename

This is the name of the pipe queue.

ADD GROUPS

add group(s) to existing access list

Format:

ADd Groups = (*group* [,...]) *queuename*

ADd Groups = *group queuename*

Description:

The ADD GROUPS command adds one or more groups to the existing list of those groups that are allowed access to a queue. The queue must be set to "no access" to add groups.

Full CXbatch manager privileges are required to use this command.

Parameters:

group

This is the name of the group that is added to the access list. *group* can be either the group name or the numerical group ID as defined by the system manager. If the group ID is used, it must be enclosed in square brackets.

queuename

This is the name of the queue to which access is granted.

ADD MANAGERS

grant a user access to `qmgr` commands

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

Description: The ADD MANAGERS command grants a user access to `qmgr` commands.

Full CXbatch manager privileges are required to use this command.

Parameters: *username*

This is the user name that is granted access. The syntax of username must be in one of the following forms:

local_account_name

[*local_account_id*]

[*remote_user_id*]@ *remote_machine_name*

remote_user_id@[*remote_machine_mid*]

If the group ID is used, it must be enclosed in square brackets

option

This designates whether the user is granted manager or operator privileges. `m` indicates that the user is granted manager access, and `o` indicates that the user is granted operator access. The colon and option are required.

ADD USERS

add user(s) to the existing access list

Format:

ADd Users = (*user* [,...]) *queuename*

ADd Users = *user queuename*

Description:

The ADD USERS command adds one or more users to the list of those users who are allowed access to the named queue. You cannot add a user when queue access is unrestricted. Users can be added only when the access has been set to 'no.'

Full CXbatch manager privileges are required to use this command.

Parameters:

user

This is the user name that is added to the access list. This can be either the user name or the user ID as defined by the system manager. If the user ID is used, it must be enclosed in square brackets.

queuename

This is the name of the queue to which access is granted.

CHKPNT REQUEST

checkpoint a running request

Format: **CHkpnt Request** *request_id* [...]

Description: The CHECKPOINT REQUEST command checkpoints the request(s) named by the request id. This command saves the state of the request in a set of checkpoint files for restart later.

Full CXbatch manager privileges are required to use this command.

Parameters: *request-id*

This is the ID number of the request that is deleted.

CREATE BATCH_QUEUE

create a new CXbatch batch queue

Format:

CReate Batch_queue *queuename* **PRiority** = *priority* **[PIpeonly] [Run_limit** = *run-limit*] **[Import_dir** = *import-option*] **[Share_policy Fixed** = *username*] **[Share_policy User]**

Description:

The CREATE BATCH_QUEUE command defines a new CXbatch batch queue. If you have installed the CONVEX Share Scheduler (even if it is not running), you must first create a /etc/passwd entry for the queue on which you want to use `Share Policy = Fixed` (the default share policy). Refer to the *CONVEX CXbatch System Manager's Guide* for more information.

Full CXbatch manager privileges are required to use this command.

Parameters:

queuename

This is the name of the queue that is created. This name can consist of any printable non-blank character except for the following: @, comma, equal sign, left or right parenthesis. The queue name must not start with a digit (0-9).

PRiority = *priority*

This is the inter-queue priority assigned to the new queue. This priority affects the order in which CXbatch assigns requests to queues. It is an unsigned integer in the range 0-63. A value of 63 is the highest inter-queue priority, and 0 is the lowest.

PIpeonly

This indicates that the queue may only accept requests submitted from a pipe queue. If you omit *PIpeonly*, the queue may accept requests from any source.

Run_limit = *run-limit*

This establishes the maximum number of requests that may run in the queue at any given time. If you omit *run-limit*, the value defaults to one.

Import_dir = *import-option*

This determines whether CXbatch imports the current working directory before processing the request. *import-option* can be one of the following:

Yes	The queue imports the current working directory unless the `-ni` option is used when a request is submitted.
Available	The queue can import the current working directory, but it only does so if the `-i` option is used when a request is submitted.
No	The queue cannot import the current working directory, and it rejects any jobs that require input files to be imported.

When importing, if a request does not originate from the machine where the files are located, CXbatch tries to gain access to the remote directory using NFS mounts.

Share_policy Fixed = *username*

This charges the queue's CPU resource usage to the specified account, *username*, on the local machine on a per-queue basis.

Share_policy User

This charges the queue's CPU resource usage to the request submitter on a per-queue basis.

CREATE PIPE_QUEUE

create a new CXbatch pipe queue

Format:

CReate Pipe_queue *queuename* **PRiority=***priority* **Server=(***server***)[Destination=(** *destination*[,...])**[PIpeonly][Run_limit=***run-limit*]

Description:

The CREATE PIPE_QUEUE command defines a new CXbatch pipe queue. Full CXbatch manager privileges are required to use this command.

Parameters:

queuename

This is the name of the queue that is created. This name can consist of any printable non-blank character except for the following: @, comma, equal sign, left or right parenthesis. It must not start with a digit (0-9).

PRiority = *priority*

This is the inter-queue priority assigned to the new queue. This priority affects the order in which CXbatch assigns requests to queues. It is an unsigned integer in the range 0-63. A value of 63 is the highest inter-queue priority, and 0 is the lowest.

Server = (*server*)

This is the name of the server that transports requests submitted to this queue to one of the destination queues. This includes the associated server command line arguments. The servers provided are /usr/lib/nqs/pipeclient and usr/lib/nqs/pipeldav. See the CONVEX CXbatch Concepts, the CONVEX CXbatch System Manager's Guide, or the pipeclient (8) man page for more information.

Destination = (*destination*)

This is the name of the destination queue(s) to which this pipe queue may send requests. The syntax of the destination queue(s) name must be in one of the following forms:

local_account_name

[*local_account_id*]

[*remote_queuename*]@ *remote_machine_name*

[*remote_queuename*]@[*remote_machine_mid*]

where *remote_queue_name* is the name of the destination queue and

remote_machine_mid is the machine ID (entered as an integer) of the remote machine. The brackets are required.

PIpeonly

This option indicates that the queue may only accept requests submitted from a pipe queue. If you omit `PIpeonly`, the queue may accept requests from any source.

Run_limit = *run-limit*

This establishes the maximum number of requests that may run in the queue at any given time. If you omit *run-limit*, the value defaults to one.

DELETE ALIAS

delete an alias from a queue

Format: **DElete Alias** *alias*

Description: The DELETE ALIAS command deletes an alias from a queue. A queue alias is an alternate name for a queue; any CXbatch command that requires a queue name also works with an alias.

Full CXbatch manager privileges are required to use this command.

Parameters: *alias*

This is the alias that is deleted from the queue.

DELETE DESTINATION

delete destination queue(s)

Format:

DElete DEStination = (*destination* [,...]) *queuename*

DElete DEStination = *destination queuename*

Description:

The DELETE DESTINATION command deletes one or more destination queues from an existing destination set for a local pipe queue.

Any request being transferred to the deleted destination is allowed to complete successfully.

If all destinations for a pipe queue are deleted in this manner, the pipe queue is effectively stopped, although its actual status remains unchanged. Thus, the addition of a new destination for a pipe queue that has been effectively stopped in this manner immediately starts the queue running again.

Full CXbatch manager privileges are required to use this command.

Parameters:

destination

This is the name of the destination queue that is deleted. The syntax for the name must be in one of the following forms:

local_account_name

[*local_account_id*]

[*remote_queue_name*]@ *remote_machine_name*

[*remote_queue_name*]@[*remote_machine_mid*]

where *remote_queue_name* is the name of the destination queue and *remote_machine_mid* is the machine ID (entered as an integer) of the remote machine.

queuename

This is the name of the pipe queue that contains the destination queue in its destination set.

DELETE GROUPS

delete group(s) from existing access list for queue

Format:

DElete Groups = (*group*[,...]) *queuename*

DElete Groups = *group queuename*

Description:

The DELETE GROUPS command deletes one or more groups from the existing list of those groups allowed access to a queue. You can only use this command to delete groups that were added with the ADD GROUPS command. You cannot delete a group when queue access is unrestricted.

Full CXbatch manager privileges are required to use this command.

Parameters:

group

This is the name of the group that is deleted from the access list. This can be either the group name or the numeric group ID as defined by the system manager. If the group ID is used, it must be enclosed in square brackets.

queuename

This is the name of the queue to which access is denied.

DELETE MANAGERS

delete manager(s) from existing access list

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

Description: The DELETE MANAGERS command deletes one or more managers from the CXbatch manager access list. You cannot delete the superuser account (root) from the CXbatch manager set.

Full CXbatch manager privileges are required to use this command.

Parameters: *username*

This is the user name that is deleted from the access list. The syntax of *username* must be in one of the following forms:

local_account_name

[*local_account_id*]

[*remote_user_id*]@ *remote_machine_name*

[*remote_user_id*]@[*remote_machine_mid*]

The brackets are required.

option

This designates whether the user was granted manager or operator privileges. *m* indicates that the user was granted manager access, and *o* indicates that the user was granted operator access. The colon and option are required.

DELETE QUEUE

delete the named queue

Format: **DElete Queue** *queuename*

Description: The DELETE QUEUE command deletes the named queue. To delete a queue, no requests can be present in the queue, and the queue must be disabled.

Full CXbatch manager privileges are required to use this command.

Parameters: *queuename*

This is the name of the queue that is deleted.

DELETE REQUEST

delete request(s) from local CXbatch machine

Format:

DElete Request *request-id* [...]

Description:

The DELETE REQUEST command deletes one or more requests from the local CXbatch machine. If the specified request is running, then CXbatch sends a SIGKILL signal to all processes in the request. If the request is not running, then it is removed from the queue and discarded.

CXbatch operator privileges are required to use this command to delete requests owned by another user.

Parameters:

request-id

This is the ID number of the request that is deleted.

request-id is in the format *seqno* or *seqno .hostname*, where *seqno* is the sequence number that CXbatch assigned to the request and *hostname* is the name of the originating local machine.

DELETE USERS

delete user(s) from the existing access list

Format:

DElete Users = (*user* [,...]) *queuename*

DElete Users = *user queuename*

Description:

The DELETE USERS command deletes one or more users from the list of those users allowed access to the named queue. You can only use this command to delete users if they were added with the ADD USERS command. You cannot delete a user when queue access is unrestricted.

Full CXbatch manager privileges are required to use this command.

Parameters:

user

This is the user name that is deleted from the access list. This can be either the user name or the user ID as defined by the system manager. If the user ID is used, it must be enclosed in square brackets.

queuename

This is the name of the queue to which access is removed.

DISABLE QUEUE

prevent queue from accepting requests

Format: **DIsable Queue** *queuename*

Description: The DISABLE QUEUE command prevents any more requests from being placed in the named queue.

Full CXbatch operator privileges are required to use this command.

Parameters: *queuename*

This is the name of the queue that is disabled.

ENABLE QUEUE

enable queue to accept new requests

Format:

ENable Queue *queuename*

The ENABLE QUEUE command enables the queue to accept new requests. A queue cannot accept new requests unless it is enabled. If the specified queue is already enabled, no operation is performed.

Full CXbatch operator privileges are required to use this command.

Parameters:

queuename

This is the name of the queue that is enabled.

EXIT

exit from `qmgr` utility

Format: **EXit**

Description: The EXIT command is used to exit from the CXbatch `qmgr` utility program. The `qmgr` utility is also ended if an end-of-file is encountered in the standard input file. Thus, you can also exit `qmgr` by pressing **CTRL-D** if your end-of-file character is **CTRL-D**.

Parameters: None

HELP

invoke the `qmgr` HELP facility

Format: **HElp** [*command*]

Description: The HELP command invokes the `qmgr` HELP facility and displays information about a `qmgr` command or topic.

If you do not specify a topic when you issue the HELP command, `qmgr` displays information describing what commands are available.

Because the first two or three characters of each word in a command uniquely identify it to `qmgr`, you need only enter those characters.

Parameters: *command*

This is the command for which you want more information.

HOLD REQUEST

put request(s) on hold, preventing their execution

Format: **HOld Request** *request-id* [...]

Description: The HOLD REQUEST command puts one or more requests on hold, preventing their execution. The request remains in the queue in a hold status until it is released with the RELEASE REQUEST command. Requests that are already running cannot be put on hold; only requests in a queued state can be put on hold.

CXbatch operator privileges are required to use this command with any request that is not your own. If a request is put on hold by a batch manager or operator, only a batch manager or operator can release it.

Parameters: *request-id*

This is the ID number of the request that is put on hold. *request-id* is in the format *seqno* or *seqno .hostname*, where *seqno* is the sequence number that CXbatch assigned to the request and *hostname* is the name of the originating machine.

MODIFY REQUEST

modify the attributes of one or more requests

Format:

MODify Request [*attribute*] = <*value*> *request-id ...*

Description:

The MODIFY REQUEST command is used to modify the priority of one or more requests. Changing the priority allows jobs to be re-ordered in the queue, so that they run in a different order. Refer to `Set Maximum Request Priority`. Running requests cannot have their request attributes changed.

An operator or manager may raise or lower a request's priority. A user may only lower a priority.

Parameters:

value

This is the value that is to be assigned to the given attribute. Valid values vary depending on the attribute.

request-id

This is the ID number of the request that is modified. *request-id* is in the format *seqno* or *seqno.hostname*, where *seqno* is the sequence number that CXbatch assigned to the request and *hostname* is the name of the originating machine.

MOVE MY REQUEST

move your request(s) to another queue

Format:

MOVe My_request *request-id* [...] *queuename*

Description:

The MOVE MY_REQUEST command moves one or more of your requests from their current queue to the indicated queue. CXbatch checks all queue limits, attributes, and access restrictions before moving the request. If there are any violations, the request is not moved.

CXbatch operator privileges are required to use this command to move any requests other than your own.

Parameters:

request-id

This is the ID number of the request that is moved.

request-id is in the format *seqno* or *seqno .hostname*, where *seqno* is the sequence number that CXbatch assigned to the request and *hostname* is the name of the originating local machine.

queuename

This is the name of the target queue to which the request is moved.

MOVE QUEUE

move all requests from one queue to another

Format: **MOVe Queue** *queuename1 queuename2*

Description: The MOVE QUEUE command moves all non-running requests currently in *queuename1* to *queuename2*. CXbatch moves the requests regardless of any queue limit violations, access restrictions, or attribute violations.

CXbatch operator privileges are required to use this command.

Parameters: *queuename1*

This is the name of the source queue from which all requests are moved.

queuename2

This is the name of the target queue to which all requests are moved.

MOVE REQUEST

move request(s) to another queue

Format:

MOVe Request *request-id* [...] *queuename*

Description:

The MOVE REQUEST command moves one or more non-running requests to the indicated queue. CXbatch does not check for any queue limit violations, access restrictions, or attribute violations at the indicated queue before moving the request.

CXbatch operator privileges are required to use this command.

Parameters:

request-id

This is the ID number of the request that is moved.

request-id is in the format *seqno* or *seqno .hostname*, where *seqno* is the sequence number that CXbatch assigned to the request and *hostname* is the name of the originating local machine. Batch jobs can be moved; pipe jobs cannot.

queuename

This is the name of the target queue to which the request is moved.

PURGE QUEUE

remove all requests not executing from the queue

Format: **Purge Queue** *queuename*

Description: The PURGE QUEUE command removes all requests that are not running (e.g., in the queued or holding states) from the named queue. Purged requests are irretrievably lost.

CXbatch operator privileges are required to use this command.

Parameters: *queuename*

This is the name of the queue that is purged.

RELEASE REQUEST

release request(s) from hold, making them eligible for execution

Format: **RELease Request** *request-id* [...]

Description: The RELEASE REQUEST command releases one or more requests previously placed on hold and makes them eligible for execution. The request must be in a holding state prior to being released.

CXbatch operator privileges are required to use this command with any request that is not your own. If a request is put on hold by a batch operator or manager, only a batch operator or manager can release it.

Parameters: *request-id*

This is the ID number of the request that is released. *request-id* is in the format *seqno* or *seqno .hostname*, where *seqno* is the sequence number that CXbatch assigned to the request and *hostname* is the name of the originating local machine.

RESTART REQUEST

restarts checkpointed request

Format: **REStart Request** *request-id* [...]

Description: The RESTART REQUEST command restarts a checkpointed request. The request begins execution from its checkpointed state.

CXbatch operator privileges are required to use this command.

Parameters: *request-id*

This is the ID number of the request that is resumed. *request-id* is in the format *seqno* or *seqno.hostname*, where *seqno* is the sequence number that CXbatch assigned to the request and *hostname* is the name of the originating local machine.

RESUME REQUEST

resumes execution of suspended request

Format: **RESume Request** *request-id* [...]

Description: The RESUME REQUEST command resumes execution of a suspended request, making the request eligible to run. When the request is about to be rerun, it starts execution from its suspended state.

CXbatch operator privileges are required to use this command.

Parameters: *request-id*

This is the ID number of the request that is resumed. *request-id* is in the format *seqno* or *seqno.hostname*, where *seqno* is the sequence number that CXbatch assigned to the request and *hostname* is the name of the originating local machine.

RUN REQUEST

force listed request(s) to begin executing immediately

Format: **RUn Request** *request-id* [...]

Description: The RUN REQUEST command forces listed requests to immediately begin executing. The run limit for the queue in which the requests resides is increased by one while the request is executing.

CXbatch operator privileges are required to use this command.

Parameters: *request-id*

This is the ID number of the request that is released. *request-id* is in the format *seqno* or *seqno .hostname*, where *seqno* is the sequence number that CXbatch assigned to the request and *hostname* is the name of the originating local machine.

SET AID_MASK

set the activity ID mask

Format: **SEt AId_mask** = *mask-value*

Description: The SET AID_MASK command sets the activity ID mask. The activity ID of a request is calculated by taking the submitter's activity ID, subtracting the modulus of the submitter's activity ID and the activity ID mask, and adding the queue activity ID offset. The formula for calculating the activity ID of a request is as follows:

$$request\_aid = submitter\_aid - (submitter\_aid\%aid\_mask + queue\_aid\_offset)$$

Full CXbatch manager privileges are required to use this command.

Parameters: *mask-value*

This is the activity ID (AID) mask. Usually, this mask is the same as the spacing between the activity IDs in the /etc/activities file.

SET ACC_LOGFILE

change the name of the accounting log file

Format: **SEt ACC_logfile** *logfile-name*

Description: The SET ACC_LOGFILE command changes the file name that is used for CXbatch accounting.

Full CXbatch manager privileges are required to use this command.

Parameters: *logfile-name*

This is the name of the new file that is used for CXbatch batch accounting.

SET ACCOUNTING

turn accounting on or off for a batch queue

Format:

SEt ACCOunting = *switch queuename*

Description:

The SET ACCOUNTING command turns accounting *ON* or *OFF* for a CXbatch batch queue.

Full CXbatch manager privileges are required to use this command.

Parameters:

switch

This tells CXbatch to turn the accounting on or off. The valid options are *OFF* and *ON*.

queuename

This is the name of the queue that has accounting enabled or disabled. This queue must already exist.

SET ACTIVITY_ID_OFFSET

set the activity ID offset for a batch queue

Format: **SEt ACTivity_id_offset** = *offset-value queuename*

Description: The SET ACTIVITY_ID_OFFSET command establishes the activity ID offset for a CXbatch batch queue. The queue named as a parameter of the command must already exist.

Full CXbatch manager privileges are required to use this command.

Parameters: *offset-value*

This is the number added to the user's activity ID to create the request ID.

queuename

This is the name of the queue that has the offset.

SET CHECKPOINT DIRECTORY

set the directory to hold checkpoint restart files

Format: **SEt CHEckpnt_directory** *directory-name*

Description: The SET CHECKPOINT_DIRECTORY command informs CXbatch that checkpoint files created by the batch system should be stored in the directory specified, or changes the directory in which CXbatch stores checkpoint files. The checkpoint-directory must already exist. No checkpointing can take place until a checkpoint directory is specified. This directory must be readable by all users.

Full CXbatch manager privileges are required to use this command.

Parameters: *directory-name*

This is the name of the directory created to hold checkpoint files.

SET CHKPNTABLE

set the checkpointable attribute of the queue

Format: **SEt CHKpntable =** *option queuename*

Description: The SET CHKPNTABLE command sets the checkpointable attribute of a queue. Full CXbatch manager privileges are required to use this command.

Parameters:

queue

This is the name of the queue on which the attribute is being set.

option

These are the attribute options:

Yes	The queue imports the current working directory unless the `-ni` option is used when a request is submitted.
Available	The queue can import the current working directory, but it only does so if the `-i` option is used when a request is submitted.
No	The queue cannot import the current working directory, and it rejects any jobs that require input files to be imported.

SET COREFILE_LIMIT

set per-process maximum core file size limit for queue

Format:

SEt CORefile_limit = *limit queuename*

Description:

The SET COREFILE_LIMIT command sets the per-process maximum core file size limit for any batch request subsequently placed in the named batch queue. This limit defines the largest size of a core file that can be created for a process. CXbatch uses this limit when you do not include the `-lc` option with the `qsub` command.

When a request is submitted to a queue, CXbatch checks the request limit to ensure that no user-specified core file size limit for the request exceeds the core file size limit (if configured) for the indicated batch queue.

If an attempt is made to queue a batch request that asks for a core file size limit larger than the destination batch queue has defined as allowable, the request is rejected.

The maximum core file size limit assigned to a particular request is determined at the time that the request is queued. If the core file size limit for the containing batch queue is later reduced, no requests in the batch queue are affected. However, if the new core file size limit value is less than the core file size limit requested by a previously queued request, then a warning message is displayed.

The core file size of any process in the running request cannot exceed the maximum in force when the request was queued.

Full CXbatch manager privileges are required to use this command.

Parameters:

limit

This is the maximum size in bytes for a core file produced by any process in any request. It is a single integer representing less than 100,000,000 bytes with an optional fractional part. The syntax for *limit* is one of the following forms:

integer [. *fraction*] [*units*]

where *integer* and *fraction* are strings of up to eight decimal digits and *units* is one of the following:

bbytes
wwords
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)

If you omit *units*, byte is assumed.

You may specify that no limit should be applied by using *unlimited* for the value of *limit*.

queuename

This is the name of the queue to which *limit* is assigned.

SET DATA_LIMIT

set per-process maximum data-segment size limit for queue

Format:

SEt DAta_limit = *limit queuename*

Description:

The SET DATA_LIMIT command sets the per-process maximum data-segment size limit for any batch request subsequently placed in the named batch queue. This limit defines the largest size of a data segment that can be created for a process. CXbatch uses this limit when you do not include the `-ld` option with the `qsub` command.

When a request is submitted to a queue, CXbatch checks the request limits to ensure that no user-specified data-segment size limit for the request exceeds the data-segment size limit (if configured) for the indicated batch queue.

If an attempt is made to queue a batch request that asks for a data segment size limit larger than the destination batch queue has defined as allowable, then the request is rejected.

The maximum data segment size limit assigned to a particular request is determined at the time that the request is queued. If the data segment size limit for the containing batch queue is later reduced, no requests in the batch queue are effected. However, if the new data segment size limit is less than the data segment size limit requested by a previously-queued request, a warning message is displayed.

The data segment size of any process in the running request cannot exceed the maximum in force when the request was queued.

Full CXbatch manager privileges are required to use this command.

Parameters:

limit

This is the maximum size for a data segment produced by any process in any request. It is a single integer representing less than 100,000,000 bytes with an optional fractional part. The syntax for *limit* is one of the following forms:

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)

If you omit units, byte is assumed.

You may specify that no limit should be applied by using *unlimited* for the value of *limit*.

queuename

This is the name of the queue to which *limit* is assigned.

SET DEBUG

enable or disable CXbatch debugging output

Format: **SEt DEBug** *level*

Description: The SET DEBUG command enables or disables CXbatch debugging output.

Currently, only three levels of debugging are supported: no debugging at all, minimum information displayed, and maximum information displayed. A debug value of 0 turns off debugging information, while a nonzero value enables debugging.

Full CXbatch manager privileges are required to use this command.

Parameters: *level*

This is the amount of debug information that is displayed. The three options are:

- 0 No debugging information displayed.
- 1 Minimum debugging information displayed.
- 2 Maximum debugging information displayed.

SET DEFAULT BATCH_REQUEST PRIORITY

set the default batch request intra-queue priority

Format:

SEt DEFault Batch_request Priority *priority*

Description:

The SET DEFAULT BATCH_REQUEST PRIORITY command sets the default batch request intra-queue priority. CXbatch uses this priority when you do not include the -p option with the qsub command.

The priority selected is termed the "intra-queue priority" because the selected priority determines the relative ordering of requests within the selected batch queue, not the execution-time priority of the request.

Full CXbatch manager privileges are required to use this command.

Parameters:

priority

This is the default priority value. It must be an integer in the range 0-63. A value of 63 is the highest intra-queue priority and 0 is the lowest.

SET DEFAULT BATCH_REQUEST QUEUE

set the default queue used for batch requests

Format: **SEt DEFault Batch_request Queue** *queuename*

Description: The SET DEFAULT BATCH_REQUEST QUEUE command sets the default queue to be used for batch requests. CXbatch uses this queue when you do not include the `-q` option with the `qsub` command.

Full CXbatch manager privileges are required to use this command.

Parameters: *queuename*

This is the name of the queue that serves as the default batch queue.

SET DEFAULT DESTINATION_RETRY TIME

set default total time allowed for resubmission of request

Format: **SEt DEFault DEStination_retry Time** *retry-time*

Description: The SET DEFAULT DESTINATION_RETRY TIME command sets the default number of hours that a pipe queue destination can be unreachable before the request fails.

Full CXbatch manager privileges are required to use this command.

Parameters: *retry-time*

This is the amount of time in hours that can elapse while CXbatch tries to resubmit a request to a destination queue through a pipe queue. After this time, the request is marked failed.

SET DEFAULT DESTINATION_RETRY WAIT

set default time to wait before resubmitting a request

Format:

SEt DEFault DEStination_retry Wait *wait-time*

Description:

The SET DEFAULT DESTINATION_RETRY WAIT command sets the default number of minutes to wait before retrying a pipe queue destination that was unreachable at the time of the last attempt.

When a pipe queue destination fails to accept a request because of a network failure or remote server failure, the request is not transferred, and the destination is disabled for the number of minutes set by *wait-time*. At the end of this time, the destination is re-enabled and retried as appropriate.

To prevent an infinite number of retries, you can use the command SET DEFAULT DESTINATION_RETRY TIME, which prevents a pipe queue destination from being endlessly retried.

Full CXbatch manager privileges are required to use this command.

Parameters:

wait-time

This is the amount of time in minutes that CXbatch waits before resubmitting a request to a destination queue through a pipe queue.

SET DESCRIPTION

change or delete the queue description

Format:

SEt DESCription = *description queuename*

SEt DESCription = *NULL queuename*

Description:

The SET DESCRIPTION command changes or deletes the description associated with the named queue. NULL is used to delete a description. The line length limit on description is 79 characters.

Full CXbatch manager privileges are required to use this command.

Parameters:

description

This is the new description of the queue.

queuename

This is the name of the queue whose description is being changed.

SET DESTINATION

create a new destination set for pipe queue

Format:

SEt DESTination = (*destination* [,...]) *queuename*

SEt DESTination = *destination queuename*

Description:

The SET DESTINATION command creates a new destination set for the named pipe queue. All previous destination queues in the pipe queue destination set are removed.

Any machine name included in the destination queue name must be defined in the local system's network host table, /etc/hosts.

Full CXbatch manager privileges are required to use this command.

Parameters:

destination

This is the name of the destination queue that is added. The destination queue name must be in one of the following forms:

local_account_name
[*local_account_id*]
[*remote_queue_name*]@ *remote_machine_name*
[*remote_queue_name*]@[*remote_machine_mid*]

where *remote_queue_name* is the name of the destination queue and *remote_machine_mid* is the machine ID (entered as an integer) of the remote machine.

queuename

This is the name of the pipe queue whose destination is being changed.

SET GLOBAL PER_USER RUN_LIMIT

set the global per-user run-limit

Format: **SEt Global Per_user Run_limit =** *run-limit*

Description: The SET GLOBAL PER_USER RUN_LIMIT prevents any one user from running more than *run-limit* requests simultaneously within the local CXbatch queues.

Full CXbatch manager privileges are required to use this command.

Parameters: *run-limit*

This is the total number of requests a user can have running at one time.

SET IMPORT_DIR

change the import attribute for a batch queue

Format: **SEt Import_dir** = *option queuename*

Description: The SET IMPORT_DIR command changes the import attribute for a batch queue. The queue must already exist.

When importing, if a request does not originate from the same machine, CXbatch tries to access the remote directory using NFS mounts.

NOTE

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

Full CXbatch manager privileges are required to use this command.

Parameters: *queuename*

This is the name of the queue whose attribute is changed. The queue must already exist.

option

This is the new import attribute for the queue. It can be one of the following:

Yes	The queue imports the current working directory unless the `-ni` option is used when a request is submitted.
Available	The queue can import the current working directory, but it only does so if the `-i` option is used when a request is submitted.
No	The queue cannot import the current working directory, and it rejects any jobs that require input files to be imported.

SET MAIL

set the account name appearing in "from:" portion of mail sent by CXbatch

Format: **SEt MAIl** *accountname*

Description: The SET MAIL command sets the account name that appears in the "from:" portion of mail sent by CXbatch. This mail notifies users (when appropriate) of events concerning their CXbatch request(s). The default account name is the superuser account.

Full CXbatch manager privileges are required to use this command.

Parameters: *accountname*

This is the user ID of the sender for CXbatch mail. It can consist of any printable nonblank character except for the following: @, comma, equal sign, left or right parenthesis. The user ID must not start with a digit (0-9).

SET MANAGERS

set the list of authorized `qmgr` managers and operators

Format:

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

Description:

The SET MANAGERS command sets the list of authorized `qmgr` managers and operators. CXbatch managers have full privileges for all `qmgr` commands, and CXbatch operators have privileges for a subset of `qmgr` commands. Existing authorizaions, other than root, will be deleted.

Full CXbatch manager privileges are required to use this command.

Parameters:

username

This is the user name that is granted access. The syntax of username must be in one of the following forms:

local_account_name
[*local_account_id*]
[*remote_user_id*]@ *remote_machine_name*
[*remote_user_id*]@[*remote_machine_mid*]

option

This designates whether the user is granted manager or operator privileges. *m* indicates that the user is granted manager access, and *o* indicates that the user is granted operator access. The colon and option are required.

SET MAXIMUM REQUEST_PRIORITY

set maximum priority of a request

Format:

SEt MAXimum Request_priority *priority queuename*

Description:

The SET MAXIMUM REQUEST_PRIORITY command assigns a (intra-queue) maximum request priority on a per-queue basis. The maximum request priority is a ceiling on the priority of requests submitted to that queue. A request that specifies a priority higher than the maximum for the queue will have its priority silently lowered to the queue maximum.

Full CXbatch manager privileges are required to use this command.

Parameters:

priority

This is the maximum request priority. It must be an integer in the range of 0-63. A value of 63 is the highest intra-queue priority, and 0 is the lowest.

queuename

This is the name of the queue to which the maximum priority is applied.

SET NICE_VALUE_LIMIT

set per-process nice value limit for queue

Format:

SEt NIce_value_limit = *nice-value queuename*

Description:

The SET NICE_VALUE_LIMIT command sets the per-process UNIX `nice` *value limit* for any batch request subsequently placed in the named batch queue. CXbatch uses this limit when you do not include the `-ln` option with the `qsub` command.

When a request is submitted to a queue, CXbatch checks the request limits to ensure that no user-specified *nice* value limit for the request exceeds the *nice* value limit (if configured) for the indicated batch queue. The user-specified *nice* value cannot be numerically less than the *nice* value as configured for the batch queue.

If an attempt is made to queue a batch request that asks for a smaller *nice* value limit than the destination batch queue has defined as allowable, then the request is rejected.

Full CXbatch manager privileges are required to use this command.

Parameters:

nice-value

This is the maximum *nice* value for any process in any request submitted to the queue. It is a single integer in the range -64 to 64.

queuename

This is the name of the queue that is assigned the *nice* value limit.

SET NO_ACCESS

delete all groups and users from existing access list for queue

Format: **SEt NO_Access** *queuename*

Description: The SET NO_ACCESS command deletes all groups and all users from the access list for the named queue, rendering it inaccessible by any user or group. Requests in the queue are allowed to remain. In order to enforce access restrictions on a queue where access is currently unrestricted, use SET NO_ACCESS first, then use ADD GROUPS or ADD USERS.

Superuser privileges are not affected by this command; the superuser always has access to all enabled queues, even in the absence of the appropriate entries in the access list.

Full CXbatch manager privileges are required to use this command.

Parameters: *queuename*

This is the name of the queue that is made inaccessible.

SET NO_DEFAULT BATCH_REQUEST QUEUE

indicates that no default batch queue exists

Format: **SEt NO_Default Batch_request Queue**

Description: The SET NO_DEFAULT BATCH_REQUEST QUEUE command indicates that there is to be no default batch queue for the CXbatch network.

Full CXbatch manager privileges are required to use this command.

Parameters: None

SET PER_PROCESS CPU_LIMIT

set per-process maximum CPU time limit for queue

Format:

SEt PER_Process Cpu_limit = (*limit*) *queuename*

Description:

The SET PER_PROCESS CPU_LIMIT command sets the per-process maximum CPU time limit for any batch request subsequently placed in the named batch queue. This limit defines the largest amount of CPU time each process can use. CXbatch uses this limit when you do not include the `-lt` option with the `qsub` command.

When a request is submitted to a queue, CXbatch checks the request limits to ensure that no user-specified CPU time limit for the request exceeds the CPU time limit (if configured) for the indicated batch queue.

If an attempt is made to queue a batch request that asks for a larger CPU time limit than the destination batch queue has defined as allowable, the request is rejected.

The maximum CPU time limit assigned to a particular request is determined at the time that the request is queued. If the CPU time limit for the containing batch queue is later reduced, no requests in the batch queue are affected. However, if the new CPU time limit value is less than the CPU time limit requested by a previously queued request, a warning message is displayed.

If the CPU time of any process created by the request becomes greater than this limit, CXbatch sends the signal SIGXCPU to the offending process. If the signal is not caught or is ignored, the process will exit.

CPU limit is recorded in a *seconds:minutes:hours* format. `qstat` by default shows CPU limit in *seconds*.

Full CXbatch manager privileges are required to use this command.

Parameters:

limit

This is the maximum CPU time allowed for any process in any request. The syntax for limit is as follows:

[[*hours* :] *minutes* :] *seconds* [. *milliseconds*]

White space may appear anywhere between the elements except around the decimal point. The accepted formats are as follows:

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)

Milleseconds are accepted but ignored on CONVEX systems.

queuename

This is the name of the queue to which the limit is assigned.

SET PER_PROCESS PERMFILE_LIMIT

set per-process maximum permanent file size limit for queue

Format:

SEt PER_Process Permfile_limit = (*limit*) *queuename*

Description:

The SET PER_PROCESS PERMFILE_LIMIT command sets the per-process maximum permanent file size limit for any batch request subsequently placed in the named batch queue. This limit defines the largest size of a permanent file that can be created by a process. CXbatch uses this limit when you do not include the `-lf` option with the `qsub` command.

When a request is submitted to a queue, CXbatch checks the request limits to ensure that no user-specified permanent file size limit for the request exceeds the permanent file size limit (if configured) for the indicated batch queue. If an attempt is made to queue a batch request that asks for a larger permanent file size limit than the destination batch queue has defined as allowable, then the request is rejected.

The maximum permanent file size limit assigned to a particular request is determined at the time that the request is queued. If the permanent file size limit for the containing batch queue is later reduced, no requests in the batch queue are affected. However, if the new permanent file size limit value is less than the permanent file size limit requested by a previously queued request, then a warning message is displayed.

If the file size of any process created by the request becomes greater than this limit, CXbatch sends the signal SIGXFSZ to the offending process. If the signal is not caught or is ignored, the process will exit.

Full CXbatch manager privileges are required to use this command.

Parameters:

limit

This is the maximum size in bytes for a permanent file produced by any process in any request. It is a single integer representing less than 100,000,000 bytes with an optional fractional part. The syntax for limit is one of the following forms:

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)
w	gigawords (2^{30} words)

If you omit *units*, byte is assumed. You may specify that no limit should be applied by using unlimited for the value of *limit*.

queuename

This is the name of the queue to which the limit is assigned.

SET PER_USER RUN_LIMIT

set the per-user run limit

Format: **SEt Per_user Run_limit =** *run-limit queuename*

Description: The SET PER_USER RUN_LIMIT command prevents any one user from running more than *run-limit* requests simultaneously within the specified queue. A per-user run-limit of 0 disables the enforcement of this limit.

Full CXbatch manager privileges are required to use this command.

Parameters: *run-limit*

This is the total number of requests a user can have running in a queue at all times.

queuename

This is the name of the queue that is assigned a user run limit.

SET PIPE_CLIENT

change pipe client associated with pipe queue

Format:

SEt PIpe_client = *client queuename*

Description:

The SET PIPE_CLIENT command changes the pipe client associated with the indicated pipe queue. The indicated queue cannot be a batch queue; it must be a pipe queue.

Full CXbatch manager privileges are required to use this command.

Parameters:

client

This is the name of the new pipe client that is associated with the indicated pipe queue. It defines the full path name of the pipe client, followed by any arguments required by the program. CONVEX supplies two pipe clients: /usr/lib/nqs/pipeclient and /usr/lib/nqs/pipeldav. Refer to *CONVEX CXbatch Concepts*, the *CONVEX CXbatch System Manager's Guide*, or the pipeclient(8) man page for more information.

queuename

This is the name of the queue that is associated with the pipe client. The queue cannot be a batch queue.

SET PRIORITY

change the inter-queue priority of existing queue

Format: **SEt PRiority** = *priority queuename*

Description: The SET PRIORITY command changes the inter-queue priority of an existing CXbatch queue. This priority affects the order in which CXbatch assigns request to queues.

Full CXbatch manager privileges are required to use this command.

Parameters: *priority*

This is the inter-queue priority assigned to the queue. It is an unsigned integer in the range 0-63. A value of 63 is the highest inter-queue priority, and 0 is the lowest.

queuename

This is the name of the queue that is assigned the priority.

SET RUN_LIMIT

change the run limit for existing queue

Format:

SEt Run_limit = *run-limit queuename*

Description:

The SET RUN_LIMIT command changes the run limit of an existing CXbatch queue. The run limit for a batch or pipe queue determines the maximum number of requests that can run in the queue at any given time.

CXbatch operator privileges are required to use this command.

Parameters:

run-limit

This establishes the maximum number of requests that can be run in the queue at any given time. If you omit *run-limit*, the value defaults to one.

queuename

This is the name of the queue that is assigned the run limit.

SET SHARE_POLICY FIXED

set a queue's share policy to charge a fixed uid's share group account

Format: **SEt SHAre_policy FIxed** = *username queuename*

Description: The SET SHARE_POLICY FIXED command specifies that CPU usage be charged to a specified account on the local machine on a per-queue basis.

Full CXbatch manager privileges are required to use this command.

Parameters:

username

This is the name of the user/account whose share polciy is set to `fixed`.

queuename

This is the name of the queue that is assigned the run limit.

SET SHARE_POLICY USER

set a queue's share policy to charge the request submitter

Format: **SEt SHAre_policy user** *queuename*

Description: The SET SHARE_POLICY USER command specifies that CPU usage be charged to a request's submitter on a per-queue basis.

Full CXbatch manager privileges are required to use this command.

Parameters: *username*

This is the name of the user/account whose share polciy is set to `user`.

SET SHELL_STRATEGY FIXED

set a specific shell to interpret shell script commands

Format:

SEt SHell_strategy FIxed = (*shell*)

Description:

The SET SHELL_STRATEGY FIXED command sets the indicated shell to be the shell to interpret the batch request shell script commands. If this strategy is selected, CXbatch uses the indicated shell to interpret shell scripts when you do not include the `-s` option with the `qsub` command.

Full CXbatch manager privileges are required to use this command.

Parameters:

shell

This is the absolute path name of the shell to interpret batch request shell script commands. The shell must exist and be executable; if not, you will get the error message: TCML_EACCESS.

SET SHELL_STRATEGY FREE

set the login shell as initial shell to interpret shell script commands

Format: **SEt SHell_strategy FRee**

Description: The SET SHELL_STRATEGY FREE command sets the user's login shell (as defined in the /etc/passwd file) as the initial shell to use to interpret commands. CXbatch then supplies the name of the script file to the login shell as standard input. Your login shell then reads the first line of the script file. If the first line specifies a shell (e.g., #!/bin/csh), that shell interprets the script file commands. Otherwise, *sh* is used. In other words, the request is interpreted as though it were run interactively.

CXbatch uses this shell strategy to interpret shell scripts when you do not include the `-s` option with the `qsub` command.

Full CXbatch manager privileges are required to use this command.

Parameters: None

SET SHELL_STRATEGY LOGIN

set the login shell to interpret all shell script commands

Format: **SEt SHell_strategy Login**

Description: The SET SHELL_STRATEGY LOGIN command sets the user's login shell as the shell to interpret the batch request shell script commands. This login shell is defined in the /etc/passwd file.

CXbatch uses this shell strategy to interpret shell scripts when you do not include the `-s` option with the `qsub` command.

Full CXbatch manager privileges are required to use this command.

Parameters: None

SET STACK_LIMIT

set per-process maximum stack size limit for queue

Format:

SEt STack_limit = (*limit*) *queuename*

Description:

The SET STACK_LIMIT command sets the per-process maximum stack size limit for any batch request subsequently placed in the named batch queue. This limit defines the largest size of a stack segment that can be created by a process. CXbatch uses this limit when you do not include the `-ls` option with the `qsub` command.

When a request is submitted to a queue, CXbatch checks the request limits to ensure that no user-specified stack size limit for the request exceeds the stack size limit (if configured) for the indicated batch queue.

If an attempt is made to queue a batch request that asks for a larger stack size limit than the destination batch queue has defined as allowable, the request is rejected.

The maximum stack size limit assigned to a particular request is determined at the time that the request is queued. If the stack size limit for the batch queue containing the request is later reduced, no requests in the batch queue are affected. However, if the new stack size limit value is less than the stack size limit requested by a previously queued request, then a warning message is displayed.

The stack size of any process created by the request cannot exceed the maximum in force when the request was queued.

Full CXbatch manager privileges are required to use this command.

Parameters:

limit

This is the maximum size in bytes for a stack produced by any process in any request. It is a single 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)

If you omit *units*, byte is assumed. You may specify that no limit should be applied by using *unlimited* for the value of *limit*.

queuename

This is the name of the queue to which the limit is assigned.

SET UNRESTRICTED_ACCESS

give all users access to queue

Format: **SEt Unrestricted_access** *queuename*

Description: The SET UNRESTRICTED_ACCESS command indicates that all users are to have access to the named queue.

Full CXbatch manager privileges are required to use this command.

Parameters: *queuename*

This is the name of the queue made accessible to all users.

SET WORKING_SET_LIMIT

set per-process maximum working-set size limit for queue

Format:

SEt Working_set_limit = (*limit*) *queuename*

Description:

The SET WORKING_SET_LIMIT command sets the per-process maximum working-set size limit for any batch request subsequently placed in the named batch queue. This limit defines the largest size of a working set that can be created by a process. CXbatch uses this limit when you do not include the `-lw` option with the `qsub` command.

When a request is submitted to a queue, CXbatch checks the request limits to ensure that no user-specified working-set size limit for the request exceeds the working-set size limit (if configured) for the indicated batch queue.

If an attempt is made to queue a batch request that asks for a larger working-set size limit than the destination batch queue has defined as allowable, then the request is rejected.

The maximum working-set size limit assigned to a particular request is determined at the time the request is queued. If the working-set size limit for the batch queue containing the request is later reduced, no requests in the batch queue are affected. However, if the new working-set size limit value is less than the working-set size limit requested by a previously queued request, then a warning message is displayed.

When physical memory becomes scarce, the system prefers to take physical memory away from those processes exceeding their working-set limit.

Full CXbatch manager privileges are required to use this command.

Parameters:

limit

This is the maximum size in bytes for a working set produced by any process in any request. It is a single integer representing less than 100,000,000 bytes with an optional fractional part.

The syntax for *limit* is:

fraction units 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)

If you omit *units*, byte is assumed. You may specify that no limit should be applied by using *unlimited* for the value of *limit*.

queuename

This is the name of the queue to which the limit is assigned.

SHOW

display information about CXbatch system

Format: **SHOW** *option*

Description: The SHOW command options provide you with information about the status of CXbatch, resource limits, queues, managers, and parameters.

Parameters: *option*

Specifies the desired operation. Table 2-1 lists the CXbatch SHOW options. The following pages describe each option in detail.

Table 2-1
SHOW Command Options

Option	Description
ALL	Displays the standard information concerning limits supported, managers, parameters, and queues.
LIMITS_SUPPORTED	Displays the resource limits supported on the local machine.
LONG QUEUE	Displays the status of CXbatch queues on the local host in an expanded format.
MANAGERS	Displays the list of authorized CXbatch managers.
PARAMETERS	Displays the general CXbatch settings.
QUEUE	Displays the status of CXbatch queues on the local host in a short format.

SHOW ALL

display standard information about CXbatch system

Format: **SHOw All**

Description: The SHOW ALL command displays standard information about the status of CXbatch resource limits, queues, managers, and settings.

Parameters: None

Examples: Mgr: **SHOw All**

This command displays the information about the CXbatch network. Following is an example of the output from this command:

```
Queues:

Queue for short jobs.
short@mach1; type=BATCH;  [ENABLED, INACTIVE];  pri=48
aliases: s, short_queue, S, SHORT
 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
 0 exit; 0 run; 0 stage; 0 queued; 0 wait; 0 hold; 0 arrive;

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;

Managers:

 root:m
 batchop:m
 leader:m

CXbatch 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
 Global per-user runlimit = NONE
 Checkpoint directory = /dir1/cxbatch/qrun/user/user.chkpnt
 Accounting log file = /dev/null
 Activity ID mask = None
 Mail account = root
 Next available sequence number = 201
 Batch request shell choice strategy = FREE

Limits supported:

 Core file size limit (-lc)
```

```
Data segment size limit (-ld)
Per-process permanent file size limit (-lf)
Nice value (-ln)
Stack segment size limit (-ls)
Per-process cpu time limit (-lt)
Working set limit (-lw)
```

SHOW LIMITS_SUPPORTED

display process and request limits enforced on local machine

Format: **SHOw LImits_supported**

Description: The SHOW LIMITS_SUPPORTED command displays the process and request limits enforced on the local machine. If a limit is not supported on the local machine and you include the limit with the `qsub` command, it is ignored. If the limit is supported on a remote machine, CXbatch honors it.

Parameters: None

Examples: Mgr: **SHOw LImits_supported**

This command displays the limits enforced on your machine. Following is an example of the output from this command:

```
Core file size limit (-lc)
Data segment size limit (-ld)
Per-process permanent file size limit (-lf)
Nice value (-ln)
Stack segment size limit (-ls)
Per-process cpu time limit (-lt)
Working set limit (-lw)
```

SHOW LONG QUEUE

display queue status information in long format

Format: **SHOw LOng Queue** [*queuename* [*username*]]

Description: The SHOW LONG QUEUE command displays status information about CXbatch queues in a long format.

Parameters:

queuename

This is the name of the queue for which the status is displayed. If no queue name is specified, CXbatch displays status information for all CXbatch queues. If a queue name is specified, CXbatch displays status information for the named queue only. The display orders the requests within the queue.

username

This is the user name whose request status is displayed. If you omit username, the status of each request in the queue is displayed.

Examples: Mgr: **SHOw Long Queue verylong**

This command displays the status for the *verylong* queue. Following is an example of the output from this command:

```
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
 Checkpoint : Not Available
 Share policy fixed = Verylong
 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
 Per-process stack size limit = UNLIMITED
 Per-process CPU time limit = UNLIMITED
 Per-process working set limit = UNLIMITED
```

SHOW MANAGERS

display the list of authorized CXbatch managers

Format: **SHOw Managers**

Description: The SHOW MANAGERS command displays the list of authorized CXbatch managers and operators.

Parameters: None

Examples: Mgr: **show managers**

This command displays the list of authorized CXbatch managers for your system. Following is an example of the output from this command:

```
root:m
batchop:o
leader:m
```

SHOW PARAMETERS

display current values for CXbatch operating parameters

Format: **SHOw Parameters**

Description: The SHOW PARAMETERS command displays the current values for the general CXbatch operating settings.

Parameters: None

Examples: Mgr: **SHOw Parameters**

This command displays the parameters used by all queues on your machine. Following is an example of the output from this command:

```
Debug level = 5
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
Checkpoint directory = /mach1/cxbatch/qrun/user.chkpnt
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

display queue status information in short format

Format: **SHOw Queue** [*queuename* [*username*]]

Description: The SHOW QUEUE command displays status information about CXbatch queues in a short format.

Parameters: *queuename*

This is the name of the queue for which status is displayed. If no queue name is specified, CXbatch displays status information for all CXbatch queues. If a queue name is specified, CXbatch displays status information for the named queue only. The display orders the requests within the queue.

username

This is the user name whose request status is displayed. If you omit username, the status of each request in the queue is displayed.

Examples: Mgr: **SHOw Queue verylong**

This command displays the status for the *verylong* queue. Following is an example of the output from this command:

```
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

shut down CXbatch on the local host

Format: **SHUtdown** [*seconds*] [*force*]

Description: The SHUTDOWN command shuts down CXbatch on the local host. CXbatch sends a SIGTERM signal to each process of each request that is running. All requests terminated as a result of the SHUTDOWN command are requeued for later execution. Unlike ABORT QUEUE, SHUTDOWN requeues all of the requests it kills. Requests that are successfully checkpointed are restarted from their checkpointed state when CXbatch is restarted.

After the time indicated in *seconds* has passed, CXbatch also sends a SIGKILL signal to all remaining processes for each request that is running in the named queue.

If you omit *seconds*, CXbatch assumes a default value of 60 seconds.

CXbatch operator privileges are required to use this command.

Parameters: *seconds*

This is the amount of real time that CXbatch waits before sending a SIGKILL signal to all remaining processes for each request. The SIGKILL signal is sent only to those processes that have not changed process groups.

START CXBATCH

start CXbatch

Format: **STArt CXbatch**

Description: The START CXBATCH command starts CXbatch from within `qmgr` on the local machine. You cannot start CXbatch from within `qmgr` the first time you install it.

CXbatch operator privileges are required to use this command.

Parameters: None

START QUEUE

start the queue

Format: **STArt Queue** *queuename*

Description: The START QUEUE command starts the queue and makes requests in the queue eligible for execution. If the queue is already started, you get a transaction completion message; no jobs are aborted.

CXbatch operator privileges are required to use this command.

Parameters: *queuename*

This is the name of the queue that is started.

STOP QUEUE

stop the queue

Format:

STOp Queue *queuename*

Description:

The STOP QUEUE command stops the queue. Any requests in the queue that are running are allowed to complete execution. All other requests eligible for execution in the queue are prevented from running. They remain in the queue and will be run when the queue is re-started.

A queue that has been stopped can still accept new requests. However, no requests that reside within the named queue are run until the queue has been explicitly started again by the START QUEUE command.

CXbatch operator privileges are required to use this command.

Parameters:

queuename

This is the name of the queue that is stopped.

SUSPEND REQUEST

suspend a specified request

Format: **SUSpend Request** *request -id*

Description: The SUSPEND REQUEST command freezes the execution of the named requests. The requests are checkpointed, then stopped from executing.

CXbatch operator privileges are required to use this command.

Parameters: *request-id*

This is the name of the request that is suspended.

qmapmgr Commands 3

Basic Information

This chapter contains a description of each `qmapmgr` command. Each command description includes

- Definition of command syntax.
- Description of command parameters.
- Examples.

The first two characters of each word in each command are unique, and you need to enter only those characters for CXbatch to recognize the command. These accepted abbreviations are indicated in the "Format" section of each command description as uppercase characters. Commands may be entered in any combination of uppercase and lowercase characters.

ADD MID

add a machine alias to the database

Format: **Add Mid** *mid principal-name*

Description: The ADD MID command adds a machine alias to the CXbatch database. A machine alias is an alternate name for a machine; any CXbatch command that requires a machine name also works with an alias.

Superuser privileges are required to use this command. .

Parameters: *mid*

This is the machine identification (MID) number of the machine that is assigned an alias. The MID must already exist in the CXbatch database.

principle-name

This is the name of the machine that is added to the network. The name must be unique and must correspond to an entry in the /etc/hosts file.

Examples: Mapmgr: **add mid 99 host1**

This command adds the machine named host1 to the CXCbatch database abd assign it an mid of 99. Following is an example of the output from this command:

```
NMAP_SUCCESS: Successful completion.
```

ADD NAME

add a machine alias to the database

Format: **Add Name** *alias mid*

Description: The ADD NAME command adds a machine alias to the CXbatch database. A machine alias is an alternate name for a machine; any CXbatch command that requires a machine name also works with an alias.

Superuser privileges are required to use this command.

Parameters: *alias*

This is the alias that is assigned to the machine. This alias is only known to the local CXbatch host and is not recognized across machines.

mid

This is the machine identification (MID) number of the machine that is assigned an alias. The MID must already exist in the CXbatch database.

Examples: Mapmgr: **add name test 99**

This command adds the alias *test* to the machine with the MID of 99 in the CXbatch database. Following is an example of the output from this command:

```
NMAP_SUCCESS: Successful completion.
```

CHANGE NAME

change a machine name associated with MID

Format: **CHange Name** *mid principal-name*

Description: The CHANGE NAME command changes the name of a machine associated with a machine identification (MID) number in the CXbatch network.

Superuser privileges are required to use this command.

Parameters: *mid*

This is the MID of the machine whose name is changed. The MID must already exist in the CXbatch database.

principal-name

This is the new machine name that is assigned to the MID. The new name must be unique and must correspond to an entry in the /etc/hosts file.

master

This command changes the name of the machine with MID 99 to *master* in the CXbatch database.

Examples: Following is an example of the output from this command:

Mapmgr: **change name 99**

```
NMAP_SUCCESS: Successful completion.
```

CREATE

create a CXbatch database

Format:	**CReate**
Description:	The CREATE command creates a new CXbatch database. Superuser privileges are required to use this command.
Parameters:	None

DELETE MID

delete a machine from the database

Format: **Delete Mid** *mid*

Description: The DELETE MID command deletes a machine from the CXbatch database.

Superuser privileges are required to use this command.

Parameters: *mid*

This is the machine identification (MID) number of the machine that is deleted. This MID must already exist in the CXbatch database.

Examples: Mapmgr: **delete mid 99**

This command deletes the machine with MID 99 from the CXbatch database. Following is an example of the output from this command:

```
NMAP_SUCCESS: Successful completion.
```

DELETE NAME

delete a machine alias from the database

Format: **Delete Name** *alias*

Description: The DELETE NAME command deletes a machine alias from the CXbatch database. Superuser privileges are required to use this command.

Parameters: *alias*

This is the alias that is deleted. The alias must already exist in the CXbatch database.

mid

This is the machine identification (MID) number of the machine that is assigned an alias. This MID must already exist in the CXbatch database.

Examples: The following command deletes the alias test in the CXbatch database:

```
NMAP_SUCCESS: Successful completion.
      NMAP_SUCCESS: Successful completion.
```

EXIT

exit from `qmapmgr` utility

Format: **Exit**

Description: The EXIT command exits from the CXbatch `qmapmgr` utility. You can also exit `qmapmgr` by entering the QUIT command or by pressing **CTRL-D**.

Parameters: None

GET MID

display the MID that matches machine name

Format: **Get Mid** *principal-name*

Description: The GET MID command displays the machine identification (MID) number of the indicated machine name.

Parameters: *principal-name*

This is the name of the machine whose MID is requested. This name can be either the actual machine name or an alias assigned to that machine.

The machine name or the alias must already exist in the CXbatch database.

Examples: Mapmgr: **get mid master**

This command displays the MID of the machine named master in the CXbatch database. Following is an example of the output from this command:

```
NMAP_SUCCESS:  Successful completion.
Mid = 99.
```

GET NAME

display the machine name that matches MID

Format: **Get Name** *mid*

Description: The GET NAME command displays the machine name of the indicated machine identification (MID) number.

Parameters: *mid*

This is the MID of the machine whose name is requested. This MID must already exist in the CXbatch database.

Examples: Mapmgr: **get name 99**

This command displays the machine name with MID 99 in the CXbatch database. Following is an example of the output from this command:

```
Name = master.
```

HELP

invoke the HELP facility

Format: **Help**

Description: The HELP command invokes the `qmapmgr` HELP facility and displays all available `qmapmgr` commands.

Parameters: None

Examples: `Mapmgr:` **`help`**

This command displays all available `qmapmgr` commands. Following is an example of the output from this command:

`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

exit from `qmapmgr` utility

Format: **Quit**

Description: The QUIT command exits from the CXbatch `qmapmgr` utility. You can also exit `qmapmgr` by entering the EXIT command or by pressing **CTRL-D**.

Parameters: None

SHOW

display all entries in the database

Format: **SHow**

Description: The SHOW command displays all entries in the CXbatch database. It lists the machine identification (MID) numbers with their corresponding machine names and aliases in MID order.

Parameters: None

Examples: `Mapmgr:` **`show`**

This command displays all MIDs in the CXbatch database and corresponding machine names and aliases. Following is an example of the output from this command:

```
Mid 1 = machine2, s
Mid 2 = machine1
Mid 3 = pluto, p
Mid 4 = test
Mid 5 = user1, u
```

Reporting Problems

A

This appendix introduces the CONVEX Technical Assistance Center (TAC) and the `contact` utility.

The `contact` utility is an online system for reporting problems to the TAC. To use it, enter **`contact`** at the system prompt and answer the questions as they appear on the screen.

This appendix describes:

- Prerequisites for using `contact`
- Tips for using `contact`
- The step-by-step process `contact` takes you through

Technical Assistance Center

The CONVEX Technical Assistance Center (TAC) is staffed by technical specialists who can address diverse questions and problems that arise in a supercomputing environment. If you have a hardware, software, or documentation question, contact the TAC. This group stands ready to solve such problems.

The `contact` Utility

The TAC recommends using the `contact` utility to report a hardware, software, or documentation problem. The `contact` utility is an interactive program that helps the TAC track reports and route them to the CONVEX personnel most qualified to fix a problem.

After you invoke `contact`, it prompts you for information about the problem. When you finish your report, `contact` electronically mails it to the TAC. The TAC notifies you within 48 hours that your report has been received.

To use `contact` requires:

- UNIX-to-UNIX Communication Protocol (UUCP) connection to the TAC
- Full path name of the program or utility in question
- Version number of the program or utility in question

UUCP Connection

Before using `contact`, ask your system administrator if your site has a UUCP connection to the TAC. A UUCP connection allows files to be copied from one UNIX--based system to another. The `uucp` (UNIX-to-UNIX copy) command relies on either a dial-up or hard-wired UUCP communication line.

Finding the Program Path Name

To determine the full path name of the program or utility in question, use the `which` command. Figure A-1 illustrates use of the `which` command to find the full path name of the loader (`ld`) utility.

Figure A-1
Using the which command

```
> which ld
/bin/ld
>
```

In this example, the full path name of the loader is /bin/ld.

If you use the C shell (`csh`), you can also use the `whence` command to find the program path name. The `whence` command works like `which`, but faster.

For more information on the `which` command, refer to the which(1) man page. You can also use the `info` online information system by entering **`info which`** at the system prompt.

Finding the Program Version Number

To determine the version number of the program or utility in question, use the `vers` command. Figure A-7 illustrates use of the `vers` command to find the version number of the loader (`ld`) utility. Enter **`vers`**, then the path name of the program or utility.

Figure A-2
Using the `vers` command

```
> vers /bin/ld
/bin/ld: 7.0
>
```

In this example, the loader version number is 7.0.

For more information on the `vers` command, refer to the vers(1) man page. You can also use the `info` online information system by entering **`info vers`** at the system prompt.

Using contact

The contact utility prompts for the following information:

- Your name, title, phone number, and corporate name
- Name and version of the product
- One-line summary of the problem
- Detailed description of the problem
- Priority of the problem
- Instructions on how to reproduce the problem
- Comments about the problem
- Comments about the documentation relating to the problem
- Files to include in the contact report

Following is a step-by-step discussion of these prompts.

Step 1a

To invoke the contact utility, enter **contact** at the system prompt. The system responds with a welcome message and a series of questions regarding your hardware, software, or documentation question. Figure A-3 illustrates use of the contact command and the resulting system response.

Figure A-3 Beginning a contact session

```
> contact
Welcome to contact version 0.11 ()

Enter your name, title, phone number, and corporate name (^D to terminate)
>
```

Step 1b

If there is a .contact file in your home directory, contact skips the first prompt. (Refer to "Using a .contact File" on page 6 for more information.) Figure A-4 illustrates the contact command and the system response when you have a .contact file in your home directory.

Figure A-4 Beginning a contact session with a .contact file

```
> contact
Welcome to contact version 0.11 ()

Enter the name of the product involved
>
```

Step 2

The `contact` utility prompts for the version number of the product. If you do not know the version number, press **CTRL-Z** to suspend the session.

Use the `which` (or `whence` if you use `csh`) and `vers` commands to find the version number of the product. Use the `fg` command to return to the session, and enter the version number in the form X.X or X.X.X.X.

Step 3

The `contact` utility prompts for a one-line summary of the problem. This summary is the subject header in any further correspondence regarding the problem. Please make this summary as descriptive as possible in one line.

Step 4

The `contact` utility prompts for a detailed description of the problem. Please make this description as complete as possible. Include source code and a stack backtrace when possible. (Refer to the adb(1) or csd(1) man page for information on obtaining a stack backtrace.) The more information you provide, the quicker the TAC can isolate and solve the problem.

Step 5

The `contact` utility prompts for the priority of the problem. Figure A-5 illustrates this prompt and priority levels from which to choose. You must enter a priority number.

Figure A-5 Specifying the priority of a problem

```
Enter a problem priority, based on the following:
1) Critical    - work cannot proceed until the problem is resolved.
2) Serious     - work can proceed around the problem, with difficulty.
3) Necessary   - problem has to be fixed.
4) Annoying    - problem is bothersome.
5) Enhancement - requested enhancement.
6) Informative - for informational purposes only.
>
```

Step 6

The `contact` utility prompts for an explanation of how to reproduce the problem. Please include the command syntax and options you used and anything else you did to make the program run.

Step 7

The `contact` utility prompts for any other pertinent comments. Please include all relevant information.

Step 8

The `contact` utility prompts for suggestions regarding documentation supporting the product. Indicate whether the documentation could be revised to address the problem.

Step 9

The `contact` utility prompts for names of files necessary to reproduce the problem. Figure A-6 illustrates this prompt and sample user response.

Figure A-6 Including files in a `contact` report

```
Are there any files that should be included in this report (yes | no)?
> yes
Please enter the names of the files, one to a line (^D to terminate)
> test.f
> ~/subroutines/sub.f
>
```

Note

"Tilde-Escape Sequences" on page 7 are not recognized in responses to this prompt. In `contact`, a tilde in this section indicates your home directory. This convention is based on use of the tilde for expanding file names in `csh`.

If files specified are small text files, they are automatically included in the `contact` report. If the files are too large to be included in this report, `contact` gives further instructions on how to submit these files.

To specify a directory, combine directory files into a single file using the `tar` command (refer to the tar(1) man page for further information) or enter each file name in the directory on a single line in the `contact` report.

Step 10

The `contact` utility prompts you to review, edit, submit, or abort the report. Figure A-7 illustrates this prompt.

Figure A-7
Prompt to review, edit, submit, or abort report

```
Please select one of the following options:
1) Review the problem report.
2) Edit the problem report.
3) Submit the problem report.
4) Abort the problem report.
>
```

Choose the number of the option you want to select. These options let you do the following:

Review	review the text of the `contact` report. You are then prompted again to select an option.
Edit	edit the text of the `contact` report. If you choose to edit the report, `contact` opens your default text editor.
Submit	sends the report to the CONVEX TAC. The TAC notifies you within 48 hours that your report has been received. Choosing this option exits the `contact` utility and returns you to the shell.
Abort	saves the text of the report in a file named ~/dead.report. Choosing this option exits `contact` and returns you to the shell.

Tips for Using `contact`

The `contact` utility is interactive and easy to use. This section lists tips to help use it efficiently. In particular, this section tells how to

- Use a .contact file
- Abort a `contact` session
- Resubmit an aborted report
- Suspend a `contact` session
- Move within `contact` from one prompt to another
- Use tilde-escape sequences in the `contact` utility

Using a .contact File

When you invoke `contact`, it first prompts for your name, title, phone number, and company name. You can, however, create a .contact file to skip this first prompt.

Follow these steps to create a .contact file.

1. Create a .contact file in your home directory.
2. Enter your name, job title, phone number, and company name, each on a new line.

When you invoke `contact`, it automatically includes the .contact file as input for the first prompt and proceeds to the next prompt.

Aborting the Report

To abort a `contact` report, either press the interrupt key (usually **CTRL-C**) or choose the `abort` option when prompted by the `contact` utility. Using **CTRL-C** to abort does not save the contents of the report. Using the `abort` option saves the contents of the report in a file named ~/dead.report.

Submitting the dead.report File

After you abort a `contact` session, the `contact` utility saves the report in a file named ~/dead.report. Using the `contact` command with the `-r` option automatically merges the contents of the ~/dead.report file into the new `contact` session. Enter

```
contact -r
```

and `contact` finds the ~/dead.report file and merges it into the `contact` report. You can then edit the report. When you end the editing session, `contact` resumes at the final prompt, which asks you to review, edit, submit, or abort the report.

Suspending a Report

Sometimes it is necessary to stop in the middle of a `contact` report and return to the shell (for instance, to suspend the `contact` session to find the program path name or version number). To suspend the `contact` session, press **CTRL-Z**.

To return to the `contact` session, press **`fg`**. Using **CTRL-Z** and the `fg` (foreground) command, you can toggle back and forth between the `contact` utility and the shell. You cannot, however, use **CTRL-Z** and `fg` to toggle back and forth in the Bourne shell (`sh`).

Ending a Response

The `contact` utility prompts for information pertinent to your hardware, software, or documentation question. Some prompts require one-line responses; to move to the next prompt, press **RETURN**. Other prompts require more than a one--line response; to move to the next prompt, press **CTRL-D**.

Tilde-Escape Sequences

The `contact` utility treats input beginning with a tilde (~) as a special sequence. The character following the tilde is considered a request for a special function. You can use the following tilde sequences within `contact`:

~e	start the text editor (defined in the EDITOR environment variable)
~h	display a list of available tilde-escape sequences
~p	print the `contact` report to the terminal screen
~r *filename*	read the contents of *filename* as a response to the current prompt. Some prompts require only a one-line response. This tilde-escape sequence works only for prompts that allow more than a one-line response.
~~	insert a single tilde as the first character in the line

Index

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V