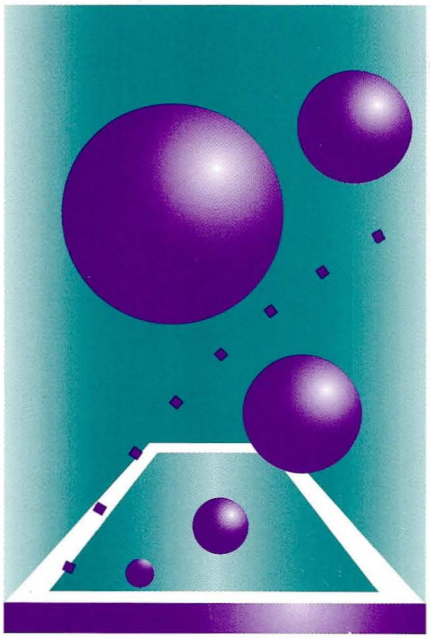


CONVEX

ConvexTMR User's Guide

First Edition

CONVEX Computer Corporation
3000 Waterview Parkway
P.O. Box 833851
Richardson, TX 75083-3851
United States of America
(214)497-4000

ConvexTMR User's Guide

Order No. DSW-481

First Edition

October 1994

CONVEX Press
Richardson, Texas
United States of America

ConvexTMR User's Guide

Order No. DSW-481

Printed in the United States of America

Revision information for

ConvexTMR User's Guide

Edition	Document No.	Description
First	710-029830-000	Initial release October, 1994.

Contents

Figures

Tables

Preface

Purpose and audience

The *ConvexTMR User's Guide* is for ConvexTMR (Convex Tape Mount Request) software users on ConvexOS and SPP-UX systems. This guide describes ConvexTMR and the ConvexTMR catalog. The catalog is an optional product; see your Convex sales representative for more information.

Using this guide

This book is organized into the following chapters:

- **Chapter 1 Overview**—Introduces ConvexTMR and the fundamentals of tape management. This chapter defines tape library organization and operations, data access methods, tape life cycles, and tape objects. This chapter also outlines steps for getting started and contains a command summary.
- **Chapter 2 Conducting tape sessions**—Explains how to initiate, manage and close a tape session. Contains a step-by-step walk through of some basic commands and sample tape sessions, and describes command options.

Note

Chapters 3 and 4 apply to the ConvexTMR CATALOG only. If your site is operating in NO-CATALOG MODE, you do not need to read these chapters.

- **Chapter 3 Using the catalog**—Describes the ConvexTMR catalog and the types of information it tracks, including object permissions, expiration dates, and vault rotations.
- **Chapter 4 Using tape pools**—Explains how to create and and use tape pools (tape pools are used only with the catalog).
- **Chapter 5 Using IBM and ANSI formats**—Describes the industry-standard methods of tape labeling and specifies what label information is recorded by the ConvexTMR catalog.
- **Chapter 6 Using special privileges**—Describes ConvexTMR special privileges, including requesting

priority levels, bypass label processing, making requests on behalf of other users, selecting specific devices, and exceeding site limits.

- Chapter 7 Using reports and logs—how to generate the various ConvexTMR reports. This chapter also explains how to interpret the ConvexTMR logs.
- Glossary—Defines terms used in this document.

Notational conventions

This section discusses notational conventions used in this book.

`Bold monospace`	In command examples, text shown in **`bold monospace`** identifies user input that must be typed exactly as shown.
`Monospace`	In paragraph text, `monospace` identifies: * Command names * System calls * Data structures and types In command examples, `monospace` identifies command output, including error messages. In command syntax diagrams, text shown in `monospace` must be typed exactly as shown.
Italic	In paragraph text, *italic* identifies: * New and important terms * Titles of documents In command syntax diagrams, *italic* identifies variables that must be supplied by the user.
{ }	In command syntax diagrams, text surrounded by curly brackets indicate a choice. The choices available are shown inside the curly brackets and separated by the pipe (|) sign. The following command example indicates that you can enter either `a` or `b`: **`command`** `{a | b}`
[ ]	In command syntax diagrams, square brackets indicate optional data.

The following command example indicates that definition of the variable *output_file* is optional:

`command` *input_file* [*output_file*]

...

In command syntax, horizontal ellipsis shows repetition of the preceding item(s).

The following command example indicates you can optionally specify more than one *input_file* on the command line.

`command` *input_file* [*input_file* ...]

KEYCAP

In paragraph text, text shown in KEYCAP indicates keyboard keys you must press to execute the command. For example, RETURN refers to the carriage return key.

Two KEYCAP terms separated by a hyphen indicate two keys that you must press simultaneously. For example, CTRL-d indicates that you must press the d key while holding down the CTRL key.

Notes, cautions, and warnings

This document presents notes, cautions, and warnings in the following formats.

Note

A Note highlights supplemental information.

Caution

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

Warning

A warning highlights information necessary to avoid injury to personnel.

Associated documents

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

For more information about ConvexTMR, you can order these books from CONVEX Computer Corporation:

- *ConvexTMR Operator's Guide* (DSW-482). This book describes the Convex Tape Mount Request operator interface.

- *ConvexTMR Administrator's Guide* (DSW-480). This book describes the Convex Tape Mount Request user interface.
- SPP-UX System Administration Guide (DSW-853). This book is the standard reference for the SPP-UX operating system.
- HP-UX Reference (multivolume set).

Ordering documents

To order the current edition of these or any other CONVEX document, send requests to:

CONVEX Computer Corporation
Customer Service
P.O. Box 833851
Richardson TX 75083-3851 USA

Please include the order number (DSW or DHW number) or the exact title of the document.

Technical assistance

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

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

You can also use the `contact` utility, if you would like to report any problems you may have with ConvexTMR or its associated documentation. For more information refer to the contact(1) man page in *ConvexOS Man Pages for Users*, or the appendix "Reporting problems" in the *ConvexOS Primer* or *Managing ConvexOS: Operations Guide*.

Overview 1

ConvexTMR is a tape management software system that integrates a full-featured tape manager with powerful cataloging abilities. The catalog is an optional product and must be purchased separately.

ConvexTMR without the catalog provides traditional tape management. It ensures only one user can access a tape at one time and processes tape labels. In addition, ConvexTMR provides

- role-based interfaces
- extensive media, device, and recording format options
- interactive and batch tape session support
- tape drive arbitration
- comprehensive logging and reporting facilities
- operational domain support

The catalog provides a higher level of abstraction by enabling a user to access tape files by name, as if they were on disk, and to set access privileges for tape files. The catalog also provides the following features and services:

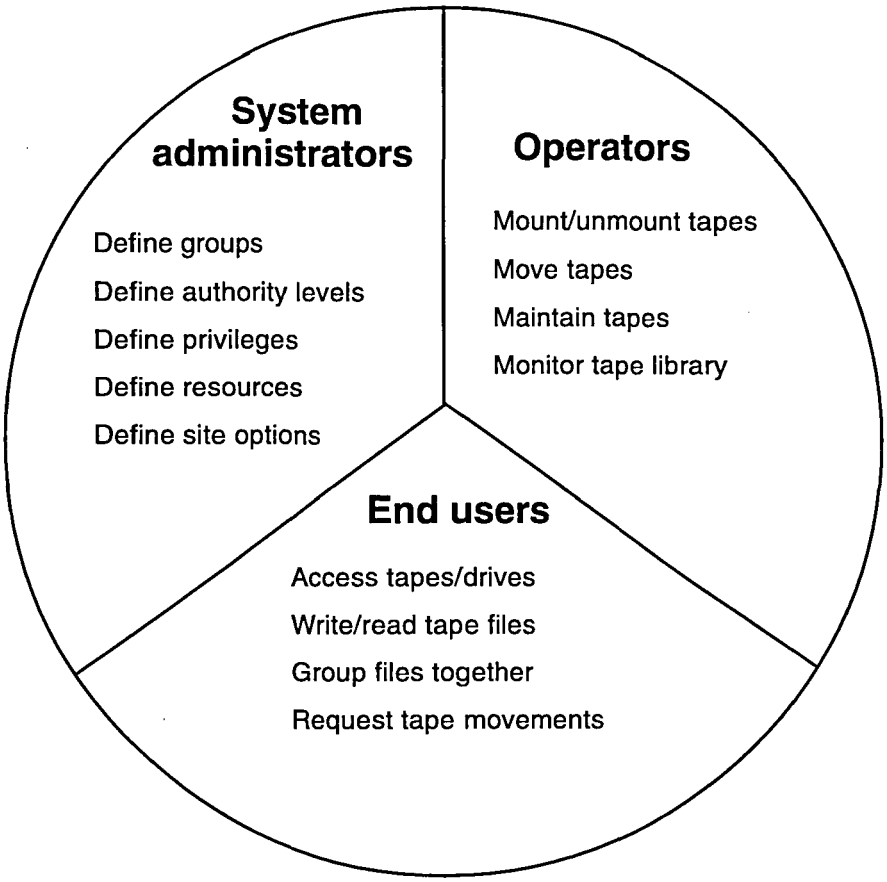
- data security control
- complete tape life-cycle management
- unlimited vaulting and rotation options
- user pool creation options

Role-based interfaces

ConvexTMR brings together three distinct role groups: system administrators, operators, and end users. ConvexTMR provides separate interfaces and programs for each of these user groups, each tailored to the specific needs of the particular group.

Figure 1 shows ConvexTMR roles groups and their tasks.

Figure 1 ConvexTMR role groups



System administrator activities

The ConvexTMR administrator configures the ConvexTMR software and supervises its operation. Administrator activities include defining

- administrator groups and authority levels
- operator groups and authority levels
- user and group privileges
- system resources
- site options

Operator activities

The ConvexTMR operator monitors the software operation and performs the day-to-day physical tape library tasks. All mounting, vaulting, and tape management requests are directed to the ConvexTMR operator via the Request Monitor.

Operator activities include

- mounting and unmounting tapes
- moving tapes to new locations
- performing tape maintenance:
  - cleaning tapes
  - erasing tapes
  - initializing tapes
  - identifying found tapes
  - monitoring and controlling the server programs

End user activities

The ConvexTMR user stores and retrieves tape file data during tape sessions. User activities include

- accessing tape drives and tapes for tape sessions
- writing and reading tape file data
- grouping tape files together in logical ways
- requesting tape movement among library sites

The catalog

The ConvexTMR online catalog is a comprehensive database that tracks all the objects in the library. The catalog maintains records on many attributes of ConvexTMR objects.

ConvexTMR catalog software keeps track of who owns the data and prevents unauthorized tape access. Ownership of data is confirmed via the catalog for each access request.

Some ConvexTMR sites choose not to order the catalog product. (The ConvexTMR catalog is an optional product; see your Convex sales representative for more information.) Parts of some commands and all of others are inactive in no-catalog mode. These differences are noted throughout the documentation.

Tape objects

A tape *object* is anything that you can reference or manipulate with the ConvexTMR catalog software. ConvexTMR manages six types of tape objects:

- *tape file*—like a disk file, a set of related bits that can span multiple tapes.
- *tape directory*—like a disk directory, a set of related files and subdirectories.
- *volume*—single tape that can contain zero or more files.
- *volumeset*—ordered set of one or more physical volumes that is treated as one logical volume.
- *pool*—group of volumes. A user can have more than one pool, but each volume is in one pool.
- *rotation*—list of vault locations and durations assigned to a volumeset. Many tape libraries vault, or store, copies of important volumesets at satellite locations.

Tape naming and identification

ConvexTMR stores identification information for all cataloged tape volumes. This allows users to reference tape volumes in a variety of ways. Users may reference volumes by

- relative path name (volumesets only)—ConvexTMR path to a file or directory from the current working directory.
- volume external label—user-defined, unique string of from 1 to 6 characters that identifies the volume.
- system-generated database key—system-generated string of characters that uniquely identifies a ConvexTMR file, volume, volumeset, pool, or rotation.
- volume submission receipt number—A five-digit, random number proceeded by "R." Receipts are generated by the ConvexTMR software when volumes are submitted to or retrieved from the tape library.

Tape sessions (with and without catalog)

ConvexTMR supports both interactive tape sessions and batch operations. It collects incoming requests in a prioritized queue, allocates the necessary resources, and directs the operator to perform the needed activities.

ConvexTMR offers two data access methods. In addition to traditional *volume-level access*, ConvexTMR also provides *file-level access*.

Volume-level access

During volume-level access, the entire tape volume is accessible to you. Each file access request may include instructions to the tape device to wind or rewind the tape to a specific position.

File-level access

An alternative to volume-level access, file-level access allows you to select a file by name, as if the tape file were a file on a disk. This frees you from the tape navigation concerns inherent in volume-level access. With file-level access, you may make different files on the same volume accessible to discrete user groups.

Privileged access

All ConvexTMR users have a basic set of access privileges. There is also a restricted set of privileges, controlled by the system administrator, that may be extended to users. These privileges are defined in "Using special privileges," on page 147.

Command summary

Table 1 and Table 2 summarize the commands available to the ConvexTMR user.

Table 1 ConvexTMR user commands (with or without catalog)

Command	Function
`rlaccess`	access off-line tapes
`rlnext`	access next off-line object
`rlemsg`	display specified error message
`rlopmsg`	send a message to the ConvexTMR operator
`rlr`	generate ConvexTMR reports
`rlrelease`	release media and/or device resources
`rlstat`	display tape management status
`rlxeov`	cross tape volume boundary

Table 2 ConvexTMR user commands (catalog only)

Command	Function
`rlcd`	change ConvexTMR storage directory
`rlchacl`	change access control list (ACL)
`rllsacl`	display access control list (ACL)
`rlchgrp`	change the group ownership of a ConvexTMR object
`rlchown`	change the owner of a ConvexTMR object
`rlflc`, `rlfle`, `rlfld`	create/edit/delete tape files
`rlls`	list objects
`rlmkdir`, `rlmoddir`, `rlrmdir`	ConvexTMR directory management commands
`rlmove`	volume or volumeset vault control
`rlmv`	move and/or rename ConvexTMR volumesets, files, and directories
`rlname`	enter a cataloged directory name for an unnamed file or volumeset
`rlpoolc`, `rlpoole`, `rlpoold`	create, edit, and delete tape pools
`rlpwd`	display pathname of current ConvexTMR storage directory
`rlretrieve`	volume or volumeset retrieval
`rlrotc`, `rlrote`, `rlrotd`	rotation management commands
`rlscratch`	scratch a volume or volumeset
`rlvole`	edit tape volume record
`rlvolsubmit`	user scratch tape volume submission

Table 2 ConvexTMR user commands (catalog only) **(continued)**

Command	Function
`rlvsc, rlvse`	create/edit a volumeset
`rlvssubmit`	volumeset submission

Conducting tape sessions 2

What is a tape session?

Data is stored and retrieved via ConvexTMR tape sessions. A tape session involves the following activities:

- acquiring temporary use of system tape devices
- requesting and receiving access to tape objects
- reading and writing files
- surrendering the tape devices and tape objects

ConvexTMR provides three basic commands for conducting tape sessions:

- `rlaccess` initiates the session
- `rlnext` manages the session
- `rlrelease` closes the session

"A short walk through ConvexTMR (catalog mode)" section on page 10, and "A short walk through ConvexTMR (no-catalog mode)" section on page 14 contain simple, step-by-step ConvexTMR sessions that you can follow to introduce yourself to some of the basic ConvexTMR functions and commands.

"The rlaccess command" section on page 16 briefly introduces all `rlaccess` options. The `rlnext` and `rlrelease` commands are described in "The rlnext command" section on page 36 and "The rlrelease command" section on page 39.

"Sample tape session (catalog mode)" section on page 41 and "Sample tape session (non-catalog mode)" section on page 60 present steps and examples for using the `rlaccess`, `rlnext`, and `rlrelease` commands to perform specific tasks.

A short walk through ConvexTMR (catalog mode)

To follow these steps, you need a scratch tape of a format that can be mounted on devices in your library.

You should first submit the tape that you wish to access to the ConvexTMR operator. You cannot access your tape until the operator has accepted the volume into the library. Depending on procedures at your site, this may take anywhere from several hours to an entire day.

The steps below use the ConvexTMR catalog. If your site is operating without a catalog, follow the NO-CATALOG steps in the following section.

Step 1 Add /usr/convex to your path. Enter

```
set path=($path /usr/convex)
```

Step 2 Verify that there are scratch volumes available to you. Enter

```
rlls -r pscrlist public
```

A list similar to that shown in Figure 2 is displayed:

Figure 2 Listing scratch volumes

```
Pool Scratch                          Fri Apr 29 14:51:09 1994


Vault: onsite


  vol_loc                 label     intlbl  mtype     rec_fmt
  -------                 -----     ------  -----     -------
  000033                  IBM       000033  round     6250
  000033                  IBM       000033  round     6250
  000050                  ANSI      000050  DAT       DAT
  000051                  NL        000051  DAT       DAT
  000111                  IBM       000111  3480      3480
  000112                  NL        000112  3480      3480
```

If this list is empty, contact your ConvexTMR administrator to find out how to access a scratch tape.

Step 3 Verify your current ConvexTMR directory. Enter

```
rlpwd
```

A message similar to the following is displayed:

```
/home/my_login
```

Note

ConvexTMR directories are not actual directories on a disk. They are maintained by the ConvexTMR catalog and are only visible inside ConvexTMR.

Step 4 Create a directory for this walk-through session; enter

```
rlmkdir walkthrough
```

Step 5 Move into the newly-created directory; enter

```
rlcd walkthrough
```

Step 6 Create a new tape volume to use for this walkthrough. Enter

```
rlaccess -RW -V :Ntape1:C my_link
```

- `-RW` indicates that you want read and write access to the volume. Read access is given by default, but if you request write access (or use options that create files) you must request read access explicitly.
- `-V` indicates that you are accessing a volumeset.
- `:N` indicates that the next argument is the volumeset name.
- `tape1` is the name that we are giving to the new volumeset.
- `:C` indicates that we are creating the volumeset.
- `my_link` is the name that we are giving to a symbolic link. that points to the actual tape device used.

This command will pend while the operator mounts the requested volume. This may take several minutes. When the command returns, a message similar to the following is displayed:

```
RL5198: Device: tc0: Allocate Scratch Tape
'000112/' (NL)

Device 1: /mnt/my_login/my_link
```

Step 7 When `rlaccess` completes, a symbolic link is created to a tape device with the proper volume mounted. To view a long listing of my_link, enter

```
ls -ls my_link
```

Step 8 `rlaccess` places the new volume into your ConvexTMR directory. To view the contents of your current ConvexTMR directory, enter

rlls

A message similar to the following is displayed:

```
tape1
```

To view a long listing, enter

rlls -l

A message similar to the following is displayed:

```
Vrwxoe>----->----->  my_login  my_grp          Sep 23 13:23 tape1
```

Step 9 To view an extended report of the new volumeset, enter

rlls -L

A report similar to the one shown Figure 3 is displayed. (Reports are defined in detail in Chapter 7, "Using Reports and Logs.")

Figure 3 Sample extended object report

```
VOLSET                                                          VOLSET

               name: tape1

              owner: my_login                  uperm: rwxoedmp---s
              group: my_group                  gperm: ------------
                                               operm: ------------
                                         acl entries: 0

               pool: public                     vault: onsite
           rotation: NONE                     catalog: set_
         media type: 3480                  expiration: :S
      recording fmt: 3480                disposition: scratch
          label fmt: NL                         erase: FALSE
      file tracking: TRUE                    scratch:  notscr
            comment:

Dates
        date expired: Dec 31 1999
            creation: Sep 21 17:49
         last access: Sep 23 13:23
   last modification: Sep 23 13:23

Keys
                 key: 2dc1657900000001Z
VOLSET                                                          VOLSET
```

Step 10 To view a list of all volumes in the new volumeset, enter

```
rlls -r vsvlist tape1
```

A report similar to that shown in Figure 4 is displayed:

Figure 4 Volumeset volume list

```
Volume Set Volume                          Fri Apr 29 14:57:08 1994

Volset: tape1

     vsvn     extlbl          vault        status
     ----     ------          -----        ------
     1        000112          onsite       active
```


Step 11 Write a file to the tape. Enter

```
cat /etc/passwd > my_link
```

Step 12 Release the device. Enter:

```
rlrelease my_link
```

Step 13 Access the tape again to read the file. Enter

```
rlaccess -V :Ntape1 my_link
```

This command will pend while the operator mounts the requested volume. This may take several minutes. When the command returns, a message similar to the following should appear on the screen:

```
Device 1: /mnt/my_login/my_link
```

Step 14 Read the file from the tape. Enter

```
cat my_link > passwd.copy
```

Step 15 Release the device. Enter

```
rlrelease my_link
```

Step 16 Verify that the new file is identical to the original. Enter

```
diff /etc/passwd passwd.copy
```

There should be no differences.

Step 17 Return your volume to the scratch state. Enter

```
rlscratch -f -V tape1
```

- `-f` forces the volume to be scratched even though it contains recent data
- `-V` indicates that the next argument is the volumeset name

Step 18 Verify that the volumeset was removed from your ConvexTMR directory. Enter

```
rlls
```

`tape1` should no longer be listed.

Step 19 Return to your original ConvexTMR directory. Enter

```
rlcd ..
```

Step 20 Remove your walkthrough directory. Enter

```
rlrmdir walkthrough
```

A short walk through ConvexTMR (no-catalog mode)

To follow these steps, you need a scratch tape of a format that can be mounted on devices in your library.

You should first submit the tape that you wish to access to the ConvexTMR operator. You cannot access your tape until the operator has accepted the volume into the library. Depending on procedures at your site, this may take anywhere from several hours to an entire day.

Step 1 Add /usr/convex to your path. Enter

```
set path=($path /usr/convex)
```

Step 2 Create a new tape volume to use for this walkthrough. Enter

```
rlaccess -RW -V000112 -l IBM my_link
```

- `-RW` indicates that you want read and write access to the volume
- `-V` indicates that you are accessing a volumeset
- `000112` is the label of the tape (your label may, of course, be different)
- `-l` indicates that the next argument is the type of the tape's label. Possible choices are IBM, ANSI, and NL.

- `my_link` is the name that we are giving to a symbolic link that will point to the actual tape device used.

This command will pend while the operator mounts the requested volume. This may take several minutes. When the command returns, a message similar to the following is displayed:

```
Device 1: /mnt/my_login/my_link
```

Step 3 When `rlaccess` completes, a symbolic link is created to a tape device with the proper volume mounted.

To view a long listing of `my_link`, enter

```
ls -ls my_link
```

Step 4 Write a file to the tape. Enter

```
cat /etc/passwd > my_link
```

Step 5 Release the device. Enter

```
rlrelease my_link
```

Step 6 Access the tape again to read the file. Enter

```
rlaccess -RW -V000112 -l IBM my_link
```

This command will pend while the operator mounts the requested volume. This may take several minutes. When the command returns, a message similar to the following is displayed:

```
Device 1: /mnt/my_login/my_link
```

Step 7 Read the file from the tape. Enter

```
cat my_link > passwd.copy
```

Step 8 Release the device. Enter

```
rlrelease my_link
```

Step 9 Verify that the new file is identical to the original. Enter

```
diff /etc/passwd passwd.copy
```

There should be no differences.

The rlaccess command

Use `rlaccess` to access data on tape. The `rlaccess` command enables you to access objects sequentially, or simultaneously, by acquiring several tape drives. The `rlnext` command, described in the next section, enables you to switch from one object to another during the session.

The following subsections briefly introduce all basic `rlaccess` options. The `rlaccess` restricted options, including making requests on behalf of others, requesting specific devices, and bypass label processing, are discussed in Chapter 5, "Using special privileges," on page 147. Several of the examples in the following subsections are included in "Extended Examples" in Appendix A. These extended examples place the command and options in the full context of mounting the device, accessing the volumesets, volumes and files, and checking the output from the session. This will make many of the brief examples below more easily understood.

Specifying devices

Use the following `rlaccess` options to select devices and device options.

Autoloader: -a

Use this option to request a drive with an automatic cartridge loader. Automatic cartridge loaders minimize operator intervention. This command is demonstrated below:

```
rlaccess -a -V:Ntape1 -V:Ntape2 tlink
```

Device model: -d *dmodel*

Use the `-d` option to specify a device model. Without this option, `rlaccess` requests any device model compatible with the specified objects.

```
rlaccess -d 3490 -V:Ntape1 tlink
```

The `-d` option indicates that what follows next is the desired device model. The above example requests that the volume `tape1` be mounted on a 3490 device. To study an extended example of this specific command see the "Device model: -d dmodel" section on page 183 of Appendix A.

The command will fail if you specify a device model that is not compatible with the specified objects.

To view a list of allowable device models, request a report with the following command:

```
rlr dmodel
```

Multiple devices: -N *ndev*

Use this option to acquire several tape drives, for simultaneous use with several objects, during a single session. The first object is mounted on the first tape drive; the second object is mounted on the second drive, and so on. If you specify more objects than tape drives, you may access the excess objects with the `rlnext` command.

The request will fail if you:

- Specify fewer objects than tape drives
- Specify more devices than your site limit allows
- Use this option with the `-g` or `-G` options (grouping)

Your site administrator can grant you authorization to exceed your site limit.

The following command requests three tape drives and three objects.

```
rlaccess -N 3 -V:Ntape1 -V:Ntape2 -V:Ntape3
```

To study an extended example of this specific command see the "Multiple devices: -N ndev" section on page 183 of Appendix A.

Media disposition: -q

Use this option to request media disposition selection handling. By default, ConvexTMR does not rewind the media when the access path name is closed. When you select `-q`, ConvexTMR creates an additional access path name for each device, *pathname*`.r`. If you access the device via *pathname*`.r`, ConvexTMR rewinds the media upon `close()`. The following command

```
rlaccess -q -V:Ntape1 tlink
```

requests media disposition selection handling; two additional access path names are established for the device. To study an extended example of this specific command see the "Media disposition: -q" section on page 184 of Appendix A.

Specifying tape objects

Access path file: -A *filename*

This option specifies the disk file into which the access path names are to be placed. By default, the path names are returned on standard output.

Upon successful completion, the file contains each of the allocated path names, one per line. If the request fails, then the file contains an appropriate error message.

```
rlaccess -A ./access_paths -V:Ntape1 tlink
```

To study an extended example of this specific command see the "Access Path file: -A filename" section on page 185 of Appendix A.

File object: -F *file_spec*

Use this option to specify a file; this option may be used alone (with a unique *file_spec*) or in conjunction with the `-V` option to locate a file on a specified volumeset.

The `rlaccess` command offers several methods of referencing files; these methods are listed below. Catalog sites can use any of these methods; no-catalog sites can use all but the last two. Files can be referenced by:

- physical file number (with or without the volume external label)
- file sequence number (with or without the file section number)
- section number (with or without the volume external label)
- generation number (with or without the volume external label)
- version number (with or without the volume external label)
- fileset identifier (with or without the volume external label)
- file identifier (with or without the volume external label)

In addition, catalog users can also reference files by

- cataloged filename (with or without the file generation and/or version)
- file database key

These file-referencing methods are described in the following sections.

Physical file number: -F [:e*ext_lbl*]:p*pno*

You may specify a file by its physical file number. The physical file number represents the order in which the file appears on the volumeset. If this value is unique, it may be given without the volume external label.

The following example demonstrates a file specified by the physical file number:

```
rlaccess -V000112 -F:p6 tlink
```

The :p modifier indicates that what follows is the physical file number. The above example specifies the file that occurs in the sixth position on the volumeset. To study an extended example of this specific command see the "Physical file number: -F[:eext_lbl]:ppno" section on page 186 of Appendix A.

File sequence number: -F:n*fseq*

You may specify a file by its file sequence number. The file sequence number represents the order in which the file occurs in the volumeset; this value is recorded on the HDR1 label of ANSI and IBM standard tapes. If this value is unique, it may be given without the file section number. In many cases the file sequence number will be the same as the physical file number.

The following example demonstrates a file specified by file sequence number:

```
rlaccess -V000112 -F:n0004 tlink
```

The :n modifier indicates that what follows is the file sequence number. The above example specifies the fourth file on the volumeset. To study an extended example of this specific command see the "File sequence number: -F:nfseq" section on page 187 of Appendix A.

File section number: -F:s*sec*

You may specify a file by its file section number The file section number represents the section of the tape that the file resides on; this value is recorded on the HDR1 label of ANSI standard tapes.

The following example demonstrates a file specified by file section number:

```
rlaccess -V000112 -F:s0007 tlink
```

The :s modifier indicates that what follows is the file section number; the above example specifies the file that resides on section seven of the volumeset.

File generation number: -F:g*gen*

You may specify a file by its file generation number. The file generation number represents the generation of the file; this value is recorded on the HDR1 label of ANSI and IBM standard tapes.

Note

Specifying a file by its generation number can be ambiguous when working with volumesets. The system will select the first file on the volumeset that matches the supplied parameter.

The following example demonstrates a file specified by file generation number:

```
rlaccess -V000112 -F:e000112:g12 tlink
```

The `:g` modifier indicates that what follows is the file generation number; the above example specifies the twelfth generation of the file. To study an extended example of this specific command see the "File generation number: -F:ggen" section on page 188 of Appendix A.

File version number: -F:v*ver*

You may specify a file by its file version number. The file version number represents the version of the file. This value is recorded on the HDR1 label of ANSI and IBM standard tapes.

The following example demonstrates a file specified by file version number:

```
rlaccess -V000112 -F:e000112:v3 tlink
```

The `:v` modifier indicates that what follows is the file version number; the above example specifies version three of the file. To study an extended example of this specific command see the "File version number: -F:vgen" section on page 189 of Appendix A.

Fileset identifier: -F:b*fsid*

You may specify a file by its fileset identifier. The fileset identifier is the volume serial number of the first volume in the volumeset; this value is recorded on the HDR1 label of IBM and ANSI standard tapes.

The following example demonstrates a file specified by the fileset identifier:

```
rlaccess -V000112 -F:bA1 tlink
```

The `:b` modifier indicates that what follows is the fileset identifier; the above example specifies a file that resides on a volumeset with the volume serial number `A1`.

File identifier: -F:f*fid*

You may specify a file by its file identifier. The file identifier is the same as the filename; this value is recorded on the HDR1 label of IBM and ANSI standard tapes.

The following example demonstrates a file specified by the file identifier:

```
rlaccess -V000112 -F:fmyfile tlink
```

The `:f` modifier indicates that what follows is the file identifier; the above example specifies the file with the file identifier

myfile. To study an extended example of this specific command see the "File identifier: -F:ffid" section on page 190 of Appendix A.

CATALOG-ONLY: To ensure that the file identifier as recorded in the HDR1 label matches the cataloged file name, use the following syntax when naming the file:

```
rlaccess -F:Nfilename:ffilename
```

Append new file: -F:A

You may specify a new file to be created and appended to the last file on the volumeset.

The following example demonstrates the creation of a file to be appended to the volumeset:

```
rlaccess -V000112 -F:ffoo:A tlink
```

The :A modifier indicates that a file with the specified filename be appended to the last file on the volumeset; the above example specifies that the file identifier of the file, as written to the HDR1 label, is foo. To study an extended example of this specific command see the "Append new file: -F:A" section on page 190 of Appendix A.

Overwrite file: -F:O*file_spec*

You may specify a new file to be created to overwrite a specified file on the volumeset.

The following example demonstrates a new file that overwrites another file:

```
rlaccess -F :fnewfile:O:ffoo tlink
```

The :O modifier indicates that what follows is a specification for the file to overwrite; in the above example, the file newfile will overwrite the file specified by :ffoo.

To avoid ambiguity, the :O modifier must come after all modifiers describing the new file.

If the :p modifier is used it is an error if end-of-tape is encountered before reaching the given file (even if the tape ends with an EOV). To study an extended example of this specific command see the "Overwrite file: -Fnew_file_spec:O:old_file_spec" section on page 191 of Appendix A.

Warning

When a file is overwritten, all data following that file on the volumeset is lost.

File name: -F :N*fname*[:G*gen*][:V*ver*]

This option for specifying files is a CATALOG-ONLY option; it is not available at sites operating without a catalog.

The `-F` option with no modifier informs the catalog that what follows is a filename. Prefix the filename with a `:N` modifier, as shown in the following example:

```
rlaccess -F:Nmyfile tlink
```

The `:N` modifier explicitly states that what follows is a filename. The catalog indexes to the latest generation and version of the requested file, unless you specify a file generation number and/or version number.

To ensure that the file identifier as recorded in the HDR1 label matches the cataloged file name, use the following syntax when naming the file:

```
rlaccess -F:Nfilename:ffilename tlink
```

File database key: -F:x*file_key*

This option for specifying files is a CATALOG-ONLY option; it is not available at sites operating without a catalog.

If your site uses a catalog, you may reference a file via the file database key, as shown in the following example:

```
rlaccess -F:x2bf50cfa00000001F tlink
```

The `:x` modifier informs the catalog that the value which follows is a unique, 18-character file database key.

Object groups: -g

Use this option instead of the `-N` option if you wish to group the objects for access by the same device. All objects grouped with the `-g` option are accessed via the same tape drive. A tape drive is assigned for each `-g` flag you specify.

The request will fail if:

- `-g` is not followed by an object
- you specify more devices than your site limit allows
- you use this option with the `-N` option

Your site administrator can grant you authorization to exceed your site limit.

`rlaccess` generates a path name for each object group. The order of the path names corresponds to the order of the object groups. This is demonstrated below.

Input:

```
rlaccess -g -V:Ntape1 -V:Ntape2 -g -V:Ntape3 \
-V:Ntape4 -g -V:Ntape5 -V:Ntape6 -V:Ntape7

Device 1: /home/xyz/tlink1

Device 2: /home/xyz/tlink2

Device 3: /home/xyz/tlink3
```

In the above example, volume `:Ntape1` is accessed via the device special file `/home/xyz/tlink1`; volume `:Ntape3` is accessed via the device special file `/home/xyz/tlink2`; volume `:Ntape5` is accessed via the device special file `/home/xyz/tlink3`. To access volume `:Ntape2`, issue a `rlnext` command for `/home/xyz/tlink1`. To study an extended example of this specific command see the "Object groups: -g" section on page 193 of Appendix A.

Named object groups: -G *path*

If you wish to specify the tape drive path names, use the `-G` option instead of the `-g` option. The following example is identical to the `-g` example, except that path names are specified. The paths can be absolute or relative.

```
rlaccess -G /usr/tape/tlink1 -V:Ntape1 \
-V:Ntape2 -G /usr/tape/tlink2 -V:Ntape3 \
-V:Ntape4 -G /usr/tape/tlink3 -V:Ntape5 \
-V:Ntape6 -V:Ntape7
```

The above command makes objects available on the specified path names. To study an extended example of this specific command see the "Named object groups: -G" section on page 194 of Appendix A.

Expiration override: -j

This option is available at sites that operate without a catalog, but there are some functional differences. These differences are clarified in the following section.

Use this option to allow file expiration overrides without user confirmation. An expiration exception occurs when a user attempts to overwrite a file which has not yet expired. Without this option, the write operation will fail. This option is useful for batch operations.

Catalog mode

The catalog contains the primary expiration date. If the file label and the catalog disagree, the catalog determines the expiration date.

No-catalog mode

The file labels contain the expiration date.

The following example demonstrates the use of this option in either mode:

```
rlaccess -RW -j -V 000001 -F:fveryolddata tlink
```

To study an extended example of this specific command see the "Expiration override: -j" section on page 195 of Appendix A.

Assume spanned volume: -J

Use this option to make the system assume, when EOT is encountered on a No-Label volumeset, that the file continues on the next volume. Otherwise, when EOT is encountered, the system assumes that the file is complete and that the next volume starts with a new file.

This option is employed in the following example.

```
rlaccess -V000112,734921,886394 -J tlink
```

Offset: -o *offset*

Use this option to specify the offset value for ANSI tapes; you may select a value from 0 to 99. The default value is 0. The following example selects an offset value of 32.

```
rlaccess -RW -V000112 -lANSI -F:fmydata \
-o 32 tlink
```

Object name: -O *objname*

Use this option to attach an arbitrary name to an object, to be used in subsequent `rlnext` and `rlrelease` invocations. Object names can be up to 16 characters in length and must be unique among names used under the same resource key. Do not use the `-O` option for named objects; it is intended to simplify the management of unnamed objects.

In the following example:

```
rlaccess -V:Ntape1 -F:n2 -O ftwo -F:fODO_DOB \
tlink
```

the file specified by `:n2` is given the arbitrary name `ftwo`.

```
rlnext -Oftwo tlink
```

To study an extended example of this specific command see the "Object Name: -O" section on page 196 of Appendix A.

Password: -P *password* (catalog only)

Use this option to assign a password to a newly-created object, or to give the password to access an existing object.

In the following example:

```
rlaccess -Pgoblue -V:Nmtdata:C tlink
```

the newly-created volumeset is assigned the password `goblue`.

Record format: -r *recfmt*

Use this option to select a record format for writing files. This information is encoded in the HDR2 label for IBM and ANSI formatted tapes.

recfmt takes the form: *fmt*`:B`*blen*`:R`*rlen*, where *blen* is the block length in bytes and *rlen* is the record length in bytes. *fmt* is one of the following:

`f`	fixed length records; equivalent to ANSI format F
`fs`	fixed length, standard records; equivalent to ANSI format F
`fb`	fixed length, blocked records; equivalent to ANSI format F
`fbs`	fixed length, blocked standard records; equivalent to ANSI format F
`v`	variable length records; equivalent to ANSI format D
`vb`	variable length, blocked records; equivalent to ANSI format D
`vs`	variable length, spanned records; equivalent to ANSI format S
`vbs`	variable length, blocked, spanned records; equivalent to ANSI format S
`u`	unformatted data

The following example specifies a fixed-length, blocked recording format, with 800 byte blocks and 80 byte records:

```
rlaccess -r fb:B800:R80 -V:Ntape1 tlink
```

Note

The ConvexTMR system maintains the definition of the data only. It does not convert the data to or from the specified format.

Password protection: -S *flag*

This option controls the password protection flag in the volume header. For IBM tapes, the following codes are allowed: `0`, `1`, `3`.

For IBM labeled tapes:

`0`	no password protection (default)
`1`	password protection for read or write
`3`	password protection for write

For ANSI labeled tapes any alpha-character is accepted, indicating password protection for read and write.

The following example demonstrates the use of the `-S` flag to assign password protection to the specified volumeset.

```
rlaccess -S 1 -V:Ntape1 tlink
```

Printer control code: -u *pcode*

Use this option if you would like to specify the printer control code to write to all subsequent volume header records on IBM labeled tapes. You may select one of the following codes:

`A`	ISO/ANSI/FIPS control character
`M`	machine control character
`N`	no control character

The following example writes the ISO/ANSI/FIPS control character to all subsequent volume header records.

```
rlaccess -u A -V:Ntape1 tlink
```

Unassigned objects: -U

Use the `-U` option to direct an object or list of objects to an unassigned access path. Unassigned objects can be assigned later via the `rlnext` command. Objects that exceed the number of requested devices are automatically unassigned; the `-U` option allows you to designate any objects you wish to as unassigned. In the example below:

```
rlaccess -N2 -V:Ntape1 -V:Ntape2 -U:Ntape3 tlink
```

the first two volumes are assigned to the two devices reserved by the `-N` option. The third volume is explicitly unassigned. To study an extended example of this specific command see the "Unassigned object: -U" section on page 197 of Appendix A.

Storage vault: -v *vault*

This option is available at sites that operate without a catalog, but there are some functional differences. These differences are clarified in the following section.

Catalog mode

Use the `-v` option to specify where a newly-created volumeset will reside. In the following example

```
rlaccess -v vault1 -V:C tlink
```

the volumeset will reside in `vault1`. If you would like to see a list of available vaults, request a report with the following command:

```
rlr vault
```

No-catalog mode

When operating without a catalog, the `-v` option tells the system operator where the tapes needed for the mount request reside.

This option can save you time if the volume you request is stored off-site and you know that mounting devices are available at that particular vault. In the following example:

```
rlaccess -v vault1 -V:000121,000449,97325 tlink
```

`-v vault1` specifies that you want your volume mounted on a device accessible to `vault1`.

Volumeset object: -V *vs_spec*

Use this option to specify a new or existing volumeset. No-catalog sites are restricted to external label referencing; catalog sites can reference volumesets by

- volume external label
- volumeset path name
- volume database key
- volumeset database key

External label: -V *extlbl*

One way to specify a volumeset is by volume external label, as shown in the following example:

```
rlaccess -V 000001,000795,003579 tlink
```

The syntax shown above is acceptable both for sites operating with a catalog and sites operating without a catalog.

Volumeset path name: -V:N*vs_name*

This option for specifying volumes is a CATALOG-ONLY option; it is not available at sites operating without a catalog.

If your site uses a catalog, you may reference a volumeset via the volume path name, as shown in the following example:

```
rlaccess -V:N/home/lfw/vol1 tlink
```

The `:N` modifier informs the catalog that the value which follows is a volume path name.

Volume database key: -V:k*volkey*

This option for specifying volumes is a CATALOG-ONLY option; it is not available at sites operating without a catalog.

If your site uses a catalog, you may reference a volumeset via the volume database key of one of the constituent volumes, as shown in the following example:

```
rlaccess -V:k2bf40e7200000001V tlink
```

The `:k` modifier informs the catalog that the value which follows is a unique, 18-character volume database key.

Database keys can be viewed via the extended object reports; refer to "Using reports and logs," on page 157.

Volumeset database key: -V:K*volkey*

This option for specifying volumes is a CATALOG-ONLY option; it is not available at sites operating without a catalog.

If your site uses a catalog, you may reference a volumeset via the volumeset database key of the volume, as shown in the following example:

```
rlaccess -V:K2c5d17a500000001Z tlink
```

The `:K` modifier informs the catalog that the value which follows is a unique, 18-character volumeset database key.

Database keys can be viewed via the extended object reports; refer to "Using reports and logs," on page 157.

Expiration: -x *expdate*

This option is available at sites that operate without a catalog, but there are some functional differences. These differences are clarified in the following section.

Use this option to specify an expiration date. Expiration dates can be associated with files or with volumesets. If a file and its volumeset have differing expiration dates, both dates must occur before the file expires.

Catalog mode

expdate can be specified in the following ways:

:I	infinite - the data never expires
:S	scratch - data expires immediately after the tape is unmounted
:R*N*	expires *N* days after creation
:A*N*	expires if not accessed in *N* days
:L	expires when all files on the volumeset have expired
:X*mm/dd/yy*	expires on the given date
:G*N*	expires when there are *N* younger generations

Only the S, R, and X date specifications can be encoded in the HDR1 label on labeled tapes. The other date specifications are maintained in the catalog and a value is written to the tape indicating that the file expires on the last day of the year 2999.

No-catalog mode

Only the R and X forms of this option are available.

The following example demonstrates the use of this option in either mode:

```
rlaccess -x:R30 -V:C tlink
```

In the above example, the newly-created volumeset will expire in thirty days. To study an extended example of this specific command see the "Expiration override: -j" section on page 195 of Appendix A.

End volume scope: -X (catalog only)

Use the -X option to end the scope of the previous volume object. This is useful when the next object specified is a file name, because if the file name is not unique, and the volume scope is not ended, it is unclear whether the file exists on the previous volume or another volume. The -X option avoids this problem by indicating that all objects that follow on the command line are unrelated to the previous volume object, as shown in the following example.

```
rlaccess -V:Nmytape -F:Nfile1 -X -F:Nmyfile \
tlink
```

Read labels file: -y *filename*

Use this option to create a disk file into which labels are placed when the specified file is read. This creates one file you can reference if you wish to view all labels or a particular group of labels for an object. You may separate different types of labels into different files or place copies of the label in multiple files (up to six). The following modifiers are available:

`:O`	overwrite file when a new label group is encountered (default: append)
`:P`*`prefix`*	save all labels that match *`prefix`*

The following example requests that all labels with the prefix `UHL1` be written to the file `UHL1_labels`.

```
rlaccess -y UHL1_labels:PUHL1 -V:Ntape1 \
-F:Nfile1 tlink
```

To study an extended example of this specific command see the "Read labels file: -y" section on page 199 of Appendix A.

Write labels file: -h *filename*

Use this option to create a disk file containing user labels. Each line in the file defines a single label. Refer to the `rlaccess`(1) man page for the rules governing the contents of this file.

When writing header labels, the system reads *`filename`* immediately after the user causes the first `write`() operation on the corresponding file. The user must put the proper values in `filename` prior to the first `write`().

When writing trailer labels, the system reads *`filename`* when the user completes the corresponding file. Completion is indicated by an explicit `close`(), or in the case of volume level access, by either a write-tapemark or rewind `ioctl`().

```
rlaccess -h label2 -V:Ntape1 tlink
```

Submitting requests

When `rlaccess` is used with the following options, it executes in the background and a request ID is assigned.

Background execution: -l

Use this option if you want the command to return immediately. By default, the `rlaccess` command will not return until the access has completed; this can take some time, due to competition for resources with other users. This option immediately returns a

request number in the place of the standard return, so that you can continue your session without waiting. This is demonstrated below.

```
rlaccess -I -F:Nfile1 tlink
```

```
1024
```

This request number identifies the request as reported by `rlstat`. If you would like to verify your request after it has completed, you may reference the request number and path name with a `rlstat` command; refer to the rlstat(1) man page for details. If you wish, you may place the path name(s) in a file via the Access Path File option (`-A`); this option is described on page 17. To study an extended example of this specific command see the "Background execution: -I" section on page 201 of Appendix A.

Resource key: -k *reskey*

Use the `-k` option to start several independent tape sessions. This option assigns each session a unique *reskey* (job name). If you do not specify a value for *reskey*, a default value is assigned. The *reskey* can be up to six characters long. the *reskey* value can be specified in the RL_RESKEY environment variable (see the rlaccess(1) man page for details).

```
rlaccess -k job1 -V:Ntape1 tlink
```

```
Device 1: /home/xyz/tlink
```

To study an extended example of this specific command see the "Resource key: -k reskey" section on page 203 of Appendix A.

New volume file: -z *filename*

This option specifies the disk file in which to place the volume identification of scratch volumes used during a tape session. This option is demonstrated in the example below.

Input

```
rlaccess -z vollist -V:C tlink
```

By default, the volume identification is returned on standard output. Volume identification is returned, one per line, in the following order:

label_type ext_lbl int_lbl

label_type can be any of the following characters:

`I`	IBM standard label
`A`	ANSI standard label
`N`	no label

O	other (non-standard label)

Close status: -Z

This option causes the system call `close()` to return the most recent error if any errors were encountered since the `open()`. Such errors are also returned during the `read()`, `write()` or `ioctl()` system calls. In the following example:

```
rlaccess -Z -V:Ntape1 tlink
```

the most recent error encountered by the tape device since `open()` is returned upon `close ()`,`read()`, `write()`, or `ioctl()`.

Upon receipt of an error notification, you may request error specification with the following command:

```
rlstat -e -P path
```

Using tapes

Use the following options to specify tape properties.

Non-transparent end-of-volume (EOV): -E

Use this option to request nontransparent end-of-volume handling. By default, ConvexTMR handles volume transitions automatically and transparently; when the end of the volume is encountered during a read or write, ConvexTMR gets the succeeding volume mounted without notification to you.

When you select `-E`, you are notified when volume transitions occur. Under this setting, the `read()` and `write()` functions receive a failure response (-1) when a volume transition must take place and an `errno` of `ENOSPC` is returned.

To clear the situation during a `read()`, issue a `rlxeov` command and attempt the `read()` again. The next read will return the first block from the next physical volume.

When encountering the end of the volume during a `write()`, you may issue either a `close()` or a `rlxeov` command. Issuing a `close()` writes EOF labels. Issuing `rlxeov` causes the write operation to continue on the next physical volume.

You may continue writing after `ENOSPC` is returned if you know that sufficient tape is left before the physical EOV. Be aware, however, that if you run out of tape before the write is complete, an unrecoverable error will occur.

Back space record(BSR) and forward space record(FSR) requests fail when volume boundaries are encountered; an error

notification of ENOSPC is returned. This will continue until rlxeov is issued.

Forward space file (FSF), back space file (BSF), and rewind (REW) requests are not effected.

The following command requests nontransparent end-of-tape handling:

```
rlaccess -E -V:Ntape1 tlink
```

When you receive end-of-volume notification during this session, issue the rlxeov command to cross the volume boundary, as demonstrated below:

```
rlxeov tlink
```

You are now able to resume normal processing. Refer to the rlxeov(1) man page for more details on rlxeov.

Recording format: -f *rformat*

Use the -f option to specify a recording format. Without this option, rlaccess requests any device compatible with the specified objects.

```
rlaccess -f 6250 -V:Ntape1 tlink
```

The -f option indicates that what follows next is the desired recording format; the above example requests that the volume tape1 be mounted on a tape drive capable of 6250 bpi recording.

The command will fail if you specify a recording format that is not compatible with the specified objects.

To view a list of allowable recording formats, request a report with the following command:

```
rlr rformat
```

No truncate flag: -K *state*

This option is available at sites that operate without a catalog, but there are some functional differences. These differences are clarified in the following sections.

Note

When a file is overwritten, all files following it on the volumeset become inaccessible. Any volumes following the newly overwritten file in the volumeset are scratched.

Use this option to determine the handling of truncated volumes. You have three options: scratch, retain, and hold. The scratch and retain options function the same with or without

a catalog. The `hold` option functions differently in catalog mode than in no-catalog mode.

Catalog mode

`scratch`	Scratch truncated volumes.
`retain`	Scratch truncated volumes. Hold volumes for later use by the volumeset.
`hold`	Do not scratch truncated volumes. **Hold** volumes for later use by the volumeset.

No-catalog mode

`scratch`	(same as catalog mode)
`retain`	(same as catalog mode)
`hold`	Do not scratch truncated volumes. **Do not hold** volumes for later use by the volumeset

The following example demonstrates the use of this option in either mode:

```
rlaccess -K retain -V:C tlink
```

In the above command, the `-K retain` option informs ConvexTMR that all volumes that are scratched and truncated from the volumeset be retained for future use in the event that the volumeset requires additional volumes. The retained volumes will be used before other scratch volumes are allocated.

Label type: -l *label_type*

The `-l` option defines the label type for a new volumeset. Valid selections are: `IBM`, `ANSI`, or `NL`.

```
rlaccess -f 3490 -l IBM -V:C tlink
```

Media type: -m *mtype*

Use the `-m` option to specify a media type. Without this option, `rlaccess` requests any device compatible with the specified objects.

```
rlaccess -f 6250 -m round -V:Ntape1 tlink
```

The above example requests that the volume `tape1` be mounted on a tape drive that can accommodate round media.

The selected media type must be compatible with the recording format specified with the `-f` option. The command will fail if you specify a media type that is not compatible with the specified objects. Media types are site-defined values.

To view a list of allowable media types, request a report with the following command:

```
rlr mtypes
```

Read access: -R

Use this option to establish read access. Read access is given by default, but if you request write access, you must request read access explicitly to receive both read and write access. In the following example:

```
rlaccess -R -V:Ntape1 tlink
```

access paths established for the object have read access permission.

Write access: -W

Use this option to establish write access. Read access is given by default, but if you request write access, you must request read access explicitly to receive both read and write access. In the following example:

```
rlaccess -W -V:Ntape1 tlink
```

access paths established for this object have write access permission.

Catalog-only options

The following options are available only at sites that operate with a catalog.

User comment: -C *comment*

Use this option to attach a comment to a file or volumeset. Your comment may contain up to 63 ASCII characters. Quotation marks are required if the comment contains spaces. See the example below.

```
rlaccess -C "initial model files" -V:C tlink
```

Erase flag: -e

Use the `-e` option to force any volume that leaves the volumeset to be erased. Volumes that leave the volumeset, either due to truncation or scratching, are erased. In the following example:

```
rlaccess -e -V:C tlink
```

all the volumes that leave the newly-created volumeset, for any reason, will be erased.

Pool: -p *pool*

Use the `-p` option when creating a volumeset to name the pool from which volumes are allocated. In order to use a pool owned by somebody else, the pool owner must grant you permission. If format, media type, or label type are specified, they must be available from the specified pool, or the request will fail. This option is shown in the example below.

```
rlaccess -f 3490 -m 3490 -l IBM -p public -V:C \
tlink
```

In the above example, the `-p` option informs the catalog that what follows is the name of the desired pool.

To view a list of available pools, issue the following command:

```
rlls /pool
```

To determine your default pool, issue the following command:

```
rlr constants
```

The value shown for `dpool=` is your default pool. If `dpool=IMPOOL`, your default pool is your private user pool. Private user pools take the form `u_`*uname*; *uname* represents the name of the user that owns them.

File tracking flag: -T

Use the `-T` option to request that any files on the newly-created volumeset be recorded in the catalog. In the following example:

```
rlaccess -T -V:C tlink
```

any file written to the newly-created volumeset will be recorded in the catalog.

The tape administrator may globally set or override this user option, rendering it inert to you.

The rlnext command

The `rlnext` command controls the tape session after the initial `rlaccess` request. With it, you can unmount an object and have another object mounted in its place.

The `rlnext` command has many of the same options as `rlaccess`; these options are listed in full in the `rlnext` (1) man page.

The `rlnext` command also has options that are not shared by the `rlaccess` command. These options, `-U` and `-N`, are discussed in the following sections.

Unassigned object: -U

Use this option to request mounting of the next object on the unassigned object list. The unassigned object list is specified by the `rlaccess -U` command. The order in which objects are listed by the `rlaccess -U` command determines the order in which the options are selected for mounting by the `rlnext -U` command. Objects must be specified by full path name when issuing a `rlnext -U` command.

The example below shows an initial access of five objects. The first object is assigned to the first path; the second object is assigned to the second path; the remaining three objects are unassigned.

Input:

```
rlaccess -G tlink1 -V:Ntape1 -G tlink2
-V:Ntape2 -U -V:Ntape3 -V:Ntape4 -V:Ntape5
```

```
Device 1: /home/xyz/tlink1

Device 2: /home/xyz/tlink2
```

The `rlnext` command, below, requests that the third object specified in the `rlaccess` command be mounted on the device represented by the first path. The second `rlnext` command requests that the fourth object specified in the `rlaccess` command be mounted on the device represented by the second path. The fifth object specified by the `rlaccess` command remains unassigned.

Input:

```
rlnext -U /home/xyz/tlink1

rlnext -U /home/xyz/tlink2
```

New object: -N

This option allows you to request that an object not specified with the initial `rlaccess` command be mounted on the designated path. The following input requests the mounting of three objects.

Input:

```
rlaccess -V:Ntape1 -V:Ntape2 -V:Ntape3 tlink

Device 1: /home/xyz/tlink
```

To request the mounting of another object with the `rlnext` command, use the `-N` option, as shown in the example below.

Input:

```
rlnext -N -V:Ntape4 /home/xzy/tlink
```

The rlstat command

The `rlstat` command provides status information for pending and active requests.

When issued without any options, it reports all active and pending `rlaccess` and `rlnext` requests issued by or on behalf of the user issuing the `rlstat` command.

When issued with the path option (-n), the command outputs the pathname associated with the *dev_no* of the device allocated by the given *reqnum*. If *reqnum* is not given, it defaults to the most recent request made by the user. Device numbers are derived from the order given in the `rlaccess` command.

The following options may be used with the `rlstat` command. Refer to the rlstat(1) man page for complete descriptions of these options.

-a	All Keys. Reports status summary for all requests the user has made under all resource key identifiers.
-A	All Users. Reports status summary for all requests made by all users.
-R	Request Detail. Reports the detail of the most recent resource request or the requests identified by the -k option.
-n	Report Path. Instructs rlstat to report the pathname for the designated request and device number. It defaults to the most recent request and the first device in that request. Must be used with -d *dev_no*.
-r *reqnum*	Request Number. Produces status information for pending resource requests and active resources with the given resource request number. The resource request number is assigned by the tape manager when a resource request is made.
-d *dev_no*	Device Number. Selects a device by giving its numeric position within the resource request that obtained it. If *reqnum* is not given, it defaults to the most recent request made by the user under the current resource key.

-O	Object List. The report lists each device associated with the request key and the object list attached to each device.
-P *path*	Path. Gives the pathname of the device.
-x	Extend Timeout. Indicates that the timeout is being extended for the designated request number. Must be used with -r *reqnum*.
-t *timeout*	Timeout. Gives the new timeout period in minutes.
-w	Wait. Waits for the pending rlaccess -I request. Must be used with -r *reqnum*.
-e	Last Error. Reports the last error for the specified path. Must be used with -P *path*.
-D	Device Name. Reports the primary name for the device currently assigned to the specified path. Must be used with -P *path*.
-k *reskey*	Resource Key. Restricts the command to the resource key *reskey*. The special value ANY selects all resource keys.

The rlrelease command

The `rlrelease` command relinquishes the resources and media allocated to a user by previous `rlaccess` or `rlnext` requests. You may also use `rlrelease` to terminate pending requests.

If you issue a `rlrelease` command **with no objects**:

- **All** volumes allocated to you under your current resource key are unmounted and returned to the tape library.
- **Any** access paths or devices you may have established are made available to other users.
- **All** active `rlaccess` or `rlnext` requests you may have made are terminated and removed from the queue.
- **Any** pending requests you may have made are terminated.
- **Any** resources reserved by pending requests are released.

If you issue a `rlrelease` command that specifies a file, volume, or other named object:

- All specified objects are unmounted and returned to the tape library.
- All specified objects are removed from the queue.

If you issue a `rlrelease` command that specifies an access path:

- The access path and related devices are released and made available to other users.

- Any active or pending `rlnext` request that assigns media resources to the access path specified by the user are terminated and removed from the queue.

The following `rlaccess` options may be used with the `rlrelease` command. Refer to the `rlrelease`(1) man page for complete descriptions of these options.

`-B` *uname*	release resource accessed on behalf of another user.
`-V` *vol_spec*	release specified volume.
`-F` *file_spec*	release volume containing specified file.
`-O` *object*	release object specified.
`-k` *reskey*	release all active or pending resources associated with specified resource key. `rlrelease` will look for the environment variable RL_RESKEY and use it if found.

`rlrelease` may also be used with the options described in the next two sections.

Release access path: -P *path*

Use this option to free the device and media associated with the given *path*. The *path* parameter is the access path returned by the `rlaccess` command. This option is demonstrated below.

```
rlrelease -P /home/xyz/tlink1
```

Release request number: -r *reqnum*

Use this option to release all devices and media associated with the given *reqnum*.The *reqnum* parameter is the release request number returned from `rlaccess` or `rlnext` commands issued with the `-I` (return immediate) option. This option is demonstrated below.

```
rlrelease -r 1024
```

Sample tape session (catalog mode)

Work through the following examples to familiarize yourself with basic tasks using ConvexTMR in catalog mode.

Creating a cataloged volumeset

In this example you use the `rlaccess` command to create a new cataloged volumeset, allocate a scratch volume, and mount and position the tape at the beginning of the first file on tape.

Step 1 Verify that there are scratch volumes available to you in your site's default pool. Enter

```
rlls -r pscrlist public
```

A list similar to that shown in Figure 5 is displayed:

Figure 5 Listing scratch volumes

```
Pool Scratch                          Fri Apr 29 14:51:09 1994


Vault: onsite


  vol_loc                 label     intlbl  mtype     rec_fmt
  -------                 -----     ------  -----     -------
  000001                  IBM       000001  round     6250
  000002                  IBM       000002  3480      3480
```

If this list is empty, contact your ConvexTMR administrator to find out how to access a scratch tape at your site.

Step 2 Create a new tape volume and give it a name. Enter

```
rlaccess -V:C:Nnew_tape my_link
```

- `-V` indicates that you are accessing a volumeset.
- `:C` indicates that you are creating a new volumeset.
- `:N` indicates that the next argument is the volumeset name and `new_tape` is the name you assign to the new volumeset.
- `my_link` is the name you are giving to the symbolic link that points to the actual tape device used.

This command pends while the operator mounts the requested volume. This may take several minutes. When the command returns, a message similar to the following is displayed:

```
Device 1: /home/xyz/my_link
```

`/home/xyz/my_link` is the path of the symbolic link name that you use to access the tape.

Step 3 Verify that the new volume was placed in your ConvexTMR directory. Enter

```
rlls
```

A message similar to the following is displayed:

```
new_tape
```

Step 4 Write a file to the tape. Enter

```
echo "HELLO WORLD" > my_link
```

Step 5 Release the device and tape. Enter

```
rlrelease my_link
```

Accessing a cataloged volumeset

This session accesses a volume that you have previously created. It assumes that you are accessing the volume named `new_tape` from the previous example.

Step 1 Verify that the volume named `new_tape` is in your current ConvexTMR directory. Enter

```
rlls
```

A message similar to the following is displayed:

```
new_tape
```

Step 2 Access the tape for reading and writing. Enter

```
rlaccess -V:Nnew_tape -RW tlink
```

- `-V` indicates that you are accessing a volumeset.
- `:N` indicates that the next argument is the volumeset name and `new_tape` is the name of the volumeset you are requesting.
- `-RW` indicates that you want read and write access to the tape.
- `tlink` is the name you are giving to a symbolic link that points to the actual tape device used.

This command pends while the operator mounts the requested volume. This may take several minutes. When the command returns, a message similar to the following is displayed:

```
Device 1: /home/xyz/tlink
```

/home/xyz/tlink is the path of the symbolic link name that you use to access the tape.

Step 3 In step 4 of the previous example, you wrote "HELLO WORLD" to a file on the tape. Now read that file back and verify its contents. Enter

```
cat tlink
```

The following is displayed:

```
HELLO WORLD
```

Step 4 Release the device and tape. Enter

```
rlrelease tlink
```


Creating cataloged, named files

This session creates a named file within an existing volumeset. By giving a name to your file, you will easily be able to find it again. You can simply access it by name and the ConvexTMR catalog will position the tape to the beginning of your file.

This example assumes that you are accessing the volume named "new_tape" that was used in the previous two examples.

Step 1 Verify that the volume named new_tape is in your current ConvexTMR directory. Enter

```
rlls
```

A message similar to the following is displayed:

```
new_tape
```

Step 2 Add a named file to the volumeset. Enter

```
rlaccess -V:Nnew_tape -F:Nnew_file:A -RW my_link
```

- -V indicates that you are accessing a volumeset.
- :N indicates that the next argument is the volumeset name and new_tape is the name of the volumeset you are requesting.
- -F indicates that you are accessing a file in the volumeset.
- :N indicates that the next argument is the file name and new_file is the name you are giving to the new file.
- :A indicates that you are appending to the end of the volumeset.
- -RW indicates that you are requesting read and write access.

- `my_link` is the name you are giving to the symbolic link that points to the actual tape device used.

This command pends while the operator mounts the requested volume. This may take several minutes. When the command returns, a message similar to the following is displayed:

```
Device 1: /home/xyz/my_link
```

`/home/xyz/my_link` is the path of the symbolic link name that you use to access the tape.

Step 3 Write a file to the tape. Enter

```
echo "I'm a named file" > my_link
```

Step 4 The first time that you write to a named file, it is created and placed in your ConvexTMR directory. Verify that the new file is in your ConvexTMR directory. Enter

```
rlls
```

A message similar to the following should appear on your screen:

```
new_tape
new_file
```

Step 5 Release the device and tape. Enter

```
rlrelease my_link
```

Accessing cataloged files

This session accesses a named file. It assumes that you are accessing the file `new_file` that was created in the previous example.

Step 1 Verify that the file named `new_file` is in your current ConvexTMR directory. Enter

```
rlls
```

A message similar to the following is displayed:

```
new_tape
new_file
```

Step 2 Access a named file inside of a volumeset. Enter

```
rlaccess -F:Nnew_file -RW my_link
```

- `-F` indicates that you are accessing a file.
- `:N` indicates that the next argument is the file name and `new_file` is the name of the requested file.

- `-RW` indicates that you are requesting read and write access.
- `my_link` is the name you are giving to the symbolic link that points to the actual tape device used.

This command pends while the operator mounts the requested volume. This may take several minutes. When the command returns, a message similar to the following is displayed:

```
Device 1: /home/xyz/my_link
```

`/home/xyz/my_link` is the path of the symbolic link name that you use to access the tape.

Step 3 In Step 3 of the previous example you wrote "I'm a named file" to the tape. Now read the file back and verify its contents. Enter

```
cat my_link
```

The following is displayed:

```
I'm a named file
```

Step 4 Since you accessed a named file, ConvexTMR granted you file-level access to the volume. This means that you can access the file that you requested, but ConvexTMR will not allow you to access anything else on the tape. To demonstrate this, try to read the next file on the tape. Enter

```
cat my_link
```

The following is displayed:

```
I'm a named file
```

Instead of going to another file, it printed `new_file` again.

Step 5 Release the device and tape. Enter:

```
rlrelease my_link
```

Writing several files to a volume

This session writes several files to an existing volume. It assumes that you are accessing the volume named `new_tape` that you created and used in the previous examples.

Step 1 Verify that the volume named `new_tape` is in your current ConvexTMR directory. Enter

```
rlls
```

A message similar to the following is displayed:

```
new_tape
new_file
```

Step 2 Access the volume for writing. Enter

```
rlaccess -V:Nnew_tape -RW my_link
```

- `-V` indicates that you are accessing a volumeset.
- `:N` indicates that the next argument is the volumeset name and `new_tape` is the name of the volumeset you are accessing.
- `-RW` indicates that you are requesting read and write access.
- `my_link` is the name you are giving to the symbolic link that points to the actual tape device used.

This command pends while the operator mounts the requested volume. This may take several minutes. When the command returns, a message similar to the following is displayed:

```
Device 1: /home/xyz/my_link
```

`/home/xyz/my_link` is the path of the symbolic link name that you use to access the tape.

Step 3 By default, you are given volume-level access to volumes that you request. This means that you can access all data that is on that volume. If you request write access to the volume, you can also write over any data that is already on the volume. You need to know what is already on your volume so that you don't overwrite any valuable data. To find out what files are on the volume named `new_tape`, enter

```
rlls -r vsflist -V new_tape
```

A list similar to that shown in Figure 6 is displayed:

Figure 6 Listing files in a volume

```
Volume Set File                          Thu Jun 16 10:43:20 1994


Volset: new_tape


    extlbl        vno   vsno  sect  expire      path

    ------        ----  ----  ----  ------      ----

    000001        1     1     1     :S          RL1553: Unnamed file
    000001        2     2     1     :S          /home/xyz/new_file
```

This shows that there are two files already on the volume.

Step 4 Move to the end of the data. There are several ways to do this, but for now just print the tape files to the screen. Enter

```
cat my_link
```

The following is displayed:

```
HELLO WORLD
```

(This is the unnamed file created in a previous example.)

Step 5 Print the second file. Enter

```
cat my_link
```

The following is displayed:

```
I'm a named file
```

(This is the file named `new_file` that was created in a previous example.)

Step 6 Now that the tape is positioned at the end of the data, write several files to it. Enter

```
echo "I'm the third file"  > my_link
echo "I'm the fourth file" > my_link
echo "I'm the fifth file"  > my_link
echo "I'm the sixth file"  > my_link
```

Step 7 Release the device and tape. Enter:

```
rlrelease my_link
```

Reading several files from a volume

The following is an example of a tape session that reads several files from a volume. (For this example to work, the volume being accessed must contain four files of data.)

Step 1 Enter

```
rlaccess -V:Nnew_tape -RW tlink
```

A message similar to the following is displayed:

```
Device 1: /home/xyz/tlink
```

Step 2 Enter

```
cat tlink > file1.copy1
cat tlink > file2.copy1
cat tlink > file3.copy1
cat tlink > file4.copy1
```

Step 3 Release the device and tape. Enter:

```
rlrelease
```

Step 4 Verify the contents of each of the newly created files by using the cat utility to print each one to your terminal.

Accessing an unnamed file

This session accesses an unnamed file on an existing volume. It assumes that you are accessing the volume named `new_tape` that you created and used in the previous examples.

Step 1 Verify that the volume named `new_tape` is in your current ConvexTMR directory. Enter

```
rlls
```

A message similar to the following is displayed:

```
new_tape
new_file
```

Step 2 Get a listing of all of the files on the volumeset. Enter

```
rlls -r vsflist -V new_tape
```

A list similar to that shown in Figure 7 is displayed:

Figure 7 Listing files in the volumeset

```
Volume Set File                          Thu Jun 16 11:28:11 1994

Volset: new_tape

     extlbl         vno   vsno  sect  expire      path

     ------         ----  ----  ----  ------      ----
     000001         1     1     1     :S          RL1553: Unnamed file
     000001         2     2     1     :S          /home/xyz/new_file
     000001         3     3     1     :S          RL1553: Unnamed file
     000001         4     4     1     :S          RL1553: Unnamed file
     000001         5     5     1     :S          RL1553: Unnamed file
     000001         6     6     1     :S          RL1553: Unnamed file
```

This shows that there are six files in the volumeset.

Step 3 Access the last file on the tape. This should be the file that contains the message "`I'm the sixth file`". Enter

```
rlaccess -V:Nnew_tape -F:p6 -RW my_link
```

- `-V` indicates that you are accessing a volumeset.
- `:N` indicates that the next argument is the volumeset name and `new_tape` is the name of the volumeset you are accessing.
- `-F` indicates that you are accessing a file inside the volumeset.
- `:p` indicates that the next argument is the physical file number of the requested file. The physical file number represents the order in which the files appear in the volumeset.
- `6` is the physical file number of the requested file.
- `-RW` indicates that you are requesting read and write access.
- `my_link` is the name you are giving to the symbolic link that points to the actual tape device used.

This command pends while the operator mounts the requested volume. This may take several minutes. When the command returns, a message similar to the following is displayed:

```
Device 1: /home/xyz/my_link
```

`/home/xyz/my_link` is the path of the symbolic link name that you use to access the tape.

Step 4 Read the file to verify that the correct file was accessed. Enter

```
cat my_link
```

The following message is displayed:

```
I'm the sixth file
```

Step 5 Release the drive and the tape. Enter

```
rlrelease my_link
```

Accessing multiple files

This session accesses multiple files on an existing volume. It assumes that you are accessing the volume named `new_tape` that you created and used in the previous examples.

Step 1 Verify that the volume named `new_tape` is in your current ConvexTMR directory. Enter

```
rlls
```

A message similar to the following is displayed:

```
new_tape
```

new_file

Step 2 Get a listing of all of the files on the volumeset. Enter

```
rlls -r vsflist -V new_tape
```

A list similar to that shown in Figure 8 is displayed:

Figure 8 Listing files in the volumeset

```
Volume Set File                          Thu Jun 16 11:28:11 1994

Volset: new_tape

     extlbl        vno   vsno  sect  expire      path

     ------        ----  ----  ----  ------      ----
     000001        1     1     1     :S          RL1553: Unnamed file
     000001        2     2     1     :S          /home/xyz/new_file
     000001        3     3     1     :S          RL1553: Unnamed file
     000001        4     4     1     :S          RL1553: Unnamed file
     000001        5     5     1     :S          RL1553: Unnamed file
     000001        6     6     1     :S          RL1553: Unnamed file
```

This shows that there are six files in the volumeset.

Step 3 Access the file named `new_file` and the fourth and fifth files on the tape. Enter

```
rlaccess -V:Nnew_tape -RW -F:Nnew_file -F:p4 \
-F:p5 my_link
```

- `-V` indicates that you are accessing a volumeset.
- `:N` indicates that the next argument is the volumeset name and `new_tape` is the name of the volumeset you are accessing.
- `-RW` sets both read and write access to the volumeset.
- `-F` indicates that you are accessing a file inside the volumeset.
- `:Nnewfile` indicated that you are accessing a catalog file name by specifying the file path.
- `:p` indicates that the next argument is the physical file number of the requested file.
- `-F` indicates that you are accessing a file inside the volumeset.
- `4` is the physical file number of the requested file.

- `-F` indicates that you are accessing a file inside the volumeset.
- `:p` indicates that the next argument is the physical file number of the requested file.
- `5` is the physical file number of the requested file.
- `my_link` is the name you are giving to the symbolic link that points to the actual tape device used.

This command pends while the operator mounts the requested volume. This may take several minutes. When the command returns, a message similar to the following is displayed:

```
Device 1: /home/xyz/my_link
```

`/home/xyz/my_link` is the path of the symbolic link name that you use to access the tape.

Step 4 Because `new_file` is the first file that you requested, verify that it is the current file. Enter

```
cat my_link
```

The following is displayed:

```
I'm a named file
```

Step 5 Each of the three files that you requested are accessed in file-access mode. When you are in file-access mode, you are restricted to one file. If you try to read another file, you will **not** be reading file number `4`. Since you are still restricted to `new_file`, you will read `new_file` again. Enter

```
cat my_link
```

The following is displayed:

```
I'm a named file
```

Step 6 The command `rlnext` is used to move to the next requested file. To move from `new_file` to file number `4`, enter

```
rlnext my_link
```

Step 7 Read the file and verify that it is correct. Enter

```
cat my_link
```

The following is displayed:

```
I'm the fourth file
```

Step 8 Move to the last requested file. Enter

```
rlnext my_link
```

Step 9 Read and verify the file. Enter

```
cat my_link
```

The following is displayed:

```
I'm the fifth file
```

If you issue another `rlnext` command, it will fail. To move back to a previously accessed file, you must issue an `rlnext` command naming the specific object you want to use.

```
rlnext -F:Nnew_file my_link
```

Step 10 Release the drive and the tape. Enter

```
rlrelease my_link
```

Using mt to navigate a tape

This session uses the `mt` command to position a tape. It assumes that you are using the volume named `new_tape` that you created and used in the previous examples.

Step 1 Verify that the volume named `new_tape` is in your current ConvexTMR directory. Enter:

```
rlls
```

A message similar to the following is displayed:

```
new_tape
new_file
```

Step 2 Get a listing of all of the files on the volumeset. Enter

```
rlls -r vsflist -V new_tape
```

A list similar to that shown in Figure 9 is displayed:

Figure 9 Listing files in the volumeset

```
Volume Set File                          Thu Jun 16 11:28:11 1994

Volset: new_tape

     extlbl         vno    vsno   sect   expire      path

     ------         ----   ----   ----   ------      ----
     000001         1      1      1      :S          RL1553: Unnamed file
     000001         2      2      1      :S          /home/xyz/new_file
     000001         3      3      1      :S          RL1553: Unnamed file
     000001         4      4      1      :S          RL1553: Unnamed file
     000001         5      5      1      :S          RL1553: Unnamed file
     000001         6      6      1      :S          RL1553: Unnamed file
```

This shows that there are six files in the volumeset.

Step 3 Acquire the volumeset `new_tape` for read/write access. Enter:

```
rlaccess -V:Nnew_tape -RW my_link
```

- `-V` indicates that you are accessing a volumeset.
- `:N` indicates that the next argument is the volumeset name and `new_tape` is the name of the volumeset you are accessing.
- `-RW` indicates that you are requesting read and write access.
- `my_link` is the name you are giving to the symbolic link that points to the actual tape device used.

This command will pend while the operator mounts the requested volume. This may take several minutes. When the command returns, a message similar to the following is displayed:

```
Device 1: /home/xyz/my_link
```

`/home/xyz/my_link` is the path of the symbolic link name that you use to access the tape.

Step 4 Read the first file. Enter

```
cat my_link
```

The following is displayed:

```
HELLO WORLD
```

Step 5 Forward space one filemark (skip over the second file). Enter

```
mt -f my_link fsf 1
```

Step 6 Read the third file. Enter

```
cat my_link
```

The following is displayed:

```
I'm the third file
```

Step 7 Back up to the beginning of the second file. The volume is presently located at the beginning of the fourth file. To locate to the beginning of the second file, it is necessary to back space three filemarks, then forward space one filemark. After back spacing three filemarks, the tape is located at the end of the first file. After forward spacing one filemark, the tape is located at the beginning of the second file.

Enter:

```
mt -f my_link bsf 3
mt -f my_link fsf 1
```

Note

To back up to the beginning of a file, always perform a forward space operation after a backward space operation.

Step 8 Read the second file (`new_file`). Enter

```
cat my_link
```

The following is displayed:

```
I'm a named file
```

Step 9 Rewind the tape. Enter

```
mt -f my_link rew
```

Step 10 Release the drive and the tape. Enter

```
rlrelease my_link
```

`mt` or any other program that uses tape I/O control (ioctl) calls to position the tape will work with an accessed volume. ioctl calls work transparently within the ConvexTMR context. For example, a rewind operation from the third volume in a three-volume volumeset positions to the beginning of the first volume. Similarly, if a volume is accessed in file access mode (i.e. you request access to only one file on a volume), a rewind operation positions the tape to the beginning of that file.

For more information on the `mt` command or ioctl calls, see the mtio(4) or the ioctl(2) man page.

Accessing multiple volumesets

This session accesses two separate volumesets. You can move from one volumeset to the another by using the `rlnext` command. This session assumes that you are accessing the volumeset named `new_tape` that you created and used in the previous examples. You will also create a second volumeset.

Step 1 Verify that there are scratch volumes available to you in your site's default pool. Enter

```
rlls -r pscrlist public
```

A list similar to that shown in Figure 10 is displayed:

Figure 10 Listing scratch volumes

```
Pool Scratch                          Fri Apr 29 14:51:09 1994


Vault: onsite


  vol_loc                 label     intlbl  mtype     rec_fmt
  -------                 -----     ------  -----     -------
  000001                  IBM       000001  round     6250
  000002                  IBM       000002  3480      3480
```

If this list is empty, contact your ConvexTMR administrator to find out how to access a scratch tape at your site.

Step 2 Verify that the volume named new_tape is in your current ConvexTMR directory. Enter

```
rlls
```

A message similar to the following is displayed:

```
new_tape
new_file
```

Step 3 Access the volumeset named new_tape for read/write access. Also create a second volumeset and name it. Enter

```
rlaccess -V:Nnew_tape -RW -V:Nsecond_tape:C \
-RW my_link
```

- `-V` indicates that you are accessing a volumeset.
- `:N` indicates that the next argument is the name of the volumeset and `new_tape` is the name of the requested volumeset.

- `-RW` indicates that you are requesting read and write access.
- `-V` indicates that you are accessing a volumeset.
- `:N` indicates that the next argument is the name of the volumeset and `second_tape` is the name to assign to the new volumeset.
- `:C` indicates that the volumeset needs to be created.
- `-RW` indicates that you are requesting read and write access.
- `my_link` is the name you are giving to a symbolic link that points to the actual tape device used.

The `-V` option must be given multiple times to access separate volumesets.

This command pends while the operator mounts the first requested volume. This may take several minutes. When the command returns, a message similar to the following is displayed:

```
Device 1: /my_login/my_link
```

`/home/xyz/my_link` is the path of the symbolic link name that you use to access the tape.

Step 4 Read the first file of `new_tape`. Enter

```
cat my_link
```

The following should appear on the screen:

```
HELLO WORLD
```

Step 5 Use the `rlnext` command to move to the newly created volumeset named `second_tape`. Enter

```
rlnext my_link
```

This command will pend while the operator mounts the requested volume. This may take several minutes.

Step 6 Verify that the new volumeset was created in your ConvexTMR directory. Enter

```
rlls
```

A message similar to the following is displayed:

```
new_tape
new_file
second_tape
```

Step 7 Write a file to the new volumeset. Enter:

```
echo "I'm named second_tape" > my_link
```

Step 8 Release the drive and tape. Enter

```
rlrelease my_link
```

Writing to two (or more) volumesets could be accomplished with several separate `rlaccess` commands. Issuing a single command, however, guarantees that all devices and all volumes are allocated in a single atomic resource request. This safeguards against resource deadlock.

Using multiple devices

This session uses two different tape devices simultaneously. It assumes that you are accessing the volumesets named `new_tape` and `second_tape` that you created and used in the previous examples.

Step 1 Verify that the volumesets named `new_tape` and `second_tape` are in your current ConvexTMR directory. Enter:

```
rlls
```

A message similar to the following is displayed:

```
new_tape
new_file
second_tape
```

Step 2 Request that your two volumesets (`new_tape` and `second_tape`) be placed on two separate drives. Enter

```
rlaccess -G my_link1 -V:Nnew_tape -RW -G \
my_link2 -V:Nsecond_tape -RW
```

- `-G` indicates that the following arguments are to be grouped together.
- `my_link1` is the name that you are giving to a symbolic link that points to the first tape device used.
- `-V` indicates that you are accessing a volumeset.
- `:N` indicates that the next argument is the name of the volumeset.
- `new_tape` is the name of the volumeset that you are requesting.
- `-RW` indicates that you are requesting read and write access.
- `-G` indicates the end of the previous grouping and that the following arguments are to be grouped together.

- `my_link2` is the name that you are giving to a symbolic link that will point to the second tape device used.
- `-V` indicates that a volumeset is being accessed.
- `:N` indicates that the next argument is the name of the volumeset `second_tape` is the name of the volumeset that you are requesting .
- `-RW` indicates that you are requesting read and write access.

This command pends while the operator mounts both of the requested volumes. This may take several minutes. When the command returns, a message similar to the following is displayed:

```
Device 1: /home/xyz/my_link1
Device 2: /home/xyz/my_link2
```

`/home/xyz/my_link1` and `/home/xyz/my_link2` are the paths of the symbolic link names that you use to access the tapes.

If any device in the request experiences an error, the entire request is cancelled. Any devices already positioned are un-mounted and freed.

Step 3 Read `second_tape`'s first file. Enter

```
cat my_link2
```

The following is displayed:

```
I'm named second_tape
```

Step 4 Copy the first file on `new_tape` to `second_tape`. Use the `dd` command to copy the file. `dd` copies from an input file (`if`) to an output file (`of`) using a specified block size (`bs`). Enter

```
dd if=my_link1 of=my_link2 bs=32k
```

A message similar to the following is displayed:

```
0+1 records in
0+1 records out
```

Step 5 Use the `mt` command to rewind `second_tape`. Enter

```
mt -f my_link2 rew
```

Step 6 Read `second_tape`'s first two files. Enter

```
cat my_link2
```

The following is displayed:

```
I'm named second_tape
```

Step 7 Enter

```
cat my_link2
```

The following is displayed:

```
HELLO WORLD
```

This is the file that you copied from `new_tape`.

Step 8 Release both drives and tapes. Enter

```
rlrelease
```

When you issue the `rlrelease` command with no arguments, it releases all of the drives that you hold.

Acquiring two (or more) devices could be accomplished with several separate `rlaccess` commands. Issuing a single command, however, guarantees all devices and all volumes are allocated in a single atomic resource request. This safeguards against resource deadlock.

For more information about the command `dd`, see the dd(1) man page. For more information about the command `mt`, see the mt(1) man page.

Returning volumesets to the scratch state

This session places volumesets into the scratch state. When you return a cataloged volumeset to the scratch state, your data is lost and the tapes become available for others to use. You should only return your volumesets to the scratch state when you are sure that you no longer need the data on the tape.

This example assumes that you are using the volumesets named `new_tape` and `second_tape` that you created and used in the previous examples.

Step 1 Verify that the volumesets named `new_tape` and `second_tape` are in your current ConvexTMR directory. Enter:

```
rlls
```

A message similar to the following is displayed:

```
new_tape
new_file
second_tape
```

Step 2 Return `new_tape` to the scratch state. Enter

```
rlscratch -f -V new_tape
```

- `-f` forces the volumeset to be scratched.
- `-V` indicates that the next argument is the volumeset name.

- `new_tape` is the name of the volumeset being scratched.

Step 3 Verify that `new_tape` and all of its files were removed from your ConvexTMR directory. Enter

```
rlls
```

A message similar to the following is displayed:

```
second_tape
```

Step 4 Return `second_tape` to the scratch state. Enter

```
rlscratch -f -V second_tape
```

- `-f` forces the volumeset to be scratched.
- `-V` indicates that the next argument is the volumeset name.
- `second_tape` is the name of the volumeset being scratched.

Step 5 Verify that `second_tape` was removed from your ConvexTMR directory. Enter

```
rlls
```

`second_tape` should not appear in the listing.

Sample tape session (non-catalog mode)

Work through the following examples to familiarize yourself with basic tasks using ConvexTMR with no catalog.

Creating a new volumeset

In this example you use the `rlaccess` command to create a new volumeset, allocate a scratch volume to it, and mount and position the tape at the beginning of the first file on tape.

Step 1 Find the media types supported by your site. Enter

```
rlr mtype
```

A list similar to that shown in Figure 11 is displayed:

Figure 11 Listing configured media types

```
mtype report:                    Wed Jul 13 10:40:45 1994

Configured Media Types:
     Media Type  Formats
     ----------  -------
     round       800       1600      3200      6250
     DAT         DAT       DATH
     3480        3480      3490
```


When you access a volume, you will need to specify both its media type and its recording format. For example, if you want to access a round tape (nine track tape), then you will specify the media type as round. You will then select the recording format from the available formats. For example, you may specify that the recording format should be 6250.

Step 2 Create a new tape volume. This examples uses DAT tapes, but you may substitute your preferred media type and recording format. Enter

```
rlaccess -V:C -m DAT -f DAT -l ANSI my_link
```

- `-V` indicates that you are accessing a volumeset.
- `:C` indicates that you are creating a new volumeset.
- **`-m`** indicates that the next argument specifies the media type of the requested volume.
- `DAT` is the media type of the volume you are requesting.
- `-f` indicates that the next argument specifies the recording format of the requested volume.
- `DAT` is the recording format of the volume you are requesting.
- `-l` indicates that the next argument specifies the label type.
- `ANSI` is the label type of the requested volume. Possible choices for label type are `ANSI`, `IBM`, and `NL`.
- `my_link` is the name you are giving to a symbolic link that points to the actual tape device used.

This command pends while the operator mounts the requested volume. This may take several minutes. When the command returns, a message similar to the message in Figure 12 is displayed:

Figure 12 rlaccess return output

```
RL5198: Device: dat0: Allocate Scratch Tape '000001/000001' (ANSI)
Device  1:  /home/xyz/my_link
```

The second number `000001` is the external label of the volume.

It is very important that you know the external label of a scratch volume that you have accessed. External labels can be written to a file (see "New volume file: -z filename," on page 31). Also, external label messages are written to the user's log and the job log located in /usr/lib/REEL/logs/u_*name* and /usr/lib/REEL/logs/ j_*name*/*reskey*.

`/home/xyz/my_link` is the path of the symbolic link name that you use to access the tape.

Step 3 ConvexTMR maintains a log file for each user. If you forget to record the external label of a scratch volume that you have accessed, you should be able to find it in this log.

To read your log file, enter

`cat /usr/lib/REEL/logs/u_mylogin`

Replace "`mylogin`" with your login name. A list similar to that shown in Figure 13 is displayed:

Figure 13 ConvexTMR log file

```
Jul 13 13:27:24> RL5239: Request 1034 obtained resources
Jul 13 13:29:09> RL5198: Device: dat0: Allocate Scratch Tape '000001/000001' (ANSI)
Jul 13 13:29:17> RL5241: Request 1034 Device /home/xyz/my_link ready
Jul 13 13:29:18> RL5242: Request 1034 return success
```

Step 4 Write a file to the tape. Enter

`echo "HELLO WORLD" > my_link`

Step 5 Release the device and tape. Enter

`rlrelease my_link`

Accessing a volumeset

This session accesses a volume that you have previously created. It assumes that you are accessing the volume created in the previous example. It also assumes that you know the external label of this volume.

Step 1 Access the tape for reading and writing. Enter

```
rlaccess -V 000001 -m DAT -f DAT -l ANSI -RW \
my_link
```

- `-V` indicates that you are accessing a volumeset.
- `000001` is the external label of the volumeset that you are requesting.
- `-m` indicates that the next argument specifies the media type of the requested volume.
- `DAT` is the media type of the volume you are requesting.
- `-f` indicates that the next argument specifies the recording format of the requested volume.
- `DAT` is the recording format of the volume you are requesting.
- `-l` indicates that the next argument specifies the label type.
- `ANSI` is the label type of the requested volume. Possible choices for label type are `ANSI`, `IBM`, and `NL`.
- `-RW` indicates that you want read and write access to the tape.
- `my_link` is the name you are giving to a symbolic link that points to the actual tape device used.

This command pends while the operator mounts the requested volume. This may take several minutes. When the command returns, a message similar to the following is displayed:

```
Device  1:  /home/xyz/my_link
```

`/home/xyz/my_link` is the path of the symbolic link name that you use to access the tape.

Step 2 In Step 4 of the previous example, you wrote "`HELLO WORLD`" to a file on the tape. Now read that file back and verify its contents. Enter

```
cat my_link
```

The following is displayed:

```
HELLO WORLD
```

Step 3 Release the device and tape. Enter

```
rlrelease my_link
```

Creating named files

This session creates a named file within an existing volumeset. By giving a name to your file, you will easily be able to find it again. You can simply access it by giving the external label of the volumeset and the name of the file and ConvexTMR will position the tape to the beginning of your file. You can only create named files if you are using either ANSI or IBM labelled volumes.

This example assumes that you are accessing the volume that was used in the previous example. It also assumes that you know the external label of this volume.

Step 1 Add a named file to the volumeset. Enter

```
rlaccess -V 000001 -F:fnew_file:A -m DAT -f \
DAT -l ANSI -RW my_link
```

- `-V` indicates that you are accessing a volumeset.
- `000001` is the external label of the volumeset that you are requesting.
- `-F` indicates that you are accessing a file in the volumeset.
- `:f` indicates that the next argument is the file's name.
- `new_file` is the name you are giving the new file.
- `:A` indicates that you are appending to the end of the volumeset.
- `-m` indicates that the next argument specifies the media type of the requested volume.
- `DAT` is the media type of the volume you are requesting.
- `-f` indicates that the next argument specifies the recording format of the requested volume.
- `DAT` is the recording format of the volume you are requesting.
- `-l` indicates that the next argument specifies the label type.
- `ANSI` is the label type of the requested volume. Possible choices for label type are `ANSI`, `IBM`, and `NL`.

- `-RW` indicates that you want read and write access to the tape.
- `my_link` is the name you are giving to a symbolic link that points to the actual tape device used.

This command pends while the operator mounts the requested volume. This may take several minutes. When the command returns, a message similar to the following is displayed:

```
Device  1:  /home/xyz/my_link
```

`/home/xyz/my_link` is the path of the symbolic link name that you use to access the tape.

Step 2 Write a file to the tape. Enter

```
echo "I'm a named file" > my_link
```

Step 3 Release the device and tape. Enter

```
rlrelease my_link
```

Accessing named files

This session accesses a named file. It assumes that you are accessing the file `new_file` that you created in the previous example. It also assumes that you know the external label of the volume that has been used in the previous examples.

Step 1 Access a named file inside of a volumeset. Enter

```
rlaccess -V 000001 -F:fnew_file -m DAT -f  \
DAT -l ANSI -RW my_link
```

- `-V` indicates that you are accessing a volumeset.
- `000001` is the external label of the volumeset that you are requesting.
- `-F` indicates that you are accessing a file in the volumeset.
- `:f` indicates that the next argument is the file's name
- `new_file` is the name you are giving the new file
- `-m` indicates that the next argument specifies the media type of the requested volume.
- `DAT` is the media type of the volume you are requesting.
- `-f` indicates that the next argument specifies the recording format of the requested volume.
- `DAT` is the recording format of the volume you are requesting.

- `-l` indicates that the next argument specifies the label type.
- `ANSI` is the label type of the requested volume. Possible choices for label type are `ANSI`, `IBM`, and `NL`.
- `-RW` indicates that you want read and write access to the tape.
- `my_link` is the name you are giving to a symbolic link that points to the actual tape device used.

This command pends while the operator mounts the requested volume. This may take several minutes. When the command returns, a message similar to the following is displayed:

```
Device  1:  /home/xyz/my_link
```

`/home/xyz/my_link` is the path of the symbolic link name that you use to access the tape.

Step 2 In Step 2 of the previous example, you wrote "`I'm a named file`" to the tape. Now read the file back and verify its contents. Enter

```
cat my_link
```

The following is displayed:

```
I'm a named file
```

Step 3 Because you accessed a named file, ConvexTMR granted you file-level access to the volume. This means that you can access the file that you requested, but ConvexTMR will not allow you to access anything else on the tape. To demonstrate this, try to read the next file on the tape. Enter

```
cat my_link
```

The following is displayed:

```
I'm a named file
```

Instead of going to another file, it printed new_file again.

Step 4 Release the device and tape. Enter:

```
rlrelease my_link
```

Writing several files to a volume

This session writes several files to an existing volume. It assumes that you are accessing the volume that was created and used in the previous examples. It also assumes that you know the external label of this volume.

Step 1 Access the volume for writing. Enter

```
rlaccess -V 000001 -m DAT -f DAT -l ANSI -RW \
my_link
```

- `-V` indicates that you are accessing a volumeset.
- `000001` is the external label of the volumeset that you are requesting.
- `-m` indicates that the next argument specifies the media type of the requested volume.
- `DAT` is the media type of the volume you are requesting.
- `-f` indicates that the next argument specifies the recording format of the requested volume.
- `DAT` is the recording format of the volume you are requesting.
- `-l` indicates that the next argument specifies the label type.
- `ANSI` is the label type of the requested volume. Possible choices for label type are `ANSI`, `IBM`, and `NL`.
- `-RW` indicates that you want read and write access to the tape.
- `my_link` is the name you are giving to a symbolic link that points to the actual tape device used.

This command pends while the operator mounts the requested volume. This may take several minutes. When the command returns, a message similar to the following is displayed:

```
Device  1:  /home/xyz/my_link
```

`/home/xyz/my_link` is the path of the symbolic link name that you use to access the tape.

Step 2 By default, you are given volume-level access to volumes that you request. This means that you can access all data that is on that volume. If you request write access to the volume, you can also write over any data that is already on the volume. You need to know what is already on your volume so that you don't overwrite any valuable data. When the ConvexTMR catalog is not used, you must know what files are on an accessed volume.

Move to the end of the data. There are several ways to do this, but for now just print the tape files to the screen. Enter

```
cat my_link
```

The following is displayed:

```
HELLO WORLD
```

(This is the unnamed file created in a previous example.) Now print the second file. Enter

```
cat my_link
```

The following is displayed:

```
I'm a named file
```

(This is the file named `new_file` that you created in a previous example.)

Step 3 Now that the tape is positioned at the end of the data, write several files to it. Enter

```
echo "I'm the third file"  > my_link
echo "I'm the fourth file" > my_link
echo "I'm the fifth file"  > my_link
echo "I'm the sixth file"  > my_link
```

Step 4 Release the device and tape. Enter

```
rlrelease my_link
```

Accessing an unnamed file

This session accesses an unnamed file on an existing volume. It assumes that you are accessing the volume that you created and used in the previous examples. It also assumes that you know the external label of this volume.

Step 1 When the ConvexTMR catalog is not used, you must know what files are on an accessed volume. Access the sixth file on the tape. This should be the file that contains the message "`I'm the sixth file`". Enter

```
rlaccess -V 000001 -F:p6 -m DAT -f DAT -l  \
ANSI -RW my_link
```

- `-V` indicates that you are accessing a volumeset.
- `000001` is the external label of the volumeset that you are requesting.
- `-F` indicates that you are accessing a file inside the volumeset.
- `:p` indicates that the next argument is the physical file number of the requested file. The physical file number represents the order in which the files appear in the volumeset.
- `6` is the physical file number of the requested file.

- `-m` indicates that the next argument will specify the media type of the requested volume.
- `DAT` is the media type of the volume you are requesting.
- `-f` indicates that the next argument will specify the recording format of the requested volume.
- `DAT` is the recording format of the volume you are requesting.
- `-l` indicates that the next argument will specify the label type.
- `ANSI` is the label type of the requested volume. Possible choices for label type are `ANSI`, `IBM`, and `NL`.
- `-RW` indicates that you are requesting read and write access.
- `my_link` is the name you are giving to a symbolic link that points to the actual tape device used.

This command pends while the operator mounts the requested volume. This may take several minutes. When the command returns, a message similar to the following is displayed:

```
Device  1:  /home/xyz/my_link
```

`/home/xyz/my_link` is the path of the symbolic link name that you use to access the tape.

Step 2 Read the file to verify that the correct file was accessed. Enter

```
cat my_link
```

The following message is displayed:

```
I'm the sixth file
```

Step 3 Release the drive and the tape. Enter

```
rlrelease my_link
```

Accessing multiple files

This session accesses multiple files on an existing volume. It assumes that you are accessing the volume that was created and used in the previous examples. It also assumes that you know the external label of this volume.

Step 1 Access the file named `new_file` and the fourth and fifth files on the tape. Enter

```
rlaccess -V 000001 -RW -F:fnew_file -F:p4 \
-F:p5 -m DAT -f DAT -l ANSI my_link
```

- `-V` indicates that you are accessing a volumeset.
- `000001` is the external label of the volumeset that you are requesting.
- `-RW` indicates that you are requesting read and write access.
- `-F` indicates that you are accessing a file inside of the volumeset.
- `:f` indicates that the next argument is the file's name.
- `new_file` is the name of the file that you are requesting.
- `-F` indicates that you are accessing a file inside of the volumeset.
- `:p` indicates that the next argument is the physical file number of the file.
- `4` is the physical file number of the requested file.
- `-F` indicates that you are accessing a file inside of the volumeset.
- `:p` indicates that the next argument is the physical file number of the file.
- `5` is the physical file number of the requested file.
- `-m` indicates that the next argument will specify the media type of the requested volume.
- `DAT` is the media type of the volume you are requesting.
- `-f` indicates that the next argument will specify the recording format of the requested volume.
- `DAT` is the recording format of the volume you are requesting.
- `-l` indicates that the next argument will specify the label type.
- `ANSI` is the label type of the requested volume. Possible choices for label type are `ANSI`, `IBM`, and `NL`.
- `my_link` is the name you are giving to a symbolic link that points to the actual tape device used.

This command pends while the operator mounts the requested volume. This may take several minutes. When the command returns, a message similar to the following is displayed:

```
Device  1:  /home/xyz/my_link
```

`/home/xyz/my_link` is the path of the symbolic link name that you use to access the tape.

Step 2 Because `new_file` is the first file that you requested, verify that it is the current file. Enter

```
cat my_link
```

The following is displayed:

```
I'm a named file
```

Step 3 Each of the three files that you requested is accessed in file-access mode. When you are in file-access mode, you are restricted to one file. If you try to read another file, you will not be reading file number 4. Since you are still restricted to `new_file`, you will read `new_file` again. Enter

```
cat my_link
```

The following is displayed:

```
I'm a named file
```

Step 4 Use the `rlnext` command to move to the next requested file. To move from `new_file` to file number 4, enter

```
rlnext my_link
```

Step 5 Read the file and verify that it is correct. Enter

```
cat my_link
```

The following is displayed:

```
I'm the fourth file
```

Step 6 Move to the last requested file. Enter

```
rlnext my_link
```

Step 7 Read and verify the file. Enter

```
cat my_link
```

The following should appear on your screen:

```
I'm the fifth file
```

If you issue another `rlnext` command, it will fail. If you want to move back to a previously accessed file, you must issue an `rlnext` command naming the specific object you want to use.

```
rlnext -F:Nnew_file my_link
```

Step 8 Release the drive and the tape. Enter

```
rlrelease my_link
```

Using mt to navigate a tape

This session uses the `mt` command to position a tape. It assumes that you are using the volume that you created and used in the previous examples.

Step 1 Acquire the volumeset for read/write access. Enter:

```
rlaccess -V 000001 -m DAT -f DAT -l ANSI -RW \
my_link
```

- `-V` indicates that you are accessing a volumeset.
- `000001` is the external label of the volumeset that you are requesting.
- `-RW` indicates that you are requesting read and write access.
- `-m` indicates that the next argument will specify the media type of the requested volume.
- `DAT` is the media type of the volume you are requesting.
- `-f` indicates that the next argument will specify the recording format of the requested volume.
- `DAT` is the recording format of the volume you are requesting.
- `-l` indicates that the next argument will specify the label type.
- `ANSI` is the label type of the requested volume. Possible choices for label type are `ANSI`, `IBM`, and `NL`.
- `my_link` is the name you are giving to a symbolic link that points to the actual tape device used.

This command pends while the operator mounts the requested volume. This may take several minutes. When the command returns, a message similar to the following is displayed:

```
Device  1:  /home/xyz/my_link
```

`/home/xyz/my_link` is the path of the symbolic link name that you use to access the tape.

Step 2 Read the first file. Enter:

```
cat my_link
```

The following is displayed:

```
HELLO WORLD
```

Step 3 Forward space one file mark (skip over the second file). Enter

```
mt -f my_link fsf 1
```

Step 4 Read the third file. Enter

```
cat my_link
```

The following is displayed:

```
I'm the third file
```

Step 5 Back up to the beginning of the second file. The tape is presently located at the beginning of the fourth file. To locate to the beginning of the second file, it is necessary to back space three filemarks, then forward space one filemark. After back spacing three filemarks, the tape is located at the end of the first file. After forward spacing one filemark, it located at the beginning of the second file. Enter

```
mt -f my_link bsf 3
mt -f my_link fsf 1
```

Note

To back up to the beginning of a file, always perform a forward space operation after a backward space operation.

Step 6 Read the second file (`new_file`). Enter

```
cat my_link
```

The following is displayed:

```
I'm a named file
```

Step 7 Rewind the tape. Enter

```
mt -f my_link rew
```

Step 8 Release the drive and the tape. Enter

```
rlrelease my_link
```

`mt` or any other program that uses tape I/O control (ioctl) calls to position the tape will work with an accessed volume. ioctl calls work transparently within the ConvexTMR context. For example, rewinding from the third volume in a three volume volumeset positions to the beginning of the first volume.

Similarly, if a volume is accessed in file access mode (i.e. you request access to only one file in a volume), rewinding positions the tape to the beginning of that file.

For more information on the mt command or ioctl calls, see the mtio(4) or the ioctl(2) man page.

Accessing multiple volumesets

This session accesses two separate volumesets. You can move from the first volumeset to the second by issuing the rlnext command. This session assumes you are accessing the volumeset that you created and used in the previous examples. It also assumes that you know the external label of this volume. You will create a second volumeset.

Step 1 Access the volumeset for read/write access and create a second volumeset. Enter

```
rlaccess -V 000001 -m DAT -f DAT -l ANSI -RW \
-V:C -m DAT -f DAT -l ANSI -RW my_link
```

- -V — indicates that you are accessing a volumeset.
- 000001 — is the external label of the volumeset that you are requesting.
- -m — indicates that the next argument will specify the media type of the requested volume.
- DAT — is the media type of the volume you are requesting.
- -f — indicates that the next argument will specify the recording format of the requested volume.
- DAT — is the recording format of the volume you are requesting.
- -l — indicates that the next argument will specify the label type.
- ANSI — is the label type of the requested volume. Possible choices for label type are ANSI, IBM, and NL.
- -RW — indicates that you are requesting read and write access.
- -V — indicates that you are accessing a volumeset.
- :C — indicates that the volumeset needs to be created.
- -m — indicates that the next argument will specify the media type of the requested volume.
- DAT — is the media type of the volume you are requesting.

- `-f` indicates that the next argument will specify the recording format of the requested volume.
- `DAT` is the recording format of the volume you are requesting.
- `-l` indicates that the next argument will specify the label type.
- `ANSI` is the label type of the requested volume. Possible choices for label type are `ANSI`, `IBM`, and `NL`.
- `-RW` indicates that you are requesting read and write access
- `my_link` is the name you are giving to a symbolic link that points to the actual tape device used.

The `-V` option must be given multiple times when you are accessing multiple volumesets.

This command pends while the operator mounts the first requested volume. This may take several minutes. When the command returns, a message similar to the following is displayed:

```
Device  1:  /home/xyz/my_link
```

`/home/xyz/my_link`is the path of the symbolic link name that you use to access the tape.

Step 2 Read the first file of the first volumeset. Enter

```
cat my_link
```

The following is displayed:

```
HELLO WORLD
```

Step 3 Use the rlnext command to move to the newly created volumeset. Enter

```
rlnext my_link
```

This command pends while the operator mounts the requested volume. This may take several minutes. When the command completes, a message similar to the following is displayed:

```
RL5198: Device: dat0: Allocate Scratch Tape '000002/000002' (ANSI)
```

The second number `000002` is the external label of the new volumeset. Make a record of the external label of the new volumeset so that you can access it again later.

Step 4 Write a file to the new volumeset. Enter

```
echo "I'm the new volumeset" > my_link
```

Step 5 Release the drive and the tape. Enter

```
rlrelease my_link
```

Using multiple devices

This session uses two different tape devices at the same time. It assumes that you are accessing the volumesets that were created and used in the previous examples. It also assumes that you know the external labels of these volumesets.

Step 1 Request that your two volumesets be placed on two separate drives. Enter

```
rlaccess -G my_link1 -V 000001 -m DAT -f DAT \
-l ANSI -RW -G my_link2 -V 000002 -m DAT -f \
DAT -l ANSI -RW
```

- `-G` indicates that the following arguments are to be grouped together.
- `my_link1` is the name that you are giving to a symbolic link that points to the first tape device used.
- `-V` indicates that you are accessing a volumeset.
- `000001` is the external label of the volumeset that you are requesting.
- `-m` indicates that the next argument will specify the media type of the requested volume.
- `DAT` is the media type of the volume you are requesting.
- `-f` indicates that the next argument will specify the recording format of the requested volume.
- `DAT` is the recording format of the volume you are requesting.
- `-l` indicates that the next argument will specify the label type.
- `ANSI` is the label type of the requested volume. Possible choices for label type are `ANSI`, `IBM`, and `NL`.
- `-RW` indicates that you are requesting read and write access.
- `-G` indicates the end of the previous grouping and that the following arguments are to be grouped together.

- `my_link2` is the name that you are giving to a symbolic link that points to the second tape device used.
- `-V` indicates that a volumeset is being accessed.
- `000002` is the external label of the volumeset that you are requesting.
- `-m` indicates that the next argument will specify the media type of the requested volume.
- `DAT` is the media type of the volume you are requesting.
- `-f` indicates that the next argument will specify the recording format of the requested volume.
- `DAT` is the recording format of the volume you are requesting.
- `-l` indicates that the next argument will specify the label type.
- `ANSI` is the label type of the requested volume. Possible choices for label type are `ANSI`, `IBM`, and `NL`.
- `-RW` indicates that you are requesting read and write access.

This command pends while the operator mounts both of the requested volumes. This may take several minutes. When the command returns, a message similar to the following is displayed:

```
Device  1:  /home/xyz/my_link1
Device  2:  /home/xyz/my_link2
```

`/home/xyz/my_link1` and `/home/xyz/my_link2` are the paths of the symbolic link names that you use to access the tapes.

If any device in the request experiences an error, the entire request is cancelled. Any devices already positioned are un-mounted and freed.

Step 2 Read the first file on the new volumeset. Enter

```
cat my_link2
```

The following is displayed:

```
I'm the new volumeset
```

Step 3 Copy the first file on the first volumeset to the new volumeset using the `dd` command. `dd` copies from an input file (`if`) to an output file (`of`) using a specified block size (`bs`). Enter

```
dd if=my_link1 of=my_link2 bs=32k
```

A message similar to the following is displayed:

```
0+1 records in
0+1 records out
```

Step 4 Use the `mt` command to rewind `second_tape`. Enter

```
mt -f my_link2 rew
```

Step 5 Read the second volumeset's first two files. Enter

```
cat my_link2
```

The following should appear on your screen:

```
I'm the new volumeset
```

Enter

```
cat my_link2
```

The following is displayed:

```
HELLO WORLD
```

This is the file that was copied from the first volumeset.

Step 6 Release both drives and tapes. Enter

```
rlrelease
```

When you issue the `rlrelease` command with no arguments, it releases all of the drives that you hold.

Acquiring two (or more) devices could be accomplished with several separate `rlaccess` commands. Issuing a single command, however, guarantees all devices and all volumes are allocated in a single atomic resource request. This safeguards against resource deadlock.

For more information about the `dd` command, see the dd(1) man page. For more information about the `mt` command, see the mt(1) man page.

Returning volumesets to the scratch state

This session places volumesets into the scratch state. When you return a volumeset to the scratch state, your data is lost and the tapes become available for others to use. You should only return your volumesets to the scratch state when you no longer need the data on the tape.

When the ConvexTMR catalog is not used, you must send a message to the ConvexTMR operator to ask that specific volumesets be scratched.

This example assumes that you are using the volumesets that you created and used in the previous examples. It also assumes that you know the external labels of these volumesets.

Step 1 Return the first volumeset to the scratch state. Enter

```
rlopmsg "I'm finished with volume 000001.  It
can be scratched."
```

Step 2 Return the second volumeset to the scratch state. Enter

```
rlopmsg "I'm finished with volume 000002.  It
can be scratched."
```

Your site may have different procedures for returning a volumeset to the scratch state. See your ConvexTMR administrator.

Using the catalog

3

Note

This is a CATALOG-ONLY chapter. If your site is operating in NO-CATALOG MODE, you do not need to read this chapter.

What is a catalog?

The ConvexTMR catalog is a comprehensive online database that tracks information about the following tape objects:

- directories
- volumes
- volumesets
- files
- rotations
- pools

Each tape object has a distinct set of information stored in the database. The catalog arranges the information to resemble the operating system file hierarchy. Many of the commands to examine or manipulate the catalog are very similar to the operating system commands to manipulate files in the file system.

The ConvexTMR catalog enables you to

- reference data on tape by symbolic name rather than by volume serial number (VSN)
- define who can read or destroy the data on tape
- allow automatic expiration of data that is no longer needed
- track media ownership, use, errors, and location

The following sections define each tape object, explain how to work with them, and describe what information is stored in the database.

For more information on any of the commands described in this chapter, refer to the man page for that command.

Working with ConvexTMR directories

A ConvexTMR directory is much like a file system directory. While a file system directory contains other directories and files, a ConvexTMR directory also contains another type of object, a volumeset. In addition, each object in a ConvexTMR directory has many more attributes than a file system object (ConvexTMR directory attributes are described below).

There are three special ConvexTMR directories:

- /pool—directory for all pool objects
- /rotations—directory for all rotation objects
- /home—base directory for user directories

ConvexTMR directory management commands (`rlcd`, `rlpwd`, `rlmkdir`, `rlmoddir`, `rlrmdir`, `rlmv`) do not affect your operating system current working directory. Rather, these commands act upon the current working directory within the ConvexTMR off-line storage hierarchy (OSH). The OSH working directory is not specific to a particular operating system shell or process; it exists within a database maintained by ConvexTMR.

The ConvexTMR working directory is shared among all processes that have the same value for the environment variable `RL_RESKEY`. This implies that multiple interactive or batch sessions can interfere with each other's view of the ConvexTMR current working directory. Unless, the environment variable `RL_RESKEY` is set to a unique value (such as the parent shell's pid).

Directory attributes

Default values for directory attributes are set by your ConvexTMR administrator. You can change the default values with the `rlmkdir`, `rlmoddir`, and `rlrmdir` commands.

Table 3 lists attributes maintained by the catalog for a directory.

Table 3 Directory attributes

Attribute	Description
owners name	directory owner's name
owners group	directory owner's group
directory acls	list of ACLs for the directory
directory comment	comment about the directory
filename template	template for creating default file names in the directory

Table 3 Directory attributes **(continued)**

Attribute	Description
file acl	default ACL for files created in this directory
default file comment	default comment for files created in the directory
default file expiration date	expiration date for files created in the directory
default file recording format	recording format for volumes referenced in the directory
volumeset name template	template for creating default volumeset names in the directory
volumeset acl	default ACL for volumesets created in the directory
default volumeset comment	default comment for volumesets created in the directory
default volumeset expiration date	expiration date for volumesets created in the directory
default volumeset recording format	recording format for volumesets referenced in the directory
default volumeset disposition	disposition of volumes referenced in volumesets in the directory
default volume erase	erases volumes released from volumesets in the directory
default file tracking	tracking for files on volumesets in the directory
default label type	label type of volumesets created in the directory
default pool	pool for volumesets created in the directory
default rotation	rotation for volumesets created in the directory
default media type	media type for volumesets created in the directory
default record format	record format for files created in the directory
default vault	vault for volumesets created in the directory

Printing the current working directory: rlpwd

Displaying your current ConvexTMR directory with `rlpwd` is much like printing the working file system directory with the `pwd` command.

When executed from the ConvexTMR `/pool` directory, the command:

```
rlpwd
```

displays the following:

```
/pool
```

Change directory: rlcd

Changing ConvexTMR directories with `rlcd` is much like changing file system directories with the `cd` command. As with the `cd` command, you may only change to directories for which you have access permission (represented by the `x` bit in the directory ACL permission string; refer to the `rllsacl`(1) man page for details).

The following command sets the current ConvexTMR directory to the directory `/pool`:

```
rlcd /pool
```

Listing contents of the current working directory: rlls

Listing the contents of a ConvexTMR catalog directory is much like listing the contents of a file system directory.

Figure 14 shows an example of displaying the current working ConvexTMR directory and listing its contents.

Figure 14 Listing contents of the current working directory

```
cnx% rlpwd
/home/johnd
cnx% rlls
Annual_System_Load_Raw_Data
cnx%
```

Figure 15 shows a slightly more detailed listing of the same directory.

Figure 15 Detailed directory listing

```
cnx% rlls -l
Vrwxoe>----->-----> johnd lp              Jun 14 23:06 Annual_System_Load_Raw_Data
cnx%
```

Permissions are described in the "Working with access control lists (ACLs)" section on page 122.

Figure 16 shows how to get a long detailed listing of each object in the directory.

Figure 16 Long directory listing

```
cnx% rlls -L

------------------------------------------------------------------------------

VOLSET                                                                  VOLSET

              name: Annual_System_Load_Raw_Data

             owner:  evan                        uperm:  rwxoedmp---s
             group:  lp                          gperm:  ------------
                                                 operm:  ------------
                                          acl entries: 0


              pool: SYSSW_DMpo               vault: silo

          rotation: NONE                    catalog: set_

        media type: 3480                 expiration: :X1/31/96

     recording fmt: 3480                disposition: scratch

         label fmt: IBM                       erase: FALSE

     file tracking: TRUE                    scratch: notscr

           comment:


Dates

       date expired: Not Expired

           creation: Jun 14 23:06

        last access: Jun 14 23:06

  last modification: Jun 14 23:06

Keys

               key: 2e007e9e00000001Z

VOLSET                                                                  VOLSET

------------------------------------------------------------------------------

cnx%
```

Creating a directory: rlmkdir

Creating ConvexTMR directories with `rlmkdir` is much like creating file system directories with the `mkdir` command. The simplest use of `rlmkdir` is demonstrated below:

```
rlmkdir tempdir
```

The command above creates the ConvexTMR directory `tempdir` beneath the current ConvexTMR working directory.

One significant difference between the ConvexTMR `rlmkdir` and file system `mkdir` commands is that ConvexTMR file and volumeset objects inherit characteristics from the directories in which they are created. When creating ConvexTMR directories, you may select default settings, for all objects that are created in that directory, for the following cataloged data:

- comments
- expiration criteria
- label type
- media type
- pool
- recording format
- storage vault
- volumeset and file name templates

Complete instructions for selecting directory default settings are provided in the `rlmkdir`(1) man page.

The following command creates a ConvexTMR directory with default settings for volumeset comment and expiration criteria:

```
rlmkdir -V -C "temp comment" -x:I tempdir
```

- `-V` indicates the following options are for volumesets
- `-C` applies the provided comment to all volumesets created in that directory
- `-x` indicates volumeset expiration. In this example, `:I` (infinite/never expired) is selected
- `tempdir` is the name of the new directory

Modifying a directory: rlmoddir

Attributes of ConvexTMR directories can be changed with the `rlmoddir` command.

Syntax for the `rlmoddir` command is the same as the `rlmkdir` syntax; refer to the `rlmoddir`(1) man page for complete details.

The following command modifies the ConvexTMR directory created in the previous example:

```
rlmoddir -V -x:X12/31/99 tempdir
```

- `-V` indicates the following modifications are to the volumeset characteristics.
- `-x` indicates volumeset expiration. In this example, `:X12/31/99` changes the expiration status from "never expired" to the expiration date December 31, 1999.
- `tempdir` is the name of the directory to change.

Removing a directory: rlrmdir

Deleting ConvexTMR directories with `rlrmdir` is much like deleting file system directories with the `rmdir` command. As in the file system, only empty directories can be removed.

The following command removes the ConvexTMR directory modified in the previous example:

```
rlrmdir tempdir
```

Moving a directory: rlmv

Moving and/or renaming named ConvexTMR directories (also files and volumesets) with `rlmv` is much like moving and/or renaming file system directories with the `mv` command.

`rlmv` cannot move or rename rotations, pools, volumes, or unnamed objects. Unnamed objects can be named with the `rlname`(1) command.

Unlike `mv`, `rlmv` does not support the usage `rlmv` *`object1 object2`*, where *`object2`* is an existing object. You must first remove the *`object2`* record from the catalog with `rlrmdir`(1) (for directories), `rlfld`(1) (for files), or `rlscratch`(1) (for volumesets).

The following commands create a new directory and move a file into the newly-created directory.

```
mkdir filesdir

rlmv myfile filesdir
```

Special ConvexTMR directories

ConvexTMR uses several reserved directories for catalog created objects. As various ConvexTMR objects are created and maintained, these object files must remain in their respective directories. These directories are:

/pool

All of the pool objects within the ConvexTMR catalog must be kept in the pool directory.

/rotations

All of the rotation objects within the ConvexTMR catalog must be kept in the rotation directory.

/home

/home is the base directory for all user directories.

Using volumeset and file name templates

ConvexTMR directories can be assigned template definitions for volumeset and file names. All volumesets and files not explicitly named at the time of creation inherit their naming scheme from their parent directory, if a template is defined in that directory. If no template is defined, ConvexTMR searches up the directory tree and employs the closest template, if one is available. If no template is available, or if the template yields an invalid name, an unnamed object is created.

Templates can specify either full or relative path names, and may contain both constant and variable text.

The following example demonstrates the creation of a directory with a volumeset name template.

Enter

```
rlmkdir -V -t pay.%y payroll
```

In 1995, the default name for a volumeset created in this directory is `pay.95`.

- `-V -t` indicates that you are defining a volumeset template.
- `pay.%y` defines the template.
- `pay.` is constant text; all volumesets created in this directory that are not explicitly assigned names when they are created will have the prefix `pay.`

- `%` indicates that what follows next is a context-sensitive variable; characters that follow this symbol are replaced with appropriate text.
- `y` is the variable for the last two digits of the year; in this example it will be replaced with the year in which the volumeset was created.
- `payroll` is the name of the new directory.

File example

The following example modifies the directory created in the previous example by adding a file naming template to the directory attributes.

Enter

```
rlmoddir -F -t pay.%2m%2d%y:G+1 payroll
```

On October 5, 1995, the default name of the first file created in this directory is `pay.10_595:G1:F0`.

- `-F -t` indicates that you are defining a file template.
- `pay.%2m%2d%y:G+1` defines the template.
- `pay.` is constant text; all files created in this directory that are not explicitly assigned names when they are created will have the prefix `pay.`
- `%` indicates that what follows next is a context-sensitive variable; characters that follow this symbol are replaced with appropriate text.
- `2m` specifies print a minimum of two `m` characters; `m` is the variable for month of the year. If the file is created in a month that has only one digit, the month will be padded to the left with an underscore character.
- `2d` specifies print a minimum of two `d` characters; `d` is the variable for day of the month. If the file is created on a day of the month that has only one digit, the day of the month will be padded to the left with an underscore character.
- `:G+1` indicates that the file generation number should be incremented by one for each subsequent file name generated by the template.

- `payroll` is the name of the directory to modify.

Many other context-sensitive variables are available for defining templates. See the rlmoddir(1) manm page for a full list.

Working with volumes

ConvexTMR expects a volume to have one-to-one relationship with a piece of media. A volume is mounted and dismounted in a single action. When the volume is mounted it is expected to be at beginning of tape (BOT) and to be rewound to BOT before dismounted.

The ConvexTMR catalog tracks the volume attributes listed in Table 4.

Table 4 Volume attributes

Attribute	Description
external label	Human-readable label usually attached to the outside of the volume.
internal label	Written in the VOL1 label for IBM and ANSI formatted volumes. If the pool the volume belongs to allows both labeled and unlabeled formats, even unlabeled volumes should have an internal label for the catalog in case they are scratched and reallocated as labeled.
slot number	Slot in a rack where the volume is stored when not in use.
container	Container where the volume is stored when archived.
finger print	Unique string generated from the first five blocks of data on the tape. Used to identify the tape when first mounted.
scheduled storage vault	Storage vault where the volume resides when it is not scratch. The specified value must be in the administrator defined list of volume storage vaults. A list of defined vaults may be obtained with the command `rlr vault`.
free storage vault	Vault where the volume resides when it is free. The specified value must be in the administrator-defined list of volume storage vaults. A list of defined vaults can be obtained with the command `rlr vault`.
pool name	Pool that the volume belongs to.

Table 4 Volume attributes **(continued)**

Attribute	Description
media type	Media Type (`mtype`). Possible values for `mtype` are site-dependent. The command `rlr mtype` produces a list of defined values.
recording format	The possible values for recording format are site-dependent. The command `rlr rformat` produces a list of defined values.
label type	Valid options include: `IBM`, `ANSI`, or `NL` (no label). The special value `NULL` indicates that no default value is given. The pool a volume belongs to may limit the values of `label_type`.
user comment	An arbitrary comment up to 80 characters in length. Enclose all comments in quotation marks.
operator comment	A comment associated with the volume by an operator. Enclose all comments in quotation marks.
volume scratch state	Scratch status: `scratch`—volume scratched; `hold`—volume held; `scrpend`—volume scratch pending; `scrvs`—volume on scratched volumeset.
owner name	The user name of the volume owner. (Default: New volumesets are owned by the user that issues the command.)
group name	The name of the group the volume belongs to. (Default: New volumesets belong to the group of the user that issues the command.)
checkout flag	Maintained by ConvexTMR. If this flag is true, the volume is checked out and mount requests fail. (Default: false.)
initialization flag	Maintained by ConvexTMR. If this flag is true, the volume is scratch. This flag is reset to false with rlinit(1). (Default: false.)

Table 4 Volume attributes **(continued)**

Attribute	Description
entry status	Maintained by ConvexTMR. Possible values: `accwait`—volume waiting acceptance; `idwait`—volume waiting identification; `avail`—volume available; `retwait`—volume waiting return.
lost flag	Maintained by ConvexTMR. If this flag is true, the volume will appear on the operator's maintenance report as lost. (Default: false.)
erase flag	Maintained by ConvexTMR. If this flag is true, the volume will appear on the operator's maintenance report as pending erasure. (Default: false.)
remove flag	Maintained by ConvexTMR. If this flag is true, and the related pool has the remove flag set to true, then the volume will appear on the operator's maintenance report as pending removal. (Default: false.)
clean flag	Maintained by ConvexTMR. If this flag is true, and the related pool has the clean flag set to true, then the volume will appear on the operator's maintenance report as pending cleaning. (Default: false.)
certify flag	Maintained by ConvexTMR. If this flag is true, and the related pool has the certify flag set to true, then the volume will appear on the operator's maintenance report as pending certification. (Default: false.)
reinitialize flag	Maintained by ConvexTMR. If this flag is true, and the related pool has the replace flag set to true, then the volume will appear on the operator's maintenance report as pending initialization. (Default: false.)
move flag	Maintained by ConvexTMR. If this flag is true the volume is pending movement. (Default: false.)

Table 4 Volume attributes **(continued)**

Attribute	Description
volume receipt	Maintained by ConvexTMR. The unique receipt number assigned the volume upon submission or retrieval. (Default: Rddddd where *ddddd* is a randomly generated, unique number.)
volumeset volume number	Maintained by ConvexTMR. If the volume is part of a volumeset, this field contains the order number of the volume within the volumeset. (Default: none.)
last modified time	Maintained by ConvexTMR. The time the volume was last mounted for writing. (Default: current time.) Last modified device. Maintained by ConvexTMR. The name of the device that last wrote to the volume. (Default: none.)
last modified host	Maintained by ConvexTMR. The name of the host that last wrote to the volume. (Default: none.)
last modified user	Maintained by ConvexTMR. The name of the user that last wrote to the volume. (Default: none.)
last accessed time	Maintained by ConvexTMR. The time the volume was last mounted for reading. (Default: current time.)
last accessed device	Maintained by ConvexTMR. The name of the device that last read from the volume. (Default: none.)
last accessed host	Maintained by ConvexTMR. The name of the host that last read from the volume. (Default: none.)
last accessed user	Maintained by ConvexTMR. The name of the user that last read from the volume. (Default: none.)
volume creation date	Maintained by ConvexTMR. The date and time the volume was created. (Default: current time.) Volume scratch date. Maintained by ConvexTMR. The date and time the volume is classified as scratch pending. (Default: none.)

Table 4 Volume attributes **(continued)**

Attribute	Description
date last cleaned	Maintained by ConvexTMR. The date and time the volume was last cleaned. (Default: none.)
lot number	This field is available for input of a purchase identification number. (Default: 0.)
blocks read	Maintained by ConvexTMR. Increments to the total number of blocks read from the volume. (Default: 0.)
blocks written	Maintained by ConvexTMR. Increments to the total number of blocks written to the volume. (Default: 0.)
mounts for read	Maintained by ConvexTMR. Increments to the number of times the volume has been mounted for reads. (Default: 0.)
mounts for write	Maintained by ConvexTMR. Increments to the number of times the volume has been mounted for writes. (Default: 0.)
recoverable read errors	Maintained by ConvexTMR. Increments to the number of recoverable read errors involving the volume. (Default: 0.)
unrecoverable read errors	Maintained by ConvexTMR. Increments to the number of unrecoverable read errors involving the volume. (Default: 0.)
recoverable write errors	Maintained by ConvexTMR. Increments to the number of recoverable write errors involving the volume. (Default: 0.)
unrecoverable write errors	Maintained by ConvexTMR. Increments to the number of unrecoverable write errors involving the volume. (Default: 0.)
other errors	Maintained by ConvexTMR. Increments to the number of non-read/write errors involving the volume. (Default: 0.)

Table 4 Volume attributes **(continued)**

Attribute	Description
mounts since last cleaned	Maintained by ConvexTMR. Increments to the number of mounts since last cleaned. Reset to 0 after cleaning is confirmed. (Default: 0.)
recoverable errors since last cleaned	Maintained by ConvexTMR. Increments to the total number of recoverable errors since last cleaned. Reset to 0 after cleaning is confirmed. (Default: 0.)
unrecoverable errors since last cleaned	Maintained by ConvexTMR. Increments to the total number of unrecoverable errors since last cleaned. Reset to 0 after cleaning is confirmed. (Default: 0.)

Adding volumes to the catalog: rlvolsubmit

To write files to a scratch tape that you keep in your possession, you need to first submit the volume to the catalog so that ConvexTMR can keep track of it. The ConvexTMR operator verifies the submission and accepts the volume into the catalog. You must supply the receipt numbers (like those shown in Figure 17) to the ConvexTMR operator to complete entry into the catalog. After the operator has completed the entry, the volumes are available as scratch volumes.

Figure 17 shows an example of submitting three 3480 cartridges to the `SYSSW_DM` pool.

Figure 17 Submitting volumes to a pool

```
cnx% rlvolsubmit -C "Data Managment" -m 3480 -p SYSSW_DM \
       -l ANSI TAC085/ADB001 TAC086/ADB001 TAC087/ADB002

Volume TAC085/ADB001 : Receipt 'R27482'
Volume TAC086/ADB001 : Receipt 'R49085'
Volume TAC087/ADB002 : Receipt 'R70687'

cnx%
```

- `-C "Data Management"` updates the `User Comment` field of the volume record in the catalog.
- `-m 3480` defines the media type.

- `-p SYSSW_DM` defines the pool in which to place the volumes.
- `-l ANSI` defines the label type on the volume.
- `TAC085/ADB001 TAC086/ADB001 TAC087/ADB002` defines the external and internal volume labels. If the external and internal label match it is not necessary to supply both.

Default values are used for those not specified on the command line. To see the set of defaults, enter

`rlr constants`

To verify the state of the submission, enter

`rlls -r vinfo :e`*<external_label>*

Modifying volumes in the catalog: rlvole

You can modify the attributes of a volume with the `rlvole` command. The example in Figure 18 modifies the vault and slot numbers of one of the volumes submitted in the example shown in Figure 17:

Figure 18 Modifying volume attributes

```
cnx% rlvole -v silo -s "0,0,10,9" :eTAC085
Volume :eTAC085 : Complete
cnx%
```

- `-v silo` specifies the new vault name.
- -s "0,0,10,9" specifies the new slot number of the volume.
- `:e`*<external_label>* specifies the external label of the volume to be edited.

You can verify the changes with the command

`rlls -r vinfo :e`*<external_label>*

Deleting volumes from the catalog: rlretrieve

Use the `rlretrieve` command to request possession of volumes and delete all records about the volumes from the database. If you want to check out volumes, an administrator can set the check-out flag with the `rlvole` command.

The example in Figure 19 retrieves the three volumes submitted in Figure 17:

Figure 19 Retrieving volumes

```
cnx% rlretrieve :eTAC085 :eTAC086 :eTAC087
receipt=R58798
receipt=R80401
receipt=R17969
cnx%
```

`The :e`*<external_label>* specifies the external label of each volume to be retrieved.

You must submit the receipt numbers to the operator so that the operator can complete the removal of the volumes from the catalog. Once the volumes are removed, the catalog no longer tracks them.

Working with volumesets

A ConvexTMR volumeset is a collection of zero or more volumes that have some logical relation to each other. You can create a volumeset before assigning volumes to it. All volumes within a volumeset must:

- have the same media type, recording format and label type
- belong to the same pool, rotation and catalog

Table 5 lists volumeset attributes.

Table 5 Volumeset attributes

Attribute	Description
volumeset name	Name given when the volumeset was created. The name may be supplied by the user or derived from a volumeset template.
volumeset generation	Cataloged volumeset generation.
volumeset version	Catalog volumeset version.
volume specification	List of unexpired volumes.

Table 5 Volumeset attributes **(continued)**

Attribute	Description
volumeset password	Specifies a volumeset password. Once a password is associated with a volumeset, the volumeset can only be read or written by users who possess both the necessary permission and the password.
volumeset comment	Arbitrary comment up to 80 characters in length. Enclose all comments in quotation marks. (Default: directory value.)
volumeset expiration date	Specifies the expiration date associated with a volumeset.
volume disposition flag	Control the disposition of volumes when the volumeset is truncated. Truncation occurs when a file on one of the first volumes in the volumeset is overwritten. Overwriting a file in this manner implicitly destroys the files that come later in the volumeset. Flags are: • scratch—truncated volumes leave the volumeset and become eligible scratch volumes. • hold— truncated volumes leave the volumeset but do not become eligible as scratch. • retain—the volumes remain attached to the volumeset and be reused the next time the volumeset requires a scratch volume. (Default: directory value.)
erase flag	If set, any volume that leaves the volumeset will be erased by the operator before it returns to scratch status. If unset, volumes that leave the volumeset are not necessarily erased. (Default: directory value.)
file tracking flag	If set, files written to the volumeset are tracked in the catalog. If unset, no file catalog entries are recorded. (Default: directory value.)

Table 5 Volumeset attributes **(continued)**

Attribute	Description
label type	Valid options include: `IBM`, `ANSI`, or `NL` (no label). (Default: directory value.) The pool a volume belongs to may limit the values of label_type.
volumeset pool	Specifies the pool from which the volumeset allocates volumes. In order to use a pool owned by somebody else, the pool owner must grant allocate permission with the `rlchacl` command. (Default: directory value.)
volume media type	The selected media type must be compatible with the selected recording format. Possible values for mtype are site-dependent. The command `rlr mtype` produces a list of defined values. (Default: directory value.) The pool a volume belongs to may limit the values of mtype.
recording format	The selected recording format must be compatible with the media type if it is given. The possible values for recording format are site-dependent. The command `rlr rformat` produces a list of defined values. The special value NULL indicates no default value. (Default: directory value.) The pool a volume belongs to may limit the values of rformat.
storage vault	Where the volumeset will reside. The specified value must be in the administrator defined list of volume storage vaults. A list of defined vaults may be obtained with the command `rlr vault`. (Default: directory value.)
rotation	Specifies a rotation to attach to the volumeset. (Default: directory value.)

Table 5 Volumeset attributes **(continued)**

Attribute	Description
owner name	The user name of the volume owner (Default: new volumesets are owned by the user that issues the creation command).
group name	The name of the group the volume belongs to. (Default: new volumesets belong to the group of the user that issues the creation command.)
volumeset creation date	Maintained by ConvexTMR.
volumeset last read date	The time the volumeset was last mounted for reading. Maintained by ConvexTMR.
volumeset last written date	The time the volumeset was last mounted for writing. Maintained by ConvexTMR.
volumeset date expired	Maintained by ConvexTMR. The expiration date of the volumeset.

Adding existing volumesets to the catalog: rlvssubmit

To read files from a tape that you keep in your possession, you must first submit the volumeset to the catalog so that ConvexTMR can keep track of it. The ConvexTMR operator verifies the submission and accepts the volumeset into the catalog. You must supply the receipt numbers (similar to those shown in Figure 20) to the ConvexTMR operator to complete entry into the catalog.

The example in Figure 20 shows how to submit a three-volume volumeset to the catalog.

Figure 20 Submitting volumesets to the catalog

```
cnx% rlvssubmit -p SYSSW_DM -v silo -m 3480 -l IBM -f 3480 \
-V TAC085,TAC086,TAC087 Annual_System_Load_Raw_Data
Volume        TAC085 : Receipt 'R04917'...Attached
Volume        TAC086 : Receipt 'R48122'...Attached
Volume        TAC087 : Receipt 'R69724'...Attached
cnx:/mnt/johnd%
```

- `-p SYSSW_DM` specifies which pool the volumes belong.
- `-v silo` specifies the vault where the volumes should be stored.
- `-m 3480` defines the media type.
- `-l IBM` defines the label type.
- `-f 3480` defines the recording format.
- `-V TAC085,TAC086,TAC087` defines the external and internal labels. This example assumes the internal and external labels are the same.
- `Annual_System_Load_Raw_Data` is the volumeset name. You can use this name to query the catalog, ask for access to the volumeset, and to perform maintenance on the set of volumes.

Default values are used for those not specified on the command line. To see the set of defaults use the command

`rlr constants`

and

`rlls -L <`*`directory_name`*`>`

where *directory_name* is the directory where the volumeset is created.

You can verify the state of the submission with the command

`rlls -r vinfo :e<`*`external_label`*`>`

Creating volumesets in the catalog: rlaccess

You can create a volumeset by using `rlaccess` with the `-V:C` switch and modifier. All attributes of the volumeset can be defined when the volumeset is created with the `rlaccess` command. See "The rlaccess command" section on page 16 for more information.

The example in Figure 21 creates a new volumeset:

Figure 21 Creating a new volumeset

```
cnx% rlaccess -m 3480 -f 3480 -v onsite -l IBM -p SYSSW_DM \
       -V:C:NRocketLaunchStats RocketLaunch_06_06_94

RL5198: Device: tc10: Allocate Scratch Tape 'TAC085/TAC085' (IBM)

Device 1: /cnx/RocketLaunch_06_06_94

cnx%
cnx% ll RocketLaunch_06_06_94
lrwxrwxrwx  1 johnd       23 Jun 20 18:17
RocketLaunch_06_06_94@ \
-> /dev/tape/upi/tc10norew
```

- `-m 3480` requests a 3480 type media.
- `-f 3480` requests that the media be written in 3480 format.
- `-v onsite` requests that the scratch tape be from the `onsite` vault.
- `-l IBM` requests IBM label type.
- `-p SYSSW_DM` the scratch tape should be from the SYSSW_DM pool.
- `-V:C:NRocketLaunchStats` volumeset specification:
  - `-V` is the volumeset switch.
  - `:C` is the create modifier.
  - `:N` is the volumeset name modifier.
- `RocketLaunch_06_06_94` is the symbolic link that points to the allocated device.

Modifying volumesets in the catalog: rlvse

You can change the catalog entry for a volumeset with the `rlvse` command. The example in Figure 22 changes the expiration date and adds a comment to the volumeset created in Figure 21.

Figure 22 Modifying a volumeset

```
cnx% rlvse -C "Old data never dies ..." -x :A100 \
Annual_System_Load_Raw_Data
cnx%
```

- `-C "Old data never dies ..."` adds a comment to the volumeset catalog entry.
- `-x :A100` changes the expiration date. The volumeset will now expire if it is not accessed in 100 days.

Modifying the volumeset expiration date: rlvse

The expiration date of a volumeset indicates when all the volumes in the volumeset are available for reuse. This means that after the expiration date, all data on the volumes is lost.

The expiration date can be defined when the volumeset is created with the `rlaccess` command or modified after creation with the `rlvse` command.

The example in Figure 23 changes the expiration date to January 31, 1996.

Figure 23 Modifying the volumeset expiration date

```
cnx% rlvse -x :X01/31/96 Annual_System_Load_Raw_Data
cnx%
```

- `-x` indicates that the expiration date is to be modified.
- `:X` specifies the date when the volumeset should be scratched.

The complete list of modifiers to the -x switch is:

- `:I` infinite (never expired).

  `cnx% rlvse -x :I Annual_System_Load_Raw_Data`

  Retains the volume set indefinitely; the volumeset can be scratched only by explicit request.

- `:S` scratch (always expired)

  ```
  cnx% rlvse -x :S temporary_volumeset
  ```

  The volumeset is not maintained after being released. This is used to create temporary volumesets.

- `:R`*NNN* expires *NNN* days after creation.

  ```
  cnx% rlvse -x :R180 \
  Annual_System_Load_Raw_Data
  ```

  The volumeset is automatically scratched 180 days after creation. You can modify the expiration date to extend the life of the volumeset or explicitly scratch the volumeset earlier.

- `:A`*NNN* expires if not accessed in *NNN* days.

  ```
  cnx% rlvse -x :A90 Annual_System_Load_Raw_Data
  ```

  The volumeset is automatically scratched after 90 days of not being accessed. You can modify the expiration date to extend the life of the volumeset or explicitly scratch the volumeset earlier.

- `:X`*mm/dd/yy* expires on given date.

  The volumeset is automatically scratched on the specified date. You can modify the expiration date to extend the life of the volumeset or explicitly scratch the volumeset earlier.

- `:G`*NNN* expires when *NNN* generations old.

  ```
  cnx% rlvse -x :G4 Annual_System_Load_Raw_Data
  ```

  After the fifth generation of the volumeset is created, the first generation is automatically scratched. When the sixth generation is created, the second generation is scratched. This continues so that only the specified number of generations are active.

Requesting possession of volumesets: rlretrieve

Use the `rlretrieve` command to request possession of volumesets.

The example in Figure 24 retrieves the volumeset created in Figure 21:

Figure 24 Retrieving a volumeset

```
cnx% rlretrieve -V Annual_System_Load_Raw_Data
volume TAC085 entry deleted
volume TAC086 entry deleted
volume TAC087 entry deleted
```

- `-V` specifies the volumeset to be retrieved.

You must take the volume list to the operator and request the volumes.

Deleting volumesets: rlscratch

When you no longer need the data on a volumeset, use the `rlscratch` command to release the volumes for reuse. You can only scratch volumesets that you own, or for which you have scratch permission. Volumeset scratch permission is represented by an `s` in the ACL permission string (ACLs are discussed in "Working with access control lists (ACLs)" section on page 122.

You can scratch only volumesets which have expired, unless you use the force option (`-f`) and are either the volumeset owner or have expiration override permission to the volumeset. Volumeset expiration override permission is represented by an `o` in the ACL permission string.

You can not scratch individual volumes that are members of a cataloged volumeset.

The following scratches (deletes) the volumeset created in Figure 21:

```
cnx% rlscratch -V Annual_System_Load_Raw_Data
```

The volumes that were previously allocated to the volumeset are now moved to the scratch state. The volumeset entry in the catalog is removed.

You can request that all the volumes within the volumeset be released, but keep the volumeset entry by using the `-E` option to the `rlscratch` command. For example,

```
cnx% rlscratch -E -V  Annual_System_Load_Raw_Data
```

releases the volumes associated with volumeset but does not delete the volumeset entry from the catalog.

Other scratch options are:

- `-F` in the place of `-V` requests scratching of the volumeset on which the specified file resides.
- `-e` schedules scratched volumes for erasure.
- `-H` places scratched volumes in the hold state rather than in the scratch state. Volumes in the hold state are reserved for future use.

Naming unnamed files and volumesets: rlname

You can give names to unnamed files and volumesets with the `rlname` command.

`rlname` cannot rename objects; this is accomplished with the `rlmv` command, described in "Moving a directory: rlmv" section on page 88.

The following is an example of using the `rlname` command to name an unnamed object:

```
rlname -V :e000001 myvolset
```

- `-V` indicates that you are naming a volumeset (use `-F` for naming files).
- `:e000001` specifies the volumeset to name; other methods of volumeset and file specification are listed in the `rlname`(1) man page.
- `myvolset` is the new name for the volumeset; you may now use this cataloged name whenever you need to reference the volumeset.

Working with files

The ConvexTMR catalog has entries for files contained in volumes under ConvexTMR control. File records can be created, modified and deleted from the catalog using ConvexTMR commands:

- rlflc
- rlfle
- rlfld

Catalog file attributes

Files are created and accessed through the ConvexTMR catalog by the following attributes:

- file path name (full or relative)
- password
- expiration date:
  - infinite (never expired)
  - scratch (always expired)
  - expiring *n* days after creation
  - expiring if not accessed in *n* days
  - expiring on a given date
  - expiring in *n* generations
- section count - number of volumes the file spans
- record format:
  - fixed-length records
  - fixed-length standard records
  - fixed-length, blocked records
  - fixed-length, blocked standard records
  - variable length records
  - variable length, blocked records
  - variable length, spanned records
  - variable length, blocked, spanned records
  - unformatted data
- keyword:
  - owner name
  - group name
  - scratch status
  - volumeset file number
  - volume file number
  - block count
  - file size

  - file creation date
  - file last accessed date
  - file last modified date
  - volumeset scratch date
- volume specification:
  - external label
  - volume key
  - volume receipt number
- file number - position on the volume
- volumeset file number - position on the volumeset
- file spec:
  - file path
  - volume identifier and file position number
  - file sequence number
  - database file key

Creating a file on a specified volume

To create a file on a specified volume, follow these steps:

Step 1 Type `rlflc` on the command line.

Step 2 Use the `-v` option followed by `:e`*`ext_lbl`* to specify a volume on which to create the file.

Step 3 Use the `-f` *`fno`* option to specify the position of the file on the volume. The *`fno`* argument is the section number on the volume of the first section of the file.

Step 4 Use the -F option to state the position of the file on the volumeset.

Step 5 Specify the file name.

Step 6 Press **Return**

The example below illustrates the steps above. Enter

```
rlflc -v:e0010 -f 1 -F 1 file1
```

Editing a file on a specified volume

To edit a files attributes on a specified volume, follow these steps:

Step 1 Type `rlfle` on the command line.

Step 2 Use the appropriate option to edit the file's password, comment, or expiration date.

Step 3 Use the `file_spec` option to specify the file position and the volume.

Step 4 Press Return

The example below illustrates the steps above. Enter

```
rlfle -x:X04/30/94 :p1@:e0010
```

Deleting a file

To delete a file from a volume, follow these steps:

Step 1 Type `rlfld` on the command line.

Step 2 Specify the file name of the file to be deleted.

Step 3 Press Return

The example below illustrates the steps above. Enter

```
rlfld file1
```

Working with pools

A tape pool is a group of volumes. Tape pools partition the tape library into sublibraries. Some organizations have a single public pool shared by all users. Other institutions assign each user a private pool. Still other sites divide their tape library into both public and private pools. There are as many ways to implement tape pools as there are tape management systems.

In the ConvexTMR tape library, each volume belongs to one pool, and each pool belongs to one owner. The owner of the tape pool has authority to grant and restrict various access permissions to individuals and groups. Although volumes can be moved among pools, a volume typically remains within the same pool its entire life. During the life of the volume it may be scratched, allocated and rescratched many times.

Types of tape pools

Tape pools provide the library with a logical way to organize groups of tapes. In determining how to implement tape pools, the library administrator has many options. Factors influencing tape pool configuration include the security and access requirements of the organization and the types of projects (individual or group, adhoc or long-term) the library should accommodate.

Default tape pools

The ConvexTMR administrator may choose to designate a specific tape pool as a default pool. If your site has a default pool, every time you enter a volume or volumeset into the library without specifying a pool, the object will be placed in the default pool. A default pool can be either a public pool or an implicitly-created private pool. These types of pools are described in the following sections.

If you would like to know if your site has a default pool, request a report with the `rlr constants` command. If your site has a default pool, it will be indicated with the `dpool=` tag. The special value `IMPOOL` indicates that the default pool is the implicitly-created private pool of the user issuing the command (see Private tape pools below).

Public tape pools

The simplest implementation of a tape pool is the public pool. Many tape libraries are composed of one or more public pools, from which users access data on active volumesets and/or draft scratch volumes into volumesets to write data to.

The "owner" of a public pool is usually the person who created the pool. If your default pool is a public pool, every tape you submit will be entered into the public pool unless you specify otherwise.

Private tape pools

ConvexTMR also supports a site-selectable option that creates private user pools automatically. If your site employs this option, the first time you enter a volume or volumeset into the library without specifying a pool, the ConvexTMR software will automatically create for you your own private tape pool. All subsequent submissions that you make will be to this pool unless you explicitly request another pool. The name of the implicitly-created pool is always `u_`*`uname`* where *`uname`* is the user name of the user submitting the volume or volumeset. For example, if your user name is lfw, your private pool, `u_lfw`, is created the first time you submit a tape to the library.

Pool creation, editing, deletion privileges

The ConvexTMR administrator reserves the right to grant pool creation, editing and deletion privileges to individual users. If you have these privileges, you can create, edit, and delete your own tape pools. For more information on these and other special privileges, see "Using special privileges," on page 147.

Tape life cycle within a pool

Every tape in a ConvexTMR pool is always in one of three states:

- scratch
- active
- maintenance

These states comprise the tape life cycle, which is discussed in detail in Chapter 1. A pool can contain both scratch volumes and volumes that belong to volumesets. When a volumeset is scratched, its constituent volumes become scratch volumes in the pool. A volume typically remains its entire life within the same pool, regardless of how many times it is drafted into a volumeset, scratched, and reallocated.

All volumes in a volumeset must belong to the same pool. If a particular pool runs out of scratch volumes, requests for scratch tapes in the affected pool will fail, but scratch requests in other pools are not affected. A pool may contain many different media types, label types, and recording formats. However, the pool owner has the option of restricting a pool so it will only accept volumes meeting certain criteria. The owner of a pool may control access to the pool. Control is also provided for allocating scratch tapes, tape submission and tape retrieval. Pools may be given a maximum capacity by the system administrator. When the number of volumes in a pool meets or exceeds the pool's maximum capacity, the pool will refuse to accept new volumes. Pools may also be given a low water mark. When the number of scratch volumes in a pool falls below the low water mark, the pool owner receives a warning message each time a scratch volume is allocated from the pool.

Using tapes from a pool

You can use scratch tapes from a pool for private data storage if you have allocation privileges granted by the pool owner.

Entering volumes into a pool

A pool must exist as a ConvexTMR object before you can submit volumes to it. Pools are either created by the ConvexTMR administrator or by users who have been granted pool creation privileges. For more information on pool creation privileges, refer to the "Configuring tape pools (POOL)" section on page 152.

If your site has a designated default pool (as indicated by the value for `dpool=` in the `rlr constants` report), it is not necessary to direct a submitted volume or volumeset to a pool unless you wish it to be submitted to a pool other than the default pool. All submissions are to the default pool unless you explicitly direct them to another pool.

A volume can enter a pool in the following ways:

- as a scratch volume
- as a member of a volumeset
- when it is edited

These methods are explained in the following sections.

Submitting scratch volumes

To submit one or more scratch volumes to an existing pool, follow these steps:

Step 1 Type `rlvolsubmit` on the command line.

Step 2 Assign any attributes you wish to the volume or volumes (optional).

Step 3 If you plan to submit more than one volume with a single `rlvolsubmit` command, indicate that (optional).

Step 4 Specify the pool the volume or volumes will belong to.

Step 5 Specify the volume or volumes to submit (optional).

Step 6 Press **Return**

The example below illustrates the steps above. Enter

```
rlvolsubmit -C "test submission" -n 5 -p poolA
000010
```

Return

```
Volume 000010 : Receipt 'R18644'
Volume 000011 : Receipt 'R19553'
Volume 000012 : Receipt 'R08506'
Volume 000013 : Receipt 'R25883'
Volume 000014 : Receipt 'R03623'
```

- `-C` allows you to assign a comment to the submitted volumes; the comment assigned to volumes in this example is "test submission."
- `-n` indicates that you are submitting multiple volumes, in this example, 5 volumes. If you provide an internal or external

label for the first volume submitted, this value will be incremented by 1 for each subsequent volume submitted.

- If you are submitting multiple volumes, but are not using a submission repetition count, you may define the volumes with a space-separated list of external labels.
- `-p poolA` indicates that the volumes are to belong to the existing pool "poolA."
- `000010` is the external label of the first volume submitted.
- The output from this command, shown above, illustrates how the volume external labels are incremented.
- The receipt numbers returned and the corresponding volumes must be presented to a tape operator for acceptance into the library.
- To view a detailed report on one of the newly-submitted volumes, including your assigned comments, the system-generated database key, pool, labels, restrictions, and other attributes, issue the following command:

```
rlls -r vinfo :e000014
```

- Many other options are available for submitting scratch volumes.

Submitting a newly-created volumeset

To submit a volumeset composed of not-yet cataloged volumes to a pool, follow these steps:

Step 1 Type `rlvssubmit` on the command line.

Step 2 Specify whether the volumeset will contain scratch volumes, active volumes, or both.

Step 3 Define the volumes to include in the volumeset.

Step 4 Direct the volumeset to a particular pool.

Step 5 Assign any attributes you wish to the volumeset (optional).

Step 6 Assign the new volumeset a name.

Step 7 Press Return

The example below illustrates the steps above. Enter

```
rlvssubmit -S 100100,100101,100102 -p poolA
myvolset
```

Return

```
Volume 100100 : Receipt 'R16884'...Attached
Volume 100101 : Receipt 'R00954'...Attached
```

```
Volume 100102 : Receipt 'R14730'...Attached
```

- `-S` indicates that the new volumeset will be composed of scratch volumes. In this example, the scratch volumes to include in the volumeset are identified by their external labels.
- `-p poolA` directs the new volumeset to the pool `poolA`.
- `myvolset` is the name of the new volumeset.
- Many other options are available for submitting volumesets.
- The return, shown above, gives the external label and receipt number of each newly-submitted volume. The receipt numbers returned and the corresponding volumes must be presented to a tape operator for acceptance into the library. "Attached" indicates that the volume is part of a volumeset object.

Submitting an empty volumeset

To submit a logical, empty volumeset to a pool, follow these steps:

Step 1 Type `rlvsc` on the command line.

Step 2 Direct the volumeset to a particular pool.

Step 3 Assign any attributes you wish to the volumeset (optional).

Step 4 Assign the new volumeset a name.

Step 5 Press **Return**

The example below illustrates the steps above. Enter

```
rlvsc -p poolA emptyset
```

- `-p` `poolA` directs the new volumeset to the pool `poolA`.
- `emptyset` is the name of the new volumeset.

Moving an unattached volume

After submitting a volume you may wish to move it to a different pool. Volumes may only be moved if they are not part of a volumeset.

Volumes that are part of volumesets cannot be moved to different pools.

To move a volume to a different pool, follow these steps:

Step 1 Type `rlvole` on the command line.

Step 2 Specify the pool that you wish to move the volume to.

Step 3 Specify the volume that you wish to move.

Step 4 Press Return

The example below illustrates the steps above. Enter

```
rlvole -p u_lfw :e000010
```

Return

```
Volume :e000010 : Complete
```

- `-p` indicates that you wish to move the volume to a different pool.
- `u_lfw` is the name of the pool you wish to move the volume to. The pool you specify must be an existing pool.
- In this example the pool name specified is the system-created private pool of user lfw. If you wish to actually work through this example, substitute the name of your private pool (if your environment creates private pools implicitly), or the name of another pool in your environment, either a public pool or a pool that you previously created.
- `:e000010` specifies the external label of the volume that you wish to move. Volumes may also be specified by volume key or receipt number.
- The `rlvole` command can only be applied to volumes that have been recorded in the catalog.

Moving an empty volumeset

After submitting an empty volumeset you may wish to move it to a different pool. Only empty volumesets may be moved to different pools.

Volumesets with member volumes cannot be moved to different pools.

To move a volumeset to a different pool, follow these steps:

Step 1 Type `rlvse` on the command line.

Step 2 Specify the pool that you wish to move the volumeset to.

Step 3 Specify the volumeset that you wish to move.

Step 4 Press Return

The example below illustrates the steps above. Enter

```
rlvse -p lfw emptyset
```

- `-p` indicates that you wish to move the volumeset to a different pool.

- `lfw` is the name of the pool you wish to move the volumeset to. The pool you specify must be an existing pool.
- In this example the pool name specified is the system-created private pool of user lfw. If you wish to actually work through this example, substitute the name of your private pool (if your environment creates private pools implicitly), or the name of another pool in your environment, either a public pool or a pool that you previously created.
- `emptyset` is the cataloged name of the volumeset that you wish to move. Volumesets may also be specified by volume key, volumeset key, or external label.
- The `rlvse` command can only be applied to volumesets that have been recorded in the catalog.

Removing a volumeset from a pool

Volumesets can be removed from a pool by using the `rlretrieve` command.

To remove a volumeset from a pool, follow these steps:

Step 1 Type `rlretrieve` on the command line.

Step 2 Use the -V option to indicate that you want to retrieve (remove) the volumeset from the pool. You can also use the -F option and specify a file in the volumeset. The entire volumeset will be removed.

Step 3 Specify the volumeset or file contained in the volumeset that you wish to retrieve.

Step 4 Press **Return**

For example, enter

```
rlretrieve -V :e0001
```

Working with rotations

A rotation is a list of vault locations and durations. A rotation can only be associated with a volumeset and defines where the volumeset is scheduled to reside during its rotation cycle. When a volumeset moves from the last vault on the rotation, it returns to the first vault in the rotation.

Storage durations are specified in days since creation of the volumeset, unless a generation-based rotation schedule is requested. A generation-based rotation schedule is useful when a number of generations of a volumeset must remain accessible.

To provide a convenient place for users to view scheduled rotations, rotation objects only exist in the /rotations directory. Therefore rotations are referenced by their name within the /rotations directory.

Creating a rotation: rlrotc

ConvexTMR allows you to create rotations. A rotation, or rotation schedule, is a list of vault locations and durations. Once you create a rotation you may assign volumes and/or volumesets to that rotation. To create a rotation you must know the list of vaults and how long each volumeset should remain in the vault. The following example uses the `rlrotc` command to create a rotation to move volumesets into the silo the day after they are created, then move them to offsite storage for six months, and finally returns them to the onsite location.

To create a rotation schedule, follow these steps:

Step 1 Type `rlrotc` on the command line.

Step 2 Specify whether the rotation is generation-based or day-based using the `-t days` or `-t gens` options (optional). The default unit is `days`.

Step 3 The `-r` option indicates that what follows next is a rotation schedule. Give the name of each vault and the duration that objects will reside at each vault you name (*`vault:n`*). You may specify up to 12 vault/duration combinations in a comma separated list.

Step 4 Give the newly-created rotation schedule a name.

The following session demonstrates the process outlined above.

Enter

```
rlrotc -t days -r  \
onsite:1,silo:30,offsite:180 SiloUsage
```

- `-t days` specifies that the rotations duration is measured in days. The other possibility is Generations, `gens`.
- `-r onsite:1,silo:30,offsite:180` specifies the rotation schedule where a volumeset should be moved to the `silo` vault 1 day after creation, moved to the offsite vault 30 after entering the silo, and should stay in the offsite vault for 180 days. The volume will then return to the onsite

vault and begin the cycle again. The syntax is *<vault_name:duration_period>*

- `SiloUsage` is the rotation's name. The name is limited to 12 characters.
- All rotations reside in the directory `/rotations`.

You can verify the creation of the rotation with the `rlls` command. For example, the command

```
cnx% rlls -L /rotations/SiloUsage
```

displays the output shown in Figure 25.

Figure 25 Verifying a rotation

```
ROTATION                                                    ROTATION

          name: SiloUsage

          owner: evan                        uperm: rxo
          group: lp                          gperm: ---
                                             operm: ---
                                   acl entries: 0

Definition
          type: DAYS
          list: onsite 1
                silo 30
                offsite 180

Keys
          key: 2e007c1f00000001R
ROTATION                                                    ROTATION
```

Modifying a rotation: rlrote

Use the `rlrote` command to modify a rotation.

The following example changes the rotation schedule for the rotation created above:

```
cnx% rlrote -r onsite:1,silo:30,offsite:90 \
SiloUsage
```

To verify the change use the `rlls` command:

```
cnx% rlls -L /rotations/SiloUsage
```

Output is shown in Figure 26.

Figure 26 Modifying a rotation

```
ROTATION                                                    ROTATION

          name: SiloUsage

          owner: evan                        uperm: rxo
          group: lp                          gperm: ---
                                             operm: ---
                                    acl entries: 0

Definition
          type: DAYS
          list: onsite 1
                silo 30
                offsite 90

Keys
          key: 2e007c1f00000001R
ROTATION                                                    ROTATION
```


Deleting a rotation: rlrotd

When you no longer need a rotation, you can delete it with the `rlrotd` command. The following example deletes the rotation previously created and modified:

```
cnx% rlrotd SiloUsage
```

`SiloUsage` is the name of the rotation to delete.

Adding a rotation to a volumeset: rlvse

When a volumeset needs to be cycled between vaults, you can associate the volumeset with a rotation. The following example shows using the `rlvse` command to associate a volumeset with the `SiloUsage` rotation created above:

```
cnx% rlvse -R SiloUsage -V \
Annual_System_Load_Raw_Data
```

Moving a volume or volumeset

Use the `rlmove` command to move a specified volume or volumeset from one vault to another. In the following example,

```
rlmove -V -v offsite /home/xyz/myvolumeset
```

- `-V` indicates that you are moving a volumeset.
- `-v` indicates that what follows is the name of the vault you wish to move the volumeset to; in this example, you are moving the volumeset to the vault "`offsite`." To view a list of available vaults, issue the following command:

```
rlr vaults
```

- `/home/xyz/myvolumeset` specifies the volumeset you wish to move.
- If the specified volumeset is already in the desired location, you will see the following message:

```
no movement necessary
```

Working with access control lists (ACLs)

An Access Control List (ACL) provides functionality similar to file protections in the file system, but allows much finer control of permissions.

By default, you are the owner of all ConvexTMR objects that you create. ConvexTMR allows you to view, set, or change access permission to objects you own. These permissions are organized by the ACL and maintained in the catalog. ACLs in ConvexTMR are not associated with the file system.

ACLs maintain permissions for the following objects:

- files
- volumesets
- directories
- pools
- rotations
- devices

Creating an ACL

ACLs are viewed with the `rllsacl`(1) command and modified with the `rlchacl`(1) command. Complete instructions are provided later in this chapter.

Every ConvexTMR ACL is composed of two parts:

- comments section
- entries section

ACL comments

The first three lines of every ACL are the ACL comments. The comments section describes the type of object that is under control and who own the object. ACL comments can be identified by the required comment initiator (#) that proceeds each comment line. The comments section has the following format

```
# <type tag>: <path name>
# owner: <owner name>
# group: <group name>
```

A sample ACL comments section for a file object is shown below.

```
# path: file1
# owner: lfw
# group: library
```

For each comment line, the text before the colon is the comment type, and the text after the colon is the comment information.

The first comment line displays the type tag (the object that is the subject of the ACL). The type tag is one of

- path
- volset
- file

depending on how the ACL was requested with the rllsacl command (by volset, path, or file).

The second comment line displays the object owner. This is always the creator of the object.

The third comment line displays the name of the group the object owner belongs to.

ACL entries

The lines following the ACL comments are the ACL entries. The entries section defines to whom and which permissions are granted. A sample ACL entries section for a file object is shown below.

```
type:file:
user::rwxoes
mask::rwx---
user:pwg:r-x---
group::rwxoes #effective:rwx---
group:accounting:r-x---
other::------
```

Each line of the ACL entries section contains three fields, separated by colons. These fields display the

- ACL entry tag type
- ACL entry qualifier
- access permissions

The fields are separated by colons in the following format:

```
<entry tag type>:<entry qualifier>:<access permissions>
```

ACL entry tags

The first field displays the ACL entry tag type. The four entry tag types are

- `type` specifies the type of object the ACL is for.
- `user` states the access granted to the identified user. If the entry qualifier (middle) field is blank, i.e. two adjacent colons, access is granted to the owner of the object only.
- `mask` specifies the maximum access granted to the members of the object owner and group class. Restrictions shown for this entry override permissions granted to individual users or groups. Every ACL that includes an entry for specific user or groups must have a "mask" entry. If the mask entry is not provided by the user it will be automatically calculated from all of the "user" and "group" entries in the ACL specification.

Note

The mask entry does not limit the protection of the "other" entry

- `group` states the access granted to the identified group. If the entry qualifier (middle) field is blank, access is granted to the group that owns the object.
- `other` specifies the access granted to any user or group that does not match any of the previous entries.

ACL entry qualifiers

The middle field displays the ACL entry qualifier. This qualifier is one of the following:

- Object type (`type` entries only)
  - file
  - volset
  - pool
  - directory
  - rotation
  - device
- User name or ID (`user` entries only)
- Group name or ID (`group` entries only)
- Empty field:
  - Empty `user` entries reference the ID of the object owner.
  - Empty `group` entries reference the ID of the owner's group.
  - `mask` and `other` entries are always empty

ACL access permissions

The last field displays the discretionary access permissions. Valid permissions for each object are listed below.

As in UNIX, the presence of an alphabetic character in a permission string indicates that the related permission is set. The presence of a hyphen (-) in a permission string indicates that the permission (represented by that position in the string) is unset.

Files: `rwxoes`

- `r` - read
- `w` - write
- `x` - view file attributes
- `o` - owner (modify attributes)
- `e` - override expiration exception
- `s` - scratch

Volumesets: `rwxoedmpbBns`

- `r` - read
- `w` - append
- `x` - view volumeset attributes
- `o` - owner (modify attributes)
- `e` - override expiration exception
- `d` - overwrite existing files

- `m` - control location
- `p` - schedule maintenance
- `b` - Bypass Label Processing (BLP) read
- `B` - BLP write
- `n` - (reserved for future use)
- `s` - scratch

Pools: `asrxo`

- `a` - allocate scratch volumes
- `s` - submit
- `r` - retrieve
- `x` - view pool attributes and members
- `o` - owner (modify attributes)

Directories: `rwxod`

- `r` - view directory entries
- `w` - add/delete owned objects
- `x` - access directory entries
- `o` - owner (modify attributes)
- `d` - delete non-owned objects

Rotations: `rxo`

- `r` - use rotation
- `x` - view rotation attributes
- `o` - owner (modify attributes)

Device objects may only be owned by ConvexTMR operators or administrators.

Devices: `rwo`

- `r` - read
- `w` - write
- `o` - (reserved for future use)

Ownership of data

Files on volumesets in the ConvexTMR tape library are owned and controlled by their creators. It is possible for a volumeset to contain files that are owned by users other than the volumeset owner. It is also possible for a file owner to wholly or partly restrict the volumeset or pool owner's access to the file.

File security is governed by ACLs. To access a file on a volumeset in a pool, a user must have access permission (with ACLs) to the file, the volumeset, and the pool.

ACL user and group entries with qualifiers

Special `user` or `group` entries can be included in an ACL to extend privileges to individuals or groups not indicated in the ACL comments section. An example is shown below.

```
# path: file1
# owner: lfw
# group: library

type:file:
user::rwxoes
mask::rwx-es
user:pwg:r-x---
group::rwxoes    #effective:rwx-es
group:accounting:r-x---
other::------
```

ACL entries without qualifiers extend privileges to the object owner only. Specific users or groups can be granted privileges if the qualifier field (between the colons) is filled in. For example, the extra `user` entry in the example above extends read and view privileges to user "`pwg`." The extra `group` entry extends read and view privileges to members of the group "`accounting`."

All group and user IDs used as ACL entry qualifiers must be registered in the `REEL/passwd` file.

The ACL mask entry

The ACL mask specifies the maximum permissions allowed to all users and groups, and overrides specific `user` and `group` entries. It does not override the entry for `other`. If the mask is not explicitly provided by the object owner, it is automatically calculated and supplied by the ConvexTMR software based on the maximum permissions allowed in all `user` and `group` ACL entries.

In the ACL example in the previous section, the mask includes all but ownership privileges; this is represented by the hyphen in the `o` position. Regardless of what privileges are extended to users and groups in later ACL entries, no users other than the original object owner may exercise ownership of the object.

Because the `other` entry is not affected by the mask, an `other` entry that includes `o` permission grants ownership to users that are not members of the listed group or other classes.

In this example, even though the main `group` entry includes the `o` bit in the list of permissions, the `mask` entry overrides owner permission. ConvexTMR automatically calculates the mask and

includes an "effective permissions" comment for all ACL entries that are effectively restricted by the mask.

Display access control list (ACL)

Prior to setting or changing an object's ACL, you may wish to view the existing ACL.

To view the ACL for a particular volumeset, follow these steps:

Step 1 Type `rllsacl` on the command line.

Step 2 Specify that you wish to view a volumeset ACL. The `-V` option indicates volumeset selection. You may choose to view the ACL for another type of object.

Step 3 Specify the volumeset. Using `:e` indicates that what follows is the volume external label of one of the volumes in the volumeset. This is just one of several methods for specifying volumes or volumesets.

Step 4 Press **Return**

Step 5 Review the access control information for the volumeset you specified when it appears on the screen.

The following session demonstrates the process outlined above.

Enter

```
rllsacl -V :e000001
```

A display similar to the following should appear on screen:

```
# path: /home/xyz/mytape
# owner: janedoe
# group: accounting
type:volumeset:
user::rwxoedmp---s
group::r-x---------
other::------------
```

- The first three lines of the return (proceeded by the symbol #) are the ACL comments. These comments inform you of the object type and name, the owner of the object, and the group the owner belongs to.
- The remainder of the return shows the ACL entries for the object you specified.

- The first line of the ACL entries shows the type of object you requested. In this example, you requested a volumeset.
- The second line of the ACL entries shows your permissions. In this example, you have permission to read, append, view volumeset attributes, modify attributes, override expiration exception, overwrite existing files, control location, schedule maintenance, and scratch. You do not have permission for Bypass Label processing read or write.
- The third line of the ACL entries shows the group permissions. In this example, the group has read and view attribute permissions only.
- The fourth line of the ACL entries shows permissions allowed to any user or group not included in the previous ACL entries. In this example, other users have no permissions.

Change access control list (ACL)

To change the ACL displayed in the previous example, follow these steps:

Step 1 Type `rlchacl` on the command line.

Step 2 Specify that you wish to change a volumeset ACL. Using the `-u` option like in the example below indicates that you wish to update a specific entry. Many other types of updates are possible.

Step 3 Indicate what kind of change you wish to make to the ACL. typing **`o::+r`** in the example below shows that you would like read permission added to the ACL for `other`. "Add read permission" is indicated by `+r`; "make update to other ACL" is indicated by `o`.

Step 4 Specify the volumeset. `-V` indicates volumeset selection. You may change the ACL for other types of objects. Using `:e` indicates that what follows is the volume external label of one of the volumes in the volumeset. This is just one of several methods for specifying volumes or volumesets.

Step 5 Press **Return**; no return indicates that your update was successful.

The following session demonstrates the process outlined above.

Enter

```
rlchacl -V -u o::+r :e000001
```

You may not change the permission bits `b` (bypass label read) or `B` (bypass label write) unless the system administrator has granted

you permission. For more information, refer to the section "Bypass label processing (BLP)" section on page 149.

File Update of a ConvexTMR Catalog ACL

The user may use the `-f` option of the rlchacl command to have the system read a complete ACL definition from a file. The file is formatted just as rllsacl prints it.

The following example demonstrates the three different command line operations to list the current ACL for a volumeset, update the file, and finally update the ACL in the catalog.

```
% rllsacl Annual_System_Load_Raw_Data > ACL.Annual_System_Load_Raw_Data

% vi ACL.Annual_System_Load_Raw_Data

% rlchacl -V -f ACL.Annual_System_Load_Raw_Data \
  Annual_System_Load_Raw_Data
```

Deleting an ACL

An ACL cannot be fully deleted but it may be reset to its default status by using the rlchacl command. The following example will delete everything but the basic ACL entries:

```
cnx% rlchacl -b -V Annual_System_Load_Raw_Data

cnx% rllsacl Annual_System_Load_Raw_Data
```

Resulting in the following output:

```
# path: Annual_System_Load_Raw_Data
# owner: janedoe
# group: lp
#
type:volset:
user::rwxoedmp---s

group::------------

other::------------
```

Pool ownership and security

Access to ConvexTMR pools is determined by the owner, or creator, of the pool, and is controlled by the pool Access Control List, or ACL.

The pool ACL

There are five different types of pool access permissions the owner, or creator, of a pool can grant to individuals, groups, and other users:

- -a permission to allocate scratch volumes from the pool
- -s permission to submit volumes and volumesets to the pool
- -r permission to retrieve volumes and volumesets from the pool
- -x permission to view pool contents and ACL information
- -o ownership (permission to modify pool ACL information)

Each of these permissions is associated with an ACL bit. If the permission bit is set in an ACL entry for a user or group, that user or group may perform that pool activity. For example, if the permission bit for scratch volume allocation, `a`, is set in the ACL entry for user `fred`, fred may allocate scratch volumes from that pool.

ACL configuration examples

Example 1 - The public pool

A common ACL configuration for a public tape pool extends allocation and viewing privileges to all users. This configuration is shown below.

```
# path: /pool/public
# owner: xyz
# group: library
#
type:pool:
user::asrxo
group::a--x-
other::a--x-
```

Notice that all the ACL bits are set for the `user` entry, but that the only bits set in the `group` and `other` entries are the `a` (allocate) and `x` (view) bits. A tape pool with this ACL allows all library users to draft scratch tapes from the pool into volumesets, and allows all library users to view pool contents and ACL information.

Because the `group` and `other` entries do not have submit, retrieve, or ownership privileges (represented with the `s`, `r`, and

o bits), these library users may not submit tapes to or retrieve tapes from the pool, and they may not make any modifications to the pool ACL.

Example 2 - The private user pool

A useful ACL configuration for a private user pool extends all pool privileges to the pool owner and no privileges to any other users or groups. This configuration is shown below.

```
# path: /pool/u_lfw
# owner: lfw
# group: stortek
#
type:pool:
user::asrxo

group::-----

other::-----
```

Notice that all the ACL bits are set for the `user` entry, and that no bits are set for the `group` or `other` entries.

Example 3 - The project pool

Another common pool configuration extends all but ownership (o) privileges to the members of a single group. This pool structure is useful for special projects, when an project team needs broad access to the data on a group of tapes. This configuration is shown below.

```
# path: /pool/project
# owner: xyz
# group: jobA
#
type:pool:
user::asrxo

group::asrx-

other::-----
```

Notice that all the ACL bits are set for the `user` entry, that all but the ownership (o) bit are set for the `group` entry, and that no bits are set for the `other` entry.

Because the `group` entry does not specify a group name in the qualifier field (indicated by the double colon) the group privileges in this ACL are extended to the group the object owner belongs to. This group is specified in the comments section (lines preceded by the pound sign). The group privileges extended in this ACL are extended to that group, the jobA group.

Ownership of volumesets within a pool

Like pools, volumesets are owned and controlled by their creators. It is possible for a pool to contain volumesets that are owned by users other than the pool owner. For this reason, a volumeset owner may wholly or partly restrict any other users, including the pool owner, from accessing the volumeset.

Pool owner status does not imply volumeset owner status. However, the pool owner may scratch expired volumesets in the pool. The pool owner may not scratch unexpired volumesets and may not force the expiration of an unexpired volumeset. Only the volumeset owner may force the expiration of a volumeset.

To access a volumeset in a pool, a user must have access permission (via ACLs) both to the pool and to the volumeset.

Using IBM and ANSI formats

4

Tape labels and the ConvexTMR catalog

ConvexTMR supports IBM and ANSI standard label tape formats. It allows you to specify the exact contents of the labels via explicit record definitions or via catalog and command parameters. It also parses existing labels and enters the literal label records and recording key parameters into the catalog. The `rlaccess` command, when used with the `-h` option, allows you to create a disk file containing user labels. The `-y` option allows you to create a disk file into which labels are placed when the specified file is read. For more information about using these options, refer to the "Read labels file" and "Write labels file" sections of Chapter 2.

The volume fingerprint

The ConvexTMR volume fingerprint helps ensure the integrity of volumes in the tape library by allowing the mount request system to recognize volumes when they are mounted. Because the fingerprint is calculated from existing data on the volume, it can be used to recognized labeled as well as unlabeled volumes.

Every time a tape is mounted, the mount request system calculates the volume fingerprint and compares it with the cataloged fingerprint of the volume the operator was asked to mount. To keep the fingerprint current, the mount request system recalculates the fingerprint and updates the catalog immediately before the volume is ejected.

Volume fingerprint contents

The volume fingerprint is simply a string of characters recorded in the catalog. This string is composed of data from label fields and other data specific to the volume. It contains the following bytes, in order:

Byte 1: Label type; one of:

- A (ANSI)

- I IBM
- U unlabeled
- X no fingerprint

Bytes 2 -7: Volume internal label; one of:

- VOL label volume serial number field (IBM)
- VOL label volume identifier field (ANSI)
- blanks unlabeled tapes

Bytes 8 - 13: HDR1 label expiration date field

Byte 14: Volume access byte; one of:

- blank IBM and unlabeled tapes
- VOL label volume access field (ANSI)

Bytes 15 - 29: Volume owner; one of:

- VOL label owner name and address field (IBM)
- VOL label owner ID field (ANSI)
- blanks unlabeled tapes

Bytes 30 - 50: Unique volume encryption; this string is generated by applying a standard algorithm to a maximum of the first 1000 bytes of the first 5 blocks on the volume. These bytes are recorded for all label types, including unlabeled tapes.

Handling unfingerprinted volumes

Because the volume internal label resides in a fixed position on the fingerprint, the mount request system can recognize a labeled volume even before its fingerprint is recorded in the catalog. This means that labeled volumes can be validated electronically the first time they are mounted.

Unrecognized fingerprints

Usually fingerprint validation is a transparent process. If a tape is cataloged incorrectly, or has been modified outside of ConvexTMR control, the fingerprint may not match the cataloged value. Depending on the severity of the discrepancy, the mount request system may fail attempts to mount the volume until the discrepancy is resolved.

In the event of a catalogued and physical fingerprint mismatch, the tape operator is asked for confirmation that the proper volume is mounted. If confirmation is given, the mount request system will either:

- update the cataloged fingerprint and allow the mount to succeed

or

- fail the mount request.

Mount requests fail when either the cataloged label type of the volume does not match byte 1 of the fingerprint, or when the cataloged internal label of a labeled volume does not match bytes 2-7 of the fingerprint. Under these circumstances, there is a significant chance that the volume was cataloged incorrectly; fulfilling the request could allow the requestor to access an incorrect volume.

Correcting catalog entries

Incorrect label types and internal label catalog records are usually the result of providing the wrong value during submission. You may correct this problem if the following conditions are true:

- the site constant `uevol=yes` (determine with `rlr constants` report)
- the user label control flag of the pool containing the volume is set to `true` (determine with long object report on the pool)

If one or both of these conditions is false, you must ask a tape operator to edit the incorrect catalog entry or entries.

If both of these conditions are true, you may correct the catalog entries yourself. To do this, follow these steps:

Step 1 Request a volume object report for the volume in question.

Step 2 Obtain the correct label type and/or internal label as displayed on the report.

Step 3 Update the volume record with the correct information.

The following session demonstrates the process outlined above.

Enter

```
rlls -r vinfo :e0001
```

An extended volume object report appears on screen.The correct label type and internal label comprise the first 7 characters of the fingerprint field. Character 1 indicates the correct label type; characters 2-7 indicate the correct internal label. The `rlls` command requests ConvexTMR reports. For more information on the volume object report, refer to "The rlls reports" section on page 163.

For the purpose of this example, the first 7 characters in the fingerprint field are

```
I000001
```

Compare these values with the cataloged values for label type and internal label. For the purpose of this example, the volume report displays the following values:

```
int label: 000100

media type: ANSI
```

Because these values do not match the correct values as displayed in the fingerprint field, you need to edit the volume record.

Enter

```
rlvole -i 000001 -l IBM :e0001

Volume :e0001 : Complete
```

- `rlvole` command edits the volume record.
- `-i 000001` specifies the correct internal label, as displayed in the fingerprint field of the volume report.
- `-l IBM` specifies the correct label type, as displayed in the fingerprint field of the volume report.
- `:e0001` specifies, by external label, the volume to edit.

Duplicate fingerprints

It is possible for two volumes to have an identical fingerprint. Except on very rare occasions, this is typically caused by duplicate (i.e. identical copies) of volumes within the library. When more then one volume has the same fingerprint, the mount request system cannot automatically recognize the volumes. In this case, the mount request system can only recognize the volume if it has an outstanding mount request for one of the volumes)

IBM standard label fields

ConvexTMR supports IBM standard label tapes as defined in the document *MVS/370 Magnetic Tape Labels and File Structure Administration, Version 3 Release 1, Order Number: SC26-4511*. This section describes the fields in the labels, gives the source of the field data, and indicates whether the field data is recorded in the ConvexTMR catalog.

Volume label: VOL

The volume header label occurs as the first file on the tape. It is only set when a tape is reinitialized. The VOL fields are described in Table 6.

Table 6 IBM volume header label

Field	Contents	Bytes	Catalog?	Source
Label ID	VOL	3	yes	system
Label Number	1	1	no	system
Volume Serial Number	alpha and/or numeric[1]	6	yes	rlaccess -V rlvolsubmit -n rlaccept -i or site convention
Reserved	0 or blank	1	no	system
VTOC Pointer	blanks	10	no	system
Reserved	blanks	16	no	system
Owner Name/ Address	alpha and/or numeric	14	yes	rlpoolc -o (catalog) volowner= constant (no catalog)
Reserved	blanks	29	no	

1. Recommended characters: A-Z, 0-9, hyphen.

Group 1 file labels: HDR1, EOF1, and EOV1

Each data file has label files before and after it. These labels are only set when a file is created or modified. The header labels consist of group 1, group 2, and user labels. The trailer labels consist of group 1 and group 2 labels. This section describes the group 1 labels: HDR1, EOF1, and EOV1.

The HDR1 label is the first tape file before the data file. The EOF1 label is the first tape file after the data file. The EOV1 label may take the place of the EOF1 label when the file is the last on the current volume and is continued on another volume.

The group 1 fields are described in Table 7.

Table 7 IBM group 1 file labels

Field	Contents	Bytes	Catalog?	Source
Label ID	HDR, EOF, or EOV	3	no	system
Label number	1	1	no	system
Data set identifier	printable characters	17	yes	rlaccess -F:f
Data set serial number	alpha or numeric	6	yes	system (preferred) or rlaccess -F:b
Volume sequence number	numeric	4	no	system (to reference: rlaccess -F:s)
Data set sequence number	numeric	4	yes	system (to reference: rlaccess -F:n or rlflc -F *vsfno*)
Generation number	numeric or blank	4	yes	rlaccess -F:g
Version number	numeric or blank	2	yes	rlaccess -F:v
Creation date	*numeric*	6	yes	system
Expiration date	*numeric*	6	yes	rlaccess -x rlflc -x rlfle -x or retain= constant
Data set security	0, 1, or 3	1	no	rlaccess -S
Block count	numeric	6	yes	system (number of blocks on first section of file)
System code	alpha or numeric	13	no	system
Reserved	blanks	7	no	system

Group 2 file labels: HDR2, EOF2, and EOV2

Table 8 defines the contents of the group 2 labels: HDR2, EOF2, and EOV2.

Table 8 IBM group 2 file labels

Field	Contents	Bytes	Catalog?	Source
Label ID	HDR, EOF, or EOV	3	no	system
Label number	2	1	no	system
Record format	F, V, or U	1	yes	rlaccess -F -r
Block length	numeric	5	yes	rlaccess -F -r
Record length	numeric	5	yes	rlaccess -f -r
Tape density	0, 1, 2, 3, or 4	1	yes	system
Data set position	0 or 1	1	no	system
Job/job step ID	blanks	17	no	system
Tape recording technique	Tb, Cb, Eb, ET, or bb	2	no	system
Control character	A, M, or b	1	no	system
Reserved	blank	1	no	system
Block attribute	B, S, R, or b	1	yes	rlaccess -F -r
Reserved	blanks	8	no	system
Checkpoint data set ID	C or blank	1	no	system
Reserved	blanks	32	no	system

User labels: UHL1 and UTL1

The IBM standard supports optional user-defined labels which trail the HDR2, EOV2, and EOF2 labels. There can be up to eight user header labels, named UHL1 through UHL8; there can be up eight user trailer labels, named UTL1 through UTL8. User header and trailer labels must be labeled consecutively.

The user label fields are described in Table 9.

Table 9 IBM user labels

Field	Contents	Bytes	Catalog?	Source
Label ID	UHL or UTL	3	no	rlaccess -h
Label number	1 - 8	1	no	rlaccess -h
User defined	any printable characters	76	no	rlaccess -h

Tapemarks

The following describes the use of tapemarks on IBM standard tapes.

- The initial tape file consists of the volume header label and the first file's file labels. A single tapemark follows the initial label file.
- The data file ends with a single tapemark.
- The trailer labels are in the next tape file. Two tapemarks are written after the trailer labels.

ANSI standard label fields

ConvexTMR supports ANSI standard label tapes as defined in the document *ANSI File Structure and Labeling of Magnetic Tape for Information Interchange*, ANSI X3.27-1987.

The ANSI standard defines a volume label, file header and trailer labels, and user labels. The following sections describe each label type.

Volume label: VOL

The first file on each tape is the volume label. The volume label fields are defined in Table 10.

Table 10 ANSI volume labels

Field	Contents	Bytes	Catalog?	Source
Label ID	VOL	3	yes	system
Label number	1	1	no	system

Table 10 ANSI volume labels **(continued)**

Field	Contents	Bytes	Catalog?	Source
Volume identifier	ANSI a characters	6	yes	rlaccess -V rlvolsubmit -n rlaccept -i or site convention
Volume access	ANSI a characters	1	no	rlaccess -8 or volacc= constant
Reserved	blanks	13	no	system
Implementation ID	ANSI a characters	13	no	system
Owner ID	ANSI a characters	14	yes	rlpoolc -o (catalog) volowner= constant (no catalog)
Reserved	blanks	28	no	system
Label standard version	4	1	no	system

Group 1 file labels: HDR1, EOF1, and EOV1

Each file on an ANSI tape has header labels HDR1 and HDR2, and trailer labels, either EOF1 and EOF2 or EOV1 and EOV2. Additional user-defined header and trailer labels can follow the required labels. These labels are named HDR3-HDR9, EOF3-EOF9 and EOV3-EOV9, and must be given consecutive names.

HDR1, EOF1 and EOF1 labels share the same format. These label fields are defined in Table 11.

Table 11 ANSI group 1 file labels

Field	Contents	Bytes	Catalog?	Source
Label ID	HDR, EOF, or EOV	3	no	system
Label number	1	1	no	system
File ID	ANSI a characters	17	yes	rlaccess -F:f
File set ID	ANSI a characters	6	yes	system (preferred) or rlaccess -F:b

Table 11 ANSI group 1 file labels **(continued)**

Field	Contents	Bytes	Catalog?	Source
File section number	numeric	4	no	system (to reference: rlaccess -F:s)
File sequence number	numeric	4	yes	system (to reference: rlaccess -F:s)
Generation number	numeric or blank	4	yes	rlaccess -F:g
Generation version number	numeric or blank	2	yes	rlaccess -F:v
Creation date	*space, digits*	6	yes	system
Expiration date	*space, digits*	6	yes	rlaccess -x rlflc -x rlfle -x or retain= constant
File access	ANSI a character	1	no	rlaccess -S
Block count	numeric	6	yes	system (number of blocks on first section of file)
Implementation ID	ANSI a characters	13	no	system
Reserved	blanks	7	no	system

Group 2 file labels: HDR2, EOF2, and EOV2

The HDR2, EOF2 and EOV2 labels share the same format. These label fields are described in Table 12.

Table 12 ANSI group 2 file labels

Field	Contents	Bytes	Catalog?	Source
Label ID	HDR, EOF, or EOV	3	no	system
Label number	2	1	no	system
Record format	F, D or S	1	yes	rlaccess -F -r
Block length	numeric	5	yes	rlaccess -F -r

Table 12 ANSI group 2 file labels **(continued)**

Field	Contents	Bytes	Catalog?	Source
Record length	numeric	5	yes	rlaccess -r
Reserved	blanks	35	no	system
Offset length	numeric	2	no	rlaccess -F -o
Reserved	blanks	28	no	system

User file labels: UHL1 and UTL1

The ANSI standard supports optional user-defined labels which trail the HDR2, EOV2, and EOF2 labels. There can be up to eight user header labels, named UHL1 through UHL8; there can be up eight user trailer labels, named UTL1 through UTL8. User header and trailer labels must be labeled consecutively.

The user label fields are described in Table 13.

Table 13 ANSI user file labels

Field	Contents	Bytes	Catalog?	Source
Label ID	UHL or UTL	3	no	rlaccess -h
Label Number	1 - 8	1	no	rlaccess -h
User Defined	any printable characters	76	no	rlaccess -h

Other File Labels: HDR3, EOF3, and EOV3

The ANSI standard supports up to seven additional header and trailer labels beyond the group 1 and group 2 labels. These labels are named HDR3-HDR9, EOF3-EOF9 and EOV3-EOV9, and must be named consecutively

The user label fields are described in Table 14.

Table 14 ANSI additional labels

Field	Contents	Bytes	Catalog?	Source
Label ID	HDR, EOF, or EOV	3	no	rlaccess -h
Label number	3 - 8	1	no	rlaccess -h

Table 14 ANSI additional labels **(continued)**

Field	Contents	Bytes	Catalog?	Source
User defined	any printable characters	76	no	rlaccess -h

Tapemarks

The following describes the use of tapemarks on ANSI standard tapes.

- The initial tape file consists of the volume header label and the first file's file labels. A single tapemark follows the initial label file.
- The data file ends with a single tapemark.
- The trailer labels are in the next tape file. Two tapemarks are written after the trailer labels.

Using special privileges

5

What are special privileges?

The ConvexTMR administrator can grant or revoke certain privileges for specified users, specified groups, or all users. These privileges are listed below.

- requesting priority levels
- bypass label processing (BLP)
- modifying the BLP permission mask
- making requests on behalf of others
- selecting specific devices
- creating tape pools
- exceeding site limits

How are privileges granted?

The ConvexTMR administrator controls user privileges via the `rlauth` command. You may view your current privileges by requesting a report with the `rlr auth` command. This is demonstrated below for user LWILLIA4.

Enter

```
rlr auth
```

```
LWILLIA4: Prio=1
Auths=BLPAR,BLPAW,BLPOR,BLPOW,BLPCM,OBO,PDA,
POOL,ESL
```

All authorizations that can be set by the ConvexTMR administrator are listed below. These privileges are defined in greater detail later in this chapter.

`Prio=`*n*	may request up to level *n* priority handling
`BLPAR`	BLP read access to all volumes
`BLPAW`	BLP write access to all volumes

BLPOR	BLP read access to owned volumes
BLPOW	BLP write access to owned volumes
BLPCM	BLP permission mask control
OBO	may make requests on behalf of others
PDA	may access specific physical devices
POOL	may configure media pools
ESL	may exceed site resource limits

Other types of privileges

Privileges granted by the ConvexTMR administrator via `rlauth` should not be confused with operator or administrator group permissions or with discretionary access permissions granted via the `rlchacl` command. These types of permissions are discussed briefly at the end of this chapter.

Using request priorities (Prio=n)

This privilege allows users to assign a priority to a `rlaccess` request. By default, user requests are queued and processed in the order in which they are received. If you have been granted this privilege, you may ask that the position of your request in the queue be determined by the priority level you specify. You may only specify a level between 1 and 9, equal to or greater than the level that the administrator has set for you. Level 1 is the highest priority.

Use of this privilege is demonstrated in the following session.

Enter

```
rlaccess -3 1 -V:C
```

- The highest priority you may request is determined by the highest priority level you have been granted. For example, if you have been granted level 5 priority, you may only request levels 9 through 5.
- `-3` indicates that this is a priority request. In this example, level 1 (highest) priority is requested; the request will be sent to the front of the queue.
- This request will be satisfied immediately if the required resources are available and any unsatisfied requests higher in the queue are not exclusively tied to the resources needed for the request.

Bypass label processing (BLP)

This group of privileges allows users to access IBM- and ANSI-labeled tapes as if they were unlabeled tapes. In BLP mode, all label files are accessible as data files, and volume and file labels are ignored. If you have been granted this privilege, you may be able to perform any or all of the following activities:

- BLP reads on volumes you own
- BLP writes to volumes you own
- BLP reads on all volumes
- BLP writes to all volumes
- BLP permission mask modification to volumes you own

If you are not familiar with the types of label files included on IBM- and ANSI-labeled tapes, see "Using IBM and ANSI formats," on page 135.

In order to perform BLP reads or writes, the associated BLP bits in the ACL of the object you are accessing must be set. For more information, refer to the `rllsacl` and `rlchacl` man pages and the "Change access control list (ACL)" section on page 129.

During default operations, if you request volume-level access to a labeled tape, you are automatically positioned at the first data file, which is actually the second physical tape file. The ConvexTMR software passes over the first tape file (the volume header label and file header label).

During BLP processing, when you access the volume you are positioned at the first physical file on the tape, the volume header label. While in this mode, you can request files by sequence number (the `-F` option; see Chapter 2). But because labels are not processed in this mode, you may not reference files by ID; the ConvexTMR software cannot recognize file IDs, because these references are stored in the header label.

Under BLP processing, end-of-volume transitions are not handled. When the last file on the tape is read, you must request via `rlnext` that the next volume in the volumeset be mounted. If the data file spans volumes, then you will have to make two reads and piece the file together on your own.

The relative order in which the BLP request occurs in the command determines which paths or objects may be accessed in BLP mode. The following rules apply:

- If `-b` before `-N` (concurrent access request), then all access paths use BLP mode.
- If `-b` before `-g` (access group), then `the` associated path uses BLP mode.
- If `-b` before a file or volumeset object, then the specified object is accessed via BLP mode.

BLP access to volumes you own (BLPOR, BLPOW)

If you have been granted BLP read and/or write access to volumes you own, a `rlr auth` request will return the values `BLPOR` and/or `BLPOW` in the string.

The following example demonstrates a request for BLP mode read access to a volumeset you own.

Enter

```
rlaccess -b -V 000001
```

- `-b` indicates BLP access.
- When read access is not specified, it is assumed.
- `-V` indicates volume-level access; this request will position to the first physical file on the tape (the volume header label).
- If you do not own the specified volume, and do not have BLP read access to all volumes in the library, the request will fail.

The following example demonstrates a request for BLP mode write access to a volumeset you own.

Enter

```
rlaccess -b -W -V 000001 -F:n3
```

- Write access must be specifically requested with the `-W` option.
- `-F` requests file access mode; `:n3` requests the third physical file on the specified volume.
- If you do not own the specified volume, and do not have BLP write access to all volumes in the library, the request will fail.

BLP access to all volumes (BLPAR, BLPAW)

If you have been granted BLP read and/or write access to all volumes, a `rlr auth` request will return the values `BLPAR` and/or `BLPAW` in the string.

If you have this privilege, you may make BLP read or write requests to all volumes in the ConvexTMR library for which you have been granted permission by the owner via the access control list (ACL; refer to Chapter 3 for more information on setting object permissions via the ACL).

Modify BLP permission mask (BLPCM) (catalog only)

This option is only available at sites that operate with a catalog.

If you have been granted this privilege, a `rlr auth` request will return the value `BLPCM` in the string.

If you have this privilege, you may change the BLP permission mask for volume objects you own.

The last field of the ACL display (requested with the `rllsacl` command) for a volume object shows the permissions. A `b` in this string indicates BLP read permission for the specified user or group; a `B` indicates BLP write permission for the specified user or group. If the administrator has granted you the privilege to change the BLP permission mask, you may change these permissions for volumes you own. This is done via `rlchacl` (refer to Chapter 3 for more information on setting object permissions via the ACL).

Reservations on behalf of others

This privilege allows you to initiate and control tape sessions on behalf of another user. If you have been granted this privilege, a `rlr auth` request will return the value `OBO` in the string.

This option is demonstrated below.

Enter

```
rlaccess -B ted -V 000001
```

- `-B` indicates that you are making a request on behalf of another user.
- `ted` is the name of the user you are making the request on behalf of.
- If you use this option without authority, or if the system does not recognize the user you specify, your request will be performed in your own behalf.

Accessing specific physical devices

This privilege allows you to request access to a specific tape device by name. If you have been granted this privileged, a `rlr auth` request will return the value `PDA` in the string.

This option is demonstrated below.

Enter

```
rlaccess -D tc0 -V 000001
```

- `-D` indicates that you are requesting a specific device.

- `tc0` is an address that specifies the desired device.
- If you use this option without authority, or if the specified drive and media object are not compatible, or if you request an invalid device, the request fails.
- To view a list of available devices, request a report with the `rlr devices` command.

Configuring tape pools (POOL)

This privilege allows you to create, edit, and delete private tape pools. If you have been granted this privilege, a rlr auth request will return the value `POOL` in the string. See the "Working with pools" section on page 111 for more information.

Creating a tape pool

To create a pool, follow these steps:

Step 1 Type `rlpoolc` on the command line.

Step 2 Assign the pool any desired restrictions and/or attributes.

Step 3 Give the pool a name.

Step 4 Press **Return**

For example, enter

```
rlpoolc -m round,8mm,3480,3490 -g 30 poolA
```

- `-m` indicates that the newly-created pool will only accept the specified media types. In this example, the pool will accept round, 8mm, 3480, and 3490 media types.
- `-g` indicates that volumesets within the pool are given an expiration grace period. In this example, volumesets will expire 30 days after all expiration criteria are satisfied.
- Many other pool restrictions and attributes are available.
- `poolA` is the name you gave the new pool. Pools always reside in the `/pool` directory. To verify that your pool was created, change to the `/pool` directory and issue an `rlls` command. This command should return a list of all objects contained in the directory; `poolA` should be among them.

Editing a tape pool

If you wish to edit the attributes of the pool created in the previous example, follow these steps:

Step 1 Type `rlpoole` on the command line.

Step 2 Add or change any restrictions and/or attributes.

Step 3 Specify the name of the pool you wish to edit.

Step 4 Press Return

For example, enter

```
rlpoole -m ANY -g 0 -l IBM poolA
```

- `-m ANY` removes the media type restrictions placed on the pool in the previous example. The pool will now accept any site-allowable media types. To view a list of the media types your site allows, issue an `rlr mtype` command.
- `-g 0` removes the grace period assigned the pool in the previous example. Volumesets in the pool will now expire immediately after all expiration criteria are met.
- `-l` (alpha) places a label type restriction on the pool. In this example, the pool will only accept volumes with IBM label formats.
- `poolA` is the name of the pool you wish to edit. If you would like to view the new attributes of `poolA`, issue an `rlls -L` command from the `/pool/poolA` directory.

Deleting a tape pool

To delete a pool, follow these steps:

Step 1 Type `rlpoold` on the command line.

Step 2 Specify the pool you wish to delete.

Enter

```
rlpoold poolA
```

- You may not delete pools that contain volumes. If you attempt to delete a pool that is not empty, you will receive a message similar to this:

```
Error: poolA: RL1342: Pool not empty
```

Exceeding site resource limits (ESL)

This privilege allows you to exceed the site resource limits as determined by the administrator. If you have been granted this privilege, a `rlr auth` request will return the value `ESL` in the string.

With this privilege, you may exceed the site limits at any time, without including any options to that effect in your `rlaccess` or `rlnext` command.

If you do not have this privilege, and would like to know your site's resource limits, request a report with the `rlr constants` command. `simdevnum=` shows the maximum number of devices that can be allocated simultaneously to one user; `jsimdevnum=` shows the maximum number of devices that can be allocated under one resource key.

Other permissions

Other permissions within ConvexTMR are not generally granted to users. The following sections explain and discuss these restrictions.

Operator and administrator group permissions

Other groups of users, such as administrators and operators, are allowed to perform certain tasks that you, as a general user, may not. Many of these tasks are performed with administrator or operator commands that you do not have access too. Other tasks, though, are performed with user commands. Although these options are shown in your user man pages, you may not select them. All restricted options are identified in the man pages with a message similar to the following:

"The following option is only available to users with level 1 administrator permission:"

If you try to use a restricted option or command without the correct permission, you will receive a message informing you that you are not authorized.

Discretionary access permissions (catalog only)

The information in this section is only correct for sites that operate with a catalog.

All ConvexTMR object owners have the authority to set and change access permissions for objects that they own. These permissions are viewed with the `rllsacl` command and changed with the `rlchacl` command. For instructions on viewing and changing object permissions, refer to Chapter 3 of this manual.

Using reports and logs

6

What are reports and logs

The extensive ConvexTMR reporting and logging functions allow you to keep track of your files, find out information about your environment, and troubleshoot.

The ConvexTMR reports make it easy for you to view statistics on your environment and on ConvexTMR objects. There are two report groups: the `rlr` reports and the `rlls` reports. The `rlr` reports list information about your site, such as available vaults, devices, recording formats, site constants, and permissions. The `rlls` reports list information about objects tracked by the ConvexTMR catalog.

ConvexTMR `rlls` reports are CATALOG-ONLY reports; they are not available at sites operating without a catalog.

The rlr reports

The following `rlr` reports are available:

- vaults
- dmodel
- rformat
- mtype
- domain
- constants
- devices
- authorizations

The rlr vaults report

The vaults report displays a list of all configured storage vaults. To generate this report, enter

```
rlr vaults
```

A sample vaults report is shown in the Figure 27 .

Figure 27 The storage vaults report

```
vaults report:                Wed May  5 13:11:24 1995

Defined Storage Vaults:

     onsite
     offsite
     archive
     mountain
```

The rlr dmodel report

The dmodel report displays the names of all configured device models. To generate this report, enter

`rlr dmodel`

A sample dmodel report is shown in Figure 28 .

Figure 28 The Device models report

```
device models report:         Wed May  5 13:11:24 1995

Configured Device Models:

     800
     1600
     3200
     6250
     qic150
     8mm
     8mmh
     dat
     dath
     3480
     3490
     disk
     3480C
     3490C
```

The rlr rformat report

The rformat report displays a list of all configured recording formats. To generate this report, enter

```
rlr rformat
```

A sample rformat report is shown in Figure 29 .

Figure 29 The recording formats report

```
rformat report:                    Wed May  5 13:13:18 1993

Configured Recording Formats:
       ①          ②
     Format     Aliases
     ------     -------
     800        nrzi       NRZI
     1600       pe         PE
     3200
     6250       grc        GRC
     qic150
     8mm        XBYTE
     8mmh       XBYTEH
     dat        DAT        4mm
     dath       DATH       4mmh
     3480
     3490
     disk
     3480C
     3490C
```

① Format name.
② Acceptable format aliases.

The rlr mtype report

The mtype report displays a list of all configured media types. To generate this report, enter

```
rlr mtype
```

A sample mtype report is shown in Figure 30 .

Figure 30 The media types report

```
mtype report:                  Wed May  5 13:15:31 1993

Configured Media Types:
        ①          ②
     Media Type   Formats
     ----------   -------
     round        800       1600      3200      6250
     qic150       qic150
     8mm          8mm       8mmh
     dat          dat       dath
     3480         3480
     disk         disk
     3490         3490
```

① Media type.

② Acceptable formats for the listed media type.

The rlr domain report

The domain report displays a list of all current domains. To generate this report, enter

```
rlr domain
```

A sample domain report is shown in Figure 31 .

Figure 31 The domains report

```
domain report:              Wed May  5 13:15:31 1993

Domain Status:
        ①         ②
     Domain     State
     ------     -----
     default    Down
     dsim       Down
     t1         Down
```

① Domain — The name of an operator domain that includes certain devices.

② State — up or down. If down, requests for devices requiring the domain will be rejected.

The rlr constants report

The constants report displays a list of current site constants. To generate this report and view it one screen at a time, enter

```
rlr constants|more
```

A sample page of the constants report is shown in Figure 32 .

Figure 32 The site constants report

```
# Configuration Constants:
mtremtime=300
    inactwtime=1800
    inactrtime=3600
        aptime=3600
     simdevnum=3
    jsimdevnum=3
        retain=180
         rpoll=60
    pretimeout=600
    opresponse=0
        clnmnt=0
        clnrer=0
        clnuer=0
        remmnt=0
        remrer=0
        remuer=0
        ctfmnt=0
        ctfrer=0
        ctfuer=0
        rplmnt=0
        rplrer=0
        rpluer=0
       poolcap=0
       maxtlog=0

      acctfile=/tmp/RL.acct
        dbjdir=
         dpool=IMPOOL
      stdlabel=ANSI
       rmpmode=process
       rmqmode=queue
      volowner=
--More--
```

The rlr devices report

The devices report displays a list of all configured devices. To generate this report, enter

```
rlr devices
```

A sample devices report is shown in Figure 33 .

Figure 33 The devices report

```
devices report:                    Wed Jul 21 11:20:03 1993

Current device status;
   ①        ②        ③        ④      ⑤       ⑥          ⑦      ⑧  ⑨      ⑩
Device     Model    Status    Tape  INTLBL  EXTLBL       ReqID   Dn User    Key
------     -----    ------    ----  ------  ------       -----   -- ----    ---
drive3     3480     robin     idle  ------  ------
drive2     3490     robin     idle  ------  ------
drive1     3490     robin     idle  ------  ------
```

①	Tape drive name	The name of the tape device.
②	Device model	The tape device model.
③	Tape device status	The status of the tape device.
④	Tape volume status	The status of the volume.
⑤	Internal label	The internal label of the mounted volume.
⑥	External label	The external label of the mounted volume.
⑦	Request ID	The request ID of the request engaging the specified device.
⑧	Device number	The device number in a multiple drive request. For requests using only one drive, this field is blank.
⑨	User	The ID of the user of the device.
⑩	Resource key	The resource key associated with the request, if applicable.

The rlr authorizations report

The authorizations report displays all the special privileges extended to you by the system administrator. To generate this report, enter

`rlr auth`

A sample authorizations report is shown in Figure 34 . ConvexTMR authorizations are defined in Chapter 6, "Using Special Permissions."

Figure 34 The authorizations report

```
auths report:                    Wed Jul 21 11:20:46 1993

     lfw: Prio=1   Auths=BLPAR,BLPAW,BLPOR,BLPOW,BLPCM,OBO,PDA,POOL,ESL
```

The rlls reports

ConvexTMR `rlls` reports are CATALOG-ONLY reports; they are not available at sites operating without a catalog.

The following `rlls` reports are available:

Object lists:

- simple object list
- detailed object list
- extended object list
- object comments list
- object expiration list
- file volume association list
- file volumeset association list

Extended object reports:

- extended file report
- extended volume report
- extended volumeset report

User lists:

- user file list
- user volume list
- user volumeset list

Contents lists:

- volumeset volume list
- volumeset file list
- volume file list

Pool lists:

- user pool list
- pool volume list

- pool scratch volume list

Other lists:

- inventory list
- maintenance list

The object lists: rlls

The simple object list: rlls (*path*)

The simple object list displays the contents of the current or specified ConvexTMR catalog directory. Its output is similar to the UNIX `ls` command.

To view a simple list of all objects in the current OSH directory, enter

```
rlls
```

To view objects in another ConvexTMR catalog directory, specify the directory path in the command, as shown in the following example:

```
rlls directory
```

A simple object list is shown in Figure 35 .

Figure 35 The simple object list

```
monthly
cycle
temp
file1
file2
file3
directory
```

The detailed object list: rlls -l (*path*)

The detailed object list displays the contents of the current or specified ConvexTMR catalog directory. Its output is similar to the UNIX `ls -l` command.

To view a detailed list of all objects in the current ConvexTMR catalog directory, enter

```
rlls -l
```

A sample detailed object list is shown in Figure 36 .

Figure 36 The detailed object list

```
      ①          ②        ③        ④        ⑤          ⑥
Vrwxoe>----->----->  lfw     stortek          Jul 20 14:54 emptyset
Vrwxoe>----->----->  lfw     stortek          Jul 20 14:55 growingset
-rwxoes-----------   lfw     stortek    UNSET Jul 20 14:56 file1
drwxod-r-x---r-x---  lfw     stortek                       directory
```

① Permissions field. The first character defines the entry, as follows:

- `-` file
- `v` volumeset
- `p` pool
- `d` directory
- `r` rotation

② Object owner.

③ Group name.

④ File size (file objects only).

⑤ Modification date.

⑥ Relative path name.

The extended object list: rlls -L (*path*)

The extended object list displays comprehensive data on the contents of the current or specified ConvexTMR catalog directory.

To view an extended object list for the contents of the current ConvexTMR catalog directory, enter

```
rlls -L
```

A sample extended object list is shown in Figure 37 .

Figure 37 The extended object list

```
VOLSET                                                          VOLSET

               name: lynne
              owner: lfw                        uperm: rwxoedmpbBns
              group: stortek                    gperm: ------------
                                                operm: ------------
                                          acl entries: 0
               pool: public                      vault: onsite
           rotation: NONE                      catalog: set_
         media type: 3490                   expiration: DEFAULT
      recording fmt: 3490                  disposition: scratch
          label fmt: IBM                         erase: FALSE
       file tracking: TRUE                     scratch: notscr
            comment:
Dates
        date expired: Not Expired
            creation: May 06 13:03
         last access: May 06 13:03
  last modification: May 06 13:03

Keys
                 key: 2be944e100000001Z
```

The object comments list: rlls -c (*path*)

The object comments list displays operator comments for volume objects, and user comments for all other objects, in the current or specified ConvexTMR catalog directory.

To view an object comments list for the contents of the current ConvexTMR catalog directory, enter

```
rlls -c
```

A sample object comments list is shown in Figure 38 .

Figure 38 The object comments list

```
myvolset      <No Comment>
file1         this is a temp file
file2         <No Comment>
file3         <No Comment>
file4         <No Comment>
```

The object expiration list: rlls {-e | -E} (*path*)

The object expiration list displays expirations for all objects in the current or specified ConvexTMR catalog directory. The `-e` option invokes the standard expiration list; the `-E` option invokes a verbose expiration list.

To view a standard object expiration list for the contents of the current ConvexTMR catalog directory, enter

```
rlls -e
```

A sample object comments list is shown in Figure 39 .

Figure 39 The object expiration list

```
  ①          ②      ③          ④
emptyset     :X     179        days
growingset   :X     179        days
file1        :I
```

① Object name.

② Expiration status:

- :I infinite (never expired)
- :S scratch (always expired)
- :R retain specified number of days
- :A access within specified number of days, or expire
- :X expire in specified number of days
- :G expire after specified number of generations

③ Days or generations until expiration.

④ Expiration unit.

The file volume association list: rlls -v (*path*)

The file volume association list displays, for each file in the current or specified OSH directory, the external label of the volume the file is stored on.

To view a file volume association list, enter

```
rlls -v
```

A sample file volume association list is shown in Figure 40 .

Figure 40 The file volume association list

```
file1           :e0001
file2           :e0001
file3           :e0001
foo             :e0005
foo2            :e0005
```

The file volumeset association list: rlls -V *(path)*

The file volumeset association list displays, for each file in the current or specified ConvexTMR catalog directory, the full ConvexTMR catalog path of the associated volumeset.

To view a file volumeset association list, enter

```
rlls -V
```

A sample file volumeset association list is shown in Figure 41 .

Figure 41 The file volumeset association list

```
file1           /poolv/irma/myvolset
file2           /poolv/irma/myvolset
file3           /poolv/irma/myvolset
foo             /poolv/irma/newvolset
foo2            /poolv/irma/newvolset
```

The extended object reports: rlls -r *report object*

The extended file report: rlls -r finfo *file_spec*

The extended file list displays comprehensive data for the specified file. The file must be specified by *file_spec*; refer to the rlls(1) man page for a complete definition of *file_spec*. This is demonstrated below.

```
rlls -r finfo poolv/irma/file1
```

A sample extended file list is shown in Figure 42 .

Figure 42 The extended file list

```
FILE                                                                FILE

             name: file1
            owner: lfw                    uperm: rwxoes
            group: wheel                  gperm: ------
                                          operm: ------
                                  acl entries: 0
          comment:
Location
         on volset: 2c27612b00000001Z   volume: 2c270e5200000001V
     volset file #: 2                vol file #: 2
    file seek addr: UNSET             sections: UNSET
Label
          File Set: set_                    Generation: UNSET
           File ID: set_                       Version: UNSET
Statistics
     record format: u:B65535:R65535
       block count: UNSET
         file size: UNSET
        expiration: :R30
           scratch: notscr
Dates
          creation: Jul 01 11:43
       last access: Jul 01 11:43
 last modification: Jul 01 11:43

Keys
               key: 2c33063000000001F
FILE                                                                FILE
```

The extended volume report: rlls -r vinfo *vol_spec*

The extended volume report displays comprehensive data for the specified volume. The volume must be specified by *vol_spec*. This is demonstrated below.

```
rlls -r vinfo :e0001
```

A sample extended volume report is shown in Figure 43 .

Figure 43 .The extended volume report

```
VOL                                                                    VOL
            ext label: 0001          cur location: onsite
            int label: 0001        sched location: onsite
                owner: lfw          free location: onsite
                group: stortek          container:
                 pool: irma                 slot:
               volset: 2c27612b00000001Z receipt:
        # in volset: 1                       key: 2c270e5200000001V
         lot number: 0
        fingerprint: X             Uninspected
  Format
         media type: 3490            rec format: 3490
         label type: IBM
  Status
             scratch: volumeset             move: FALSE
          initialize: FALSE               erase: FALSE
                 new: not used           remove: FALSE
        checked out: FALSE               clean: FALSE
              entry: avail              certify: FALSE
               lost: FALSE              reinit: FALSE
                                        replace: FALSE
  Last access
       write on:                           read on:
    using drive:                        using drive:
        by user:                            by user:
             on: Jun 22 09:51                    on: Jun 30 11:14
  Statistics
        blocks read: 0           read error(r): 0
       blocks write: 0           read error(u): 0
   read-only mounts: 0          write error(r): 0
    writable mounts: 0          write error(u): 0
 mounts since clean: 0            other errors: 0
                         errors since clean(r): 0
                         errors since clean(u): 0
Dates
     record creation: Jun 22 09:51
        scratch date: Jun 22 09:51
          last clean: Dec 31 1969
Comments
        user comment:
          op comment:
VOL                                                                    VOL
```

The extended volumeset report: rlls -r vsinfo *vs_spec*

The extended volumeset report displays comprehensive data for the specified volumeset. The volumeset must be specified by *vs_spec*; This is demonstrated below:

```
rlls -r vsinfo myvolset
```

A sample extended volumeset report is shown in Figure 44 .

Figure 44 The extended volumeset report

```
VOLSET                                                          VOLSET
               name: myvolset
              owner: lfw                        uperm: rwxoedmpbBns
              group: stortek                    gperm: ------------
                                                operm: ------------
                                          acl entries: 0
                pool: irma                      vault: onsite
            rotation: NONE                    catalog: set_
          media type: 3490                 expiration: :X12/12/99
      recording fmt: 3490                 disposition: scratch
          label fmt: IBM                        erase: FALSE
      file tracking: TRUE                     scratch: notscr
             comment:
Dates
       date expired: Not Expired
           creation: Jun 22 15:44
        last access: Jun 22 15:44
  last modification: Jun 22 15:44
Keys
                key: 2c27612b00000001Z
VOLSET                                                          VOLSET
```

The user lists: rlls -r *list* {*uname* | ALL}

The user lists display critical physical information for objects owned by the specified user or by all users (specified with ALL).

The user file list: rlls -r flist {*uname* | ALL}

The user file list displays critical physical information for all files owned by the specified user or by all users (specified with ALL). This is demonstrated below:

```
rlls -r flist lfw
```

A sample user file list is shown in Figure 45 .

Figure 45 The user file list

```
User File Information                        Thu Jul  1 11:49:44 1993

User: lfw

      ①          ②        ③        ④        ⑤          ⑥
    extlbl      vno      vsno     sect     expire     path
    ------      ------   -----    ----     ------     ----
    0001        1        1        1        :I         /tempfile
    0001        2        2        1        :I         /poolv/irma/file1
    0001        3        3        1        :I         /poolv/irma/file2
    0001        4        4        1        :I         /poolv/irma/file3
    0001        5        5        1        :I         /home/lfw/file4
```

①	External label	The external label of the tape on which the file begins.
②	Volume number	The position among other files the file occurs in on the volume.
③	Volumeset number	The position among other files the file occurs in on the volumeset.
④	Sections	The number of physical volumes the file spans.
⑤	Expiration	Expiration date and/or status.
⑥	Path	Relative path of the file (or file key, for unnamed files).

User volume list: rlls -r vlist {*uname* | ALL}

The user volume list displays critical physical information for all volumes owned by the specified user or by all users (specified with ALL). This is demonstrated below:

```
rlls -r vlist lfw
```

A sample user volume list is shown in Figure 46 .

Figure 46 The user volume list

```
User Volume Information                    Sat May 15 13:04:22 1993
User: lfw

     ①          ②          ③          ④          ⑤          ⑥          ⑦
   extlbl     intlbl     mtype     rec_fmt     vault      pname      status
   ------     ------     -----     -------     -----      -----      -----
   0001       0001       3490      3490        offsite    pub        use
   0002       0002       3490      3490        offsite    pub        use
   0003       0003       3490      3490        offsite    pub        use
   0004       0004       3490      3490        offsite    pub        scr
   0005       0005       3490      3490        onsite     pub        scr
   0006       0006       3490      3490        onsite     pub        scr
   0007       0007       3490      3490        onsite     pub        scr
   0008       0008       3490      3490        onsite     pub        scr
   0009       0009       3490      3490        onsite     pub        scr
```

① External label.
② Internal label.
③ Media type.
④ Recording format.
⑤ Vault.
⑥ Pool name.
⑦ Volume status code. These codes are defined below.

use - part of a volumeset
scr - scratch volume
hld - hold state
scrp - scratch pending
ops - operator scratch volume
out - checked out
lst - lost
ret - awaiting return
rem - awaiting removal
rep - awaiting replacement
- cln - awaiting cleaning
- ctf - awaiting certification
- ers - awaiting erasure
- ini - awaiting initialization

- idw - awaiting identification
- mov - awaiting movement to another vault
- acc - awaiting acceptance

User volumeset list: rlls -r vslist {*uname* | ALL}

The user volumeset list displays critical physical information for all volumesets owned by the specified user or by all users (specified with `ALL`). This is demonstrated below.

```
rlls -r vslist lfw
```

A sample user volumeset list is shown in Figure 47 .

Figure 47 The user volumeset list

```
 User Volumeset Information                Sat May 15 13:12:14 1993

User: lfw

    ①     ②          ③          ④          ⑤              ⑥           ⑦
  label  mtype      rec_fmt    vault      pname          expire      path
  -----  -----      -------    -----      -----          ------      ----
  IBM    3490       3490       onsite     irma           :X12/12/99  ./myvolset
  IBM    3490       3490       onsite     irma           :L100       ./newvolset
```

① Label type
② Media type
③ Recording format
④ Vault
⑤ Pool name
⑥ Expiration date or status
⑦ Relative path name (or volumeset key, for unnamed volumesets)

The contents lists: rlls -r *list object*

The volumeset volume list: rlls -r vsvlist *vol_spec*

The volumeset volume list displays a list of all the volumes that are members of the specified volumeset. The volume must be specified by *vol_spec*. This is demonstrated below.

```
rlls -r vsvlist myvolset
```

A sample volumeset volume list is shown in Figure 48 .

Figure 48 The volumeset volume list

```
Volumeset Volume                          Sat May 15 13:19:55 1993

Volset: myvolset

     ①          ②              ③           ④
   vsvn       extlbl         vault       status
   ----       ------         -----       ------
   1          0001           offsite     active
   2          0002           offsite     active
   3          0003           offsite     active
```

① Volumeset volume number. The order in which the volume occurs in the volumeset.

② External label.

③ Vault.

④ Status (active or scratch).

The volumeset file list: rlls -r vsflist *vs_spec*

The volumeset file list displays a list of all the files that reside on the specified volumeset. The volumeset must be specified by *vs_spec*. This is demonstrated below.

```
rlls -r vsflist myvolset
```

A sample volumeset file list is shown in Figure 49 .

Figure 49 The volumeset file list

```
Volumeset File                            Mon Jul  5 11:53:21 1993

Volset: myvolset

     ①          ②     ③     ④     ⑤          ⑥
   extlbl       vno   vsno  sect  expire     path
   ------       ----  ----  ----  ------     ----
   0001         1     1     1     :I         /tempfile
   0001         2     2     1     :R30       /poolv/irma/file1
   0001         3     3     1     :I         /poolv/irma/file2
   0001         4     4     1     :G5        /poolv/irma/file3
   0001         5     5     1     :I         /home/lfw/file4
```

① External label of the volume the file begins on.

② Volume file number. The order in which the file occurs on the volume.

③ Volumeset file number. The order in which the file occurs on the volumeset.

④ Section. The number of tapes the file spans.

⑤ Expiration date or status.

⑥ Relative path name (or file key, for unnamed files).

The volume file list: rlls -r volflist *vol_spec*

The volume file list displays a list of all the files that reside on the specified volume. The volume must be specified by *vol_spec*. This is demonstrated below.

```
rlls -r volflist :e0001
```

A sample volume file list is shown in Figure 50 .

Figure 50 The volume file list

```
Volume File                                  Mon Jul  5 12:01:32 1993

Volume: 0001

     ①        ②            ③
     vno      expire       path
     ----     ------       ----
     1        :I           /tempfile
     2        :R30         /poolv/irma/file1
     3        :I           /poolv/irma/file2
     4        :G5          /poolv/irma/file3
     5        :I           /home/lfw/file4
```

① Volume file number. The order in which the file occurs on the specified volume.

② Expiration date or status.

③ Relative path name (or file key, for unnamed files).

The pool lists: rlls -r *list list_opts*

The pool lists display critical information about ConvexTMR pools.

The user pool list: rlls -r plist *uname*

The user pool list displays critical information about all pools owned by the specified user. This is demonstrated below.

```
rlls -r plist lfw
```

A sample user pool list is shown in Figure 51 .

Figure 51 The pool list

```
User Pool List                                  Thu Jul  1 11:56:35 1993

   ①           ②          ③        ④        ⑤        ⑥
  pname       svault     num_scr  tot_vol  low      maxvol
  -----       ------     -------  -------  ---      ------
  irma        onsite     10       10       10       UNSET
  new         onsite     0        0        UNSET    1000
```

① Pool name.
② Storage vault (current).
③ Number of scratch volumes in pool.
④ Total number of volumes in pool.
⑤ Low water mark. If the number of scratch tapes falls below this value, the pool owner receives warning messages.
⑥ Pool capacity. The maximum number of volumes allowed in the pool.

The pool volume list: rlls -r pvlist *pool*

The pool volume list displays critical information about all the volumes in the specified pool. This is demonstrated below.

```
rlls -r pvlist irma
```

A sample pool volume list is shown in Figure 52 .

Figure 52 The pool volume list

```
Pool Volume List                              Thu Jul  1 12:00:14 1993

Pool: irma

   ①          ②        ③        ④        ⑤        ⑥        ⑦    ⑧
vol_loc     intlbl   label    vault    mtype    rec_fmt   stat owner
-------     ------   -----    -----    -----    -------   ---- -----
0001        0001     IBM      onsite   3490     3490      use  lfw
0002        0002     IBM      onsite   3490     3490      use  lfw
0009        0009     IBM      onsite   3490     3490      scr  ----
0010        0010     IBM      onsite   3490     3490      scr  ----
0003        0003     IBM      onsite   3490     3490      acc  lfw
0004        0004     IBM      onsite   3490     3490      acc  ----
0005        0005     IBM      onsite   3490     3490      acc  ----
0006        0006     IBM      onsite   3490     3490      acc  ----
0007        0007     IBM      onsite   3490     3490      acc  ----
0008        0008     IBM      onsite   3490     3490      acc  ----
```

① Volume location (given as *ext_lbl*|[*slot*]|[*container*])
② Internal label
③ Label type
④ Vault (current)
⑤ Media type
⑥ Recording format
⑦ Volume status code (defined in the *User Volume List* section)
⑧ Owner

The pool scratch volume list: rlls -r pscrlist (-x *vault*) *pool*

The pool scratch volume list displays critical information about all scratch volumes in the specified pool. This is demonstrated below.

```
rlls -r pscrlist -x onsite irma
```

A sample pool scratch volume list is shown Figure 53

Figure 53 The pool scratch volume list

```
Pool Scratch                              Thu Jul  1 11:59:23 1993

Vault: onsite

    ①                    ②         ③        ④         ⑤
  vol_loc                label     intlbl   mtype     rec_fmt
  -------                -----     ------   -----     -------
  0009                   IBM       0009     3490      3490
  0010                   IBM       0010     3490      3490
```

① Volume location (given as *ext_lbl*|[*slot*]|[*container*])
② Label type
③ Internal label
④ Media type
⑤ Recording format

Other lists: rlls -r *list list_opts*

The inventory list: rlls -r inventory (*vault*|ALL)

The inventory list displays critical physical information for all volumes in the library or in the specified vault. This is demonstrated below.

```
rlls -r inventory onsite
```

A sample inventory list is shown in Figure 54 below.

Figure 54 The inventory list

```
Vault Inventory                          Sat May 15 13:15:55 1993

Vault: onsite

   ①                ②      ③     ④        ⑤        ⑥          ⑦
vol_loc          intlbl  label mtype    rec_fmt  pname      owner
-------          ------  ----- -----    -------  -----      -----
0005             0005    IBM   3490     3490     pub        lfw
0006             0006    IBM   3490     3490     pub        lfw
0007             0007    IBM   3490     3490     pub        lfw
0008             0008    IBM   3490     3490     pub        lfw
0009             0009    IBM   3490     3490     pub        lfw
0010             0010    IBM   3490     3490     pub        lfw
```

① Volume location (given as *ext_lbl*|[*slot*]|[*container*])
② Internal label
③ Label type
④ Media type
⑤ Recording format
⑥ Pool name
⑦ Owner

The maintenance list: rlls -r maint (-a *state*) (*vault*)

The maintenance list displays, by external label, a list of all the volumes in the library or in the specified vault that are undergoing any or specified maintenance tasks.This is demonstrated below.

```
rlls -r maint onsite
```

Any of the following maintenance states may be specified:

- checkout
- lost
- remove
- replace
- clean
- certify
- erase
- reinit
- idwait

- move
- accwait

A sample maintenance list is shown in Figure 55 .

Figure 55 The maintenance list

```
       Catalog Maintenance                        Sat May 15 13:52:16 1993

Vault: onsite

              vol_loc
              -------
      accwait:
              0003
              0007
              0008
      checkout:
              0004
      lost:
              0001
      clean:
              0006
```

ConvexTMR logs

ConvexTMR employs comprehensive logging to allow you to track system activity. ConvexTMR log files are located in the directory *LIBDIR*/REEL/logs, where *LIBDIR* is the RLLIBDIR value specified in the file /etc/reelenv.

Two logs are available to ConvexTMR users:

- user log
- job log

The user log: REEL/logs/u_*uname*

The user log records all user messages, including error messages and messages from the system operator. The date, time, system message number and message text are displayed for each message, in chronological order. To keep a constant, dynamically updated user log on display, issue the following command from a separate terminal session (substitute your RLLIBDIR value for *LIBDIR* and your own user ID for *uname*):

```
tail -f LIBDIR/REEL/logs/u_uname
```

A sample user log is shown in Figure 56 .

Figure 56 The user log

```
Jul  1 12:37:12> RL5239: Request 1027 obtained resources
Jul  1 12:37:41> RL5198: Device: 380: Allocate Scratch Tape 'test/tape1' (ANSI)
Jul  1 12:37:44> RL5241: Request 1027 Device /trial/stress ready
Jul  1 12:37:44> RL5242: Request 1027 return success
Jul  1 12:38:15> RL5198: Device: 380: Allocate Scratch Tape 'test1/tape2' (ANSI)
Jul  1 12:38:41> RL5198: Device: 380: Allocate Scratch Tape 'test2/tape3' (ANSI)
Jul  1 12:39:44> RL5198: Device: 380: Allocate Scratch Tape 'test4/tape5' (ANSI)
```

The job log: REEL/logs/J_*uname/key*

The job log records the same types of information as the user log, but for a specific job, as identified by a job key. Job logs are stored in the directory `logs/J_`*uname*, in files identified by specific job keys. For more information on job keys, refer to the section titled "Resource Key" in Chapter 2 of this manual.

Extended examples

A

Contents

This appendix includes extended user examples from Chapter 2. The extended examples allow you to study the ConvexTMR commands in the full context of reserving the device, accessing the volumeset, volumes, and files, and examining the output from the session. This will allow you to obtain a more comprehensive understanding of the user commands and the flexibility of ConvexTMR.

Device model: -d dmodel
(on page 16)

```
rlaccess -RW -d 3480 -V 000015,000016,000017 \
tlink
Device 1: /mnt/UG/examples/tlink
cp /etc/passwd tlink
mt -f tlink rew
diff tlink /etc/passwd
rlrelease tlink
```

Multiple devices: -N ndev
(on page 17)

```
rlaccess -RW -N 2 -V 000016 -V000015
Device 1: /dev/tape/R1059.D1
Device 2: /dev/tape/R1059.D2
cp /etc/group /dev/tape/R1059.D1
cp /etc/motd /dev/tape/R1059.D2
mt -f /dev/tape/R1059.D1 rew
```

```
diff /etc/group /dev/tape/R1059.D1
rlrelease /dev/tape/R1059.D1
mt -f /dev/tape/R1059.D2 rew
diff /etc/motd /dev/tape/R1059.D2
rlrelease /dev/tape/R1059.D2
```

Media disposition: -q
(on page 17)

```
rlaccess -RW -q -V 000015 tlink
Device 1: /mnt/UG/examples/tlink
ls -ls tlink tlink.r
2 lrwxrwxrwx  1 joeuser          22 Sep 22 \
13:31 tlink -> /dev/tape/upi/tc2norew
2 lrwxrwxrwx  1 joeuser          20 Sep 22 \
13:31 tlink.r -> /dev/tape/upi/tc2rew
cp /etc/reelenv tlink.r
cat tlink
   RLLIBDIR /guest
   RLBINDIR /usr/convex
   RLPBASE 654321000
   RL_MACH orion
   CLNTNAME orion
   RLLOGDIR /usr/adm/log/REEL
   ODdefault (orion:54545)
   ODsilo (orion:54555)
   RLSRVID joeuser
   #RLCATALOG set
cp /usr/lib/REEL/Librarian/vault_map tlink.r
cat tlink
   RLLIBDIR /guest
   RLBINDIR /usr/convex
   RLPBASE 654321000
   RL_MACH orion
```

```
   CLNTNAME orion
   RLLOGDIR /usr/adm/log/REEL
   ODdefault (orion:54545)
   ODsilo (orion:54555)
   RLSRVID joeuser
   #RLCATALOG set
cat tlink
   #
   # Storage Vaults
   #
   # fmt: <index> <vault>
   #
   1 onsite
   2 offsite
   3 silo
rlrelease tlink
```

Access Path file: -A filename
(on page 17)

```
rlaccess -RW -A AccessPathes -N 2 -V 000015 -V \
000017
ls -l AccessPathes
-rw-------  1 joeuser           60 Sep 22 \
13:33 AccessPathes
cat AccessPathes
Device 1: /dev/tape/R1061.D1
Device 2: /dev/tape/R1061.D2
ls -l /dev/tape/R1061.D1 /dev/tape/R1061.D2
lrwxrwxrwx  1 joeuser           22 Sep 22 \
13:33 /dev/tape/R1061.D1 ->
/dev/tape/upi/tc6norew
lrwxrwxrwx  1 joeuser           22 Sep 22 \
13:33 /dev/tape/R1061.D2 ->
/dev/tape/upi/tc2norew
```

Release all of the devices:

```
rlrelease
rm AccessPathes
```

Physical file number: -F(:eext_lbl):ppno (on page 18)

Create three files on two separate tapes. These files do not fill the tape, so one file spans a volume gap.

```
rlaccess -RW -V 000015 -V 000016 tlink
Device 1: /mnt/UG/examples/tlink
echo Hello Volume 000015, I'm File #1 > tlink
echo Hello Volume 000015, I'm File #2 >> tlink
echo Hello Volume 000015, I'm File #3 >> tlink
rlnext tlink
echo Hello Volume 000016, I'm File #1 > tlink
echo Hello Volume 000016, I'm File #2 >> tlink
echo Hello Volume 000016, I'm File #3 >> tlink
rlrelease
```

You cannot span volumes without using the :e modifier

```
rlaccess -RW -V 000015 -F:p2 -F:p3 -F:p1 tlink
Device 1: /mnt/UG/examples/tlink
```

You are positioned at the first file object: -F:p2

```
cat tlink
Hello Volume 000015, I'm File #2
```

Move to the next file object: -F:p3

```
rlnext tlink
cat tlink
Hello Volume 000015, I'm File #3
```

Move to the next file object: -F:p1

```
rlnext tlink
cat tlink
Hello Volume 000015, I'm File #1
```

```
rlrelease tlink
```

Now the :e modifier is used to move across noncontiguous volumes (the volumes are on different tapes):

```
rlaccess -RW -V 000015,000016 -F:e000015:p2 \
-F:e000016:p3 -F:e000015:p1 -F:e000016:p2 tlink
Device 1: /mnt/UG/examples/tlink
```

Positioned at the first file object: -F:e000015:p2

```
cat tlink
Hello Volume 000015, I'm File #2
```

Move to the next file object: -F:e000016:p3 (second volume)

```
rlnext tlink
cat tlink
Hello Volume 000016, I'm File #3
```

Move to the next file object: -F:e000015:p1 (first volume)

```
rlnext tlink
cat tlink
Hello Volume 000015, I'm File #1
```

Move to the next file object: -F:e000016:p2 (second volume)

```
rlnext tlink
cat tlink
Hello Volume 000016, I'm File #2
rlrelease tlink
```

File sequence number: -F:nfseq
(on page 19)

The file sequence number is a field in the ANSI/IBM header HDR1. Therefore, :nfseq is not useful for nonlabeled tapes.

```
rlaccess -RW -V 000015 tlink
Device 1: /mnt/UG/examples/tlink
echo Hello Volume 000015, I'm File #1 > tlink
echo Hello Volume 000015, I'm File #2 >> tlink
echo Hello Volume 000015, I'm File #3 >> tlink
rlrelease
```

```
rlaccess -RW -V 000015 -F:n0003 -F:n0002 \
-F:n0001 tlink
Device 1: /mnt/UG/examples/tlink
```

Positioned at the first file object: -F:n0003

```
cat tlink
Hello Volume 000015, I'm File #3
```

Move to the next file object: -F:n0002

```
rlnext tlink
cat tlink
Hello Volume 000015, I'm File #2
```

Move to the next file object: -F:n0001

```
rlnext tlink
cat tlink
Hello Volume 000015, I'm File #1
rlrelease tlink
```

File generation number: -F:ggen
(on page 19)

Create a new generation for each month. The generation number will match the month number.

```
rlaccess -RW -V 000015 -F:fAprilData:g4:A \
-F:fMayData:g5:A -F:fJuneData:g6:A tlink
Device 1: /mnt/UG/examples/tlink
cp /usr/adm/log/tapelog.4 tlink
rlnext tlink
cp /usr/adm/log/tapelog.5 tlink
rlnext tlink
cp /usr/adm/log/tapelog.6 tlink
rlrelease
```

Access the data by generation number:

```
rlaccess -RW -V 000015 -F:g6 -F:g5 -F:g4 tlink
Device 1: /mnt/UG/examples/tlink
diff tlink /usr/adm/log/tapelog.6
```

```
rlnext tlink

diff tlink /usr/adm/log/tapelog.5

rlnext tlink

diff tlink /usr/adm/log/tapelog.4

rlrelease tlink
```

File version number: -F:vgen
(on page 20)

Create a new generation version for each day of the month. The generation number matches the number of the month. The version number matches the day of the month.

```
rlaccess -RW -V 000015 -F:fApril1st:g4:v1:A \
-F:fApril2nd:g4:v2:A -F:fApril3rd:g4:v3:A tlink

Device 1: /mnt/UG/examples/tlink

cp /usr/adm/log/tapelog.4 tlink

rlnext tlink

cp /usr/adm/log/tapelog.5 tlink

rlnext tlink

cp /usr/adm/log/tapelog.6 tlink

rlrelease
```

Access the data by generation and version number:

```
rlaccess -RW -V 000015 -F:g4:v3 -F:g4:v2
-F:g4:v1 tlink

Device 1: /mnt/UG/examples/tlink

diff tlink /usr/adm/log/tapelog.6

rlnext tlink

diff tlink /usr/adm/log/tapelog.5

rlnext tlink

diff tlink /usr/adm/log/tapelog.4

rlrelease tlink
```

File identifier: -F:ffid
(on page 20)

Create a new generation for each month. The Generation number will match the month number. The Name of the file (the *fileid*) stored in the file header HDR1 will be the Month Name followed by "Data."

```
rlaccess -RW -V 000015 -F:fAprilData:g4:A \
-F:fMayData:g5:A -F:fJuneData:g6:A tlink
```

```
Device 1: /mnt/UG/examples/tlink
```

```
cp /usr/adm/log/tapelog.4 tlink
```

```
rlnext tlink
```

```
cp /usr/adm/log/tapelog.5 tlink
```

```
rlnext tlink
```

```
cp /usr/adm/log/tapelog.6 tlink
```

```
rlrelease
```

Access the data by generation number:

```
rlaccess -RW -V 000015 -F:fJuneData \
-F:fMayData -F:fAprilData tlink
```

```
Device 1: /mnt/UG/examples/tlink
```

```
diff tlink /usr/adm/log/tapelog.6
```

Move to the next file object: MayData

```
rlnext tlink
```

```
diff tlink /usr/adm/log/tapelog.5
```

Move to the next file object: AprilData

```
rlnext tlink
```

```
diff tlink /usr/adm/log/tapelog.4
```

```
rlrelease tlink
```

Append new file: -F:A
(on page 21)

Access a volumeset three separate times; each time appending a file to the end of the volumeset.

```
rlaccess -RW -V 000015 -F:fAprilData:g4:A tlink
```

```
Device 1: /mnt/UG/examples/tlink
```

```
cp /usr/adm/log/tapelog.4 tlink
rlrelease tlink
rlaccess -RW -V 000015 -F:fMayData:g5:A tlink
Device 1:
/mnt/eroberts/Projects/ConvexTMR.SPP/UG/example
s/tlink
cp /usr/adm/log/tapelog.5 tlink
rlrelease
```

Access the data by filename:

```
rlaccess -RW -V 000015 -F:fJuneData:g6:A tlink
Device 1: /mnt/UG/examples/tlink
cp /usr/adm/log/tapelog.6 tlink
rlrelease
rlaccess -RW -V 000015 -F:fJuneData \
-F:fMayData -F:fAprilData tlink
Device 1: /mnt/UG/examples/tlink
diff tlink /usr/adm/log/tapelog.6
rlnext tlink
diff tlink /usr/adm/log/tapelog.5
rlnext tlink
diff tlink /usr/adm/log/tapelog.4
rlrelease tlink
```

Overwrite file: -Fnew_file_spec:O:old_file_spec (on page 21)

Access a volumeset three separate times, each time appending a file to the end of the volumeset, then overwriting the one file.

```
rlaccess -RW -V 000015 -F:fAprilData:g4:A tlink
Device 1: /mnt/UG/examples/tlink
cp /usr/adm/log/tapelog.4 tlink
rlrelease tlink
rlaccess -RW -V 000015 -F:fMayData:g5:A tlink
Device 1: /mnt/UG/examples/tlink
```

```
cp /usr/adm/log/tapelog.5 tlink

rlrelease

rlaccess -RW -V 000015 -F:fJuneData:g6:A tlink

Device 1: /mnt/UG/examples/tlink

cp /usr/adm/log/tapelog.6 tlink

rlrelease
```

Access the data by filename:

```
rlaccess -RW -V 000015 -F:fJuneData \
-F:fMayData -F:fAprilData tlink

Device 1: /mnt/UG/examples/tlink

diff tlink /usr/adm/log/tapelog.6

rlnext tlink

diff tlink /usr/adm/log/tapelog.5

rlnext tlink

diff tlink /usr/adm/log/tapelog.4

rlrelease tlink
```

Now, overwrite JuneData:

```
rlaccess -RW -V 000015
-F:fJuneUPDATE:O:fJuneData:g6 tlink

Device 1: /mnt/UG/examples/tlink

cp /usr/adm/log/tapelog.1 tlink

rlrelease
```

The new file to be written is "JuneUPDATE." Because JuneData was the last file on the tape, the preceding data (AprilData and MayData) is still valid. If we had overwritten AprilData, then MayData and JuneData would have been lost.

```
rlaccess -RW -V 000015 -F:fJuneUPDATE \
-F:fMayData -F:fAprilData tlink

rlaccess: RL1112: Unknown FID

diff tlink /usr/adm/log/tapelog.1

diff: tlink: Mount device busy

rlnext tlink

rlnext: RL1175: Unknown device name

diff tlink /usr/adm/log/tapelog.5
```

```
diff: tlink: No such file or directory
rlnext tlink
rlnext: RL1175: Unknown device name
diff tlink /usr/adm/log/tapelog.4
diff: tlink: No such file or directory
rlrelease tlink
rlrelease: RL1175: Unknown device name
```


Object groups: -g
(on page 22)

Group a set of volume and file objects on a tape device. The volumesets in the first group are in VOLUME ACCESS MODE. The volumesets in the second group are in FILE ACCESS MODE.

```
rlaccess -RW -g -V 000015 -V 000016 -g -V \
000017 -F:fpasswd:A -F:fhosts:A -F:fgettytab:A
Device 1: /dev/tape/R1115.D1
Device 2: /dev/tape/R1115.D2
cp /etc/motd /dev/tape/R1115.D1
rlnext /dev/tape/R1115.D1
cp /etc/hosts /dev/tape/R1115.D2
rlnext /dev/tape/R1115.D2
cp /etc/group /dev/tape/R1115.D2
cp /etc/passwd /dev/tape/R1115.D1
rlrelease /dev/tape/R1115.D1
rlnext /dev/tape/R1115.D2
cp /etc/gettytab /dev/tape/R1115.D2
rlrelease /dev/tape/R1115.D2
```

Access each volume and verify files:

```
rlaccess -RW -V 000015 -V 000016 -V 000017 \
tlink
Device 1: /mnt/UG/examples/tlink
```

First file on volume 000015

```
diff /etc/motd tlink
```

Move to volume 000016:

```
rlnext tlink
```

First file on volume 000016

```
diff /etc/passwd tlink
```

Move to volume 000017:

```
rlnext tlink
```

First file on volume 000017

```
diff /etc/hosts tlink
```

Second file on volume 000017

```
diff /etc/group tlink
```

Third file on volume 000017

```
diff /etc/gettytab tlink
```

```
rlrelease tlink
```

Named object groups: -G
(on page 23)

Group a set of volume and file objects on a tape device The volumesets in the first group are in VOLUME ACCESS MODE. The volumesets on the second group are in FILE ACCESS MODE.

```
rlaccess -RW -G tlink1 -V 000015 -V 000016 \
-G tlink2 -V 000017 -F:fpasswd:A -F:fhosts:A \
-F:fgettytab:A
```

```
Device 1: /mnt/UG/examples/tlink1
```

```
Device 2: /mnt/UG/examples/tlink2
```

```
cp /etc/motd tlink1
```

```
rlnext tlink1
```

```
cp /etc/hosts tlink2
```

```
rlnext tlink2
```

```
cp /etc/group tlink2
```

```
cp /etc/passwd tlink1
```

```
rlrelease tlink1
```

```
rlnext tlink2
```

```
cp /etc/gettytab tlink2
```

```
rlrelease tlink2
```

Access each volume and verify files:

```
rlaccess -RW -V 000015 -V 000016 -V 000017 \
tlink
Device 1: /mnt/UG/examples/tlink
diff /etc/motd tlink
rlnext tlink
diff /etc/passwd tlink
rlnext tlink
diff /etc/hosts tlink
diff /etc/group tlink
diff /etc/gettytab tlink
rlrelease tlink
```


Expiration override: -j
(on page 23)

Create a set of files with expiration dates in the future:

```
rlaccess -RW -V 000015 -F:fpasswd:A -x:X1/1/99 \
-F:fhosts:A -x:X2/2/98 -F:fgettytab:A \
-x:R200 tlink
Device 1: /mnt/UG/examples/tlink
cp /etc/motd tlink
rlnext tlink
cp /etc/hosts tlink
rlnext tlink
cp /etc/group tlink
rlrelease tlink
```

Diff the files. This shows the files are readable with no problem.

```
rlaccess -RW -V 000015 -F:fpasswd -F:fhosts \
-F:fgettytab tlink
Device 1: /mnt/UG/examples/tlink
diff /etc/motd tlink
rlnext tlink
```

```
diff /etc/hosts tlink

rlnext tlink

diff /etc/group tlink

rlrelease tlink
```

Now try to write over the files, without the override, this will fail.

```
rlaccess -RW -V 000015 -F:fdisktab:O:fpasswd \
tlink

Device 1: /mnt//UG/examples/tlink

cp /etc/disktab tlink

cp: tlink: I/O error

rlstat -e -P tlink

Device Error: RL1113: File not expired

rlrelease tlink
```

Now try it with the -j (Override Expiration Date) option. Because we overwrote the first file, the next two files were lost even though they were not expired!

```
rlaccess -RW -V 000015 -F:fdisktab: \
O:fpasswd -j tlink

Device 1: /mnt/UG/examples/tlink

cp /etc/disktab tlink

rlrelease tlink

rlaccess -RW -V 000015 -F:fdisktab tlink

Device 1: /mnt/UG/examples/tlink

diff /etc/disktab tlink

rlrelease tlink
```

Object Name: -O
(on page 24)

Create a set of files with expiration dates in the future:

```
rlaccess -RW -V 000015 -F:p1:A -F:p2:A \
-F:p3:A tlink

Device 1: /mnt/UG/examples/tlink

cp /etc/motd tlink

rlnext tlink
```

```
cp /etc/hosts tlink

rlnext tlink

cp /etc/group tlink

rlrelease tlink
```

Diff the files. This shows the files are readable with no problem.

```
rlaccess -RW -V 000015 -F:p1 -O motd -F:p2 -O
\ hosts -F:p3 -O group tlink

Device 1: /mnt/UG/examples/tlink

rlnext -O hosts tlink

diff /etc/hosts tlink

rlnext -O group tlink

diff /etc/group tlink

rlnext -O motd tlink

diff /etc/motd tlink

rlrelease tlink
```


Unassigned object: -U
(on page 26)

Create a set of files. The last volume object is not assigned to a drive. This could be useful when the job doesn't know which drive will become available first. By leaving volume 000017 unassigned it can be mounted on either drive.

```
rlaccess -RW -g -V 000015 -g -V 000016 -U
-V000017

Device 1: /dev/tape/R1148.D1

Device 2: /dev/tape/R1148.D2
```

Volume 000015 is assigned to the first drive:

```
cp /etc/motd /dev/tape/R1148.D1
```

Volume 000016 is assigned to the second drive:

```
cp /etc/hosts /dev/tape/R1148.D2
```

Request volume 000017 on the first drive:

```
rlnext -U /dev/tape/R1148.D1

cp /etc/group /dev/tape/R1148.D1
```

The following rlrelease command releases both devices:

```
rlrelease
rlaccess -RW -V 000015 -V 000016 -V 000017 tlink
Device 1: /mnt/UG/examples/tlink
```

Volume 000015 is loaded first:

```
diff /etc/motd tlink
```

Request volume 000016:

```
rlnext tlink
diff /etc/hosts tlink
```

Request volume 000017:

```
rlnext tlink
diff /etc/group tlink
rlrelease tlink
```

Expiration: -x expdate
(on page 28)

Create a set of files with expiration dates in the future:

```
rlaccess -RW -V 000015 -F:fpasswd:A -x:X1/1/99 \
-F:fhosts:A -x:X2/2/98 -F:fgettytab:A \
-x:R200 tlink
Device 1: /mnt/UG/examples/tlink
cp /etc/motd tlink
rlnext tlink
cp /etc/hosts tlink
rlnext tlink
cp /etc/group tlink
rlrelease tlink
```

Diff the files. This shows the files are readable with no problem.

```
rlaccess -RW -V 000015 -F:fpasswd -F:fhosts \
-F:fgettytab tlink
Device 1: /mnt/UG/examples/tlink
diff /etc/motd tlink
rlnext tlink
diff /etc/hosts tlink
```

```
rlnext tlink

diff /etc/group tlink

rlrelease tlink
```

Now try to write over the files, without the over ride, this will fail.

```
rlaccess -RW -V 000015 \
-F:fdisktab:O:fpasswd tlink

Device 1: /mnt/UG/examples/tlink

cp /etc/disktab tlink

cp: tlink: I/O error

rlstat -e -P tlink

Device Error: RL1113: File not expired

rlrelease tlink
```

Now try it with the -j (Override Expiration Date) option. Because we overwrote the first file, the next two files were lost even though they were not expired.

```
rlaccess -RW -V 000015 -F:fdisktab \
:O:fpasswd -j tlink

Device 1: /mnt/UG/examples/tlink

cp /etc/disktab tlink

rlrelease tlink

rlaccess -RW -V 000015 -F:fdisktab tlink

Device 1: /mnt/UG/examples/tlink

diff /etc/disktab tlink

rlrelease tlink
```

Read labels file: -y
(on page 30)

Create a new generation for each month. The Generation number will match the month number. The Name of the file (the *fileid*) stored in the file header HDR1 will be the Month Name followed by "Data."

```
rlaccess -RW -V 000015 -F:fAprilData:g4:A \
-F:fMayData:g5:A -F:fJuneData:g6:A tlink

Device 1: /mnt/UG/examples/tlink

cp /usr/adm/log/tapelog.4 tlink
```

```
rlnext tlink
cp /usr/adm/log/tapelog.5 tlink
rlnext tlink
cp /usr/adm/log/tapelog.6 tlink
rlrelease
```

Access the data by generation number.

```
rlaccess -RW -y LabelData -V 000015 \
-F:fJuneData -F:fMayData -F:fAprilData tlink
Device 1: /mnt/UG/examples/tlink
```

Positioned at JuneData file:

```
diff tlink /usr/adm/log/tapelog.6
```

Move to next file object: MayData

```
rlnext tlink
diff tlink /usr/adm/log/tapelog.5
```

Move to next file object: AprilData

```
rlnext tlink
diff tlink /usr/adm/log/tapelog.4
rlrelease tlink
```

Show the data in the label file:

```
cat LabelData
VOL1000015                REEL
HDR1AprilData      00001500010001000400 94265000000 000000
HDR2U3276800000 0
EOF1AprilData      00001500010001000400 94265000000 000001
EOF2U3276800000 0
HDR1MayData        00001500010002000500 94265000000 000000
HDR2U3276800000 0
EOF1MayData        00001500010002000500 94265000000 000001
EOF2U3276800000 0
HDR1JuneData       00001500010003000600 94265000000 000000
HDR2U3276800000 0
HDR1JuneData       00001500010003000600 94265000000 000000
```

HDR2U3276800000 0

HDR1JuneData 0000150001000300600 94265000000 000000

HDR2U3276800000 0

EOF1JuneData 0000150001000300600 94265000000 000001

EOF2U3276800000 0

EOF1JuneData 0000150001000300600 94265000000 000001

EOF2U3276800000 0

HDR1MayData 0000150001000200500 94265000000 000000

HDR2U3276800000 0

HDR1MayData 0000150001000200500 94265000000 000000

HDR2U3276800000 0

EOF1MayData 0000150001000200500 94265000000 000001

EOF2U3276800000 0

VOL1000015 REEL

HDR1AprilData 0000150001000100400 94265000000 000000

HDR2U3276800000 0

VOL1000015 REEL

HDR1AprilData 0000150001000100400 94265000000 000000

HDR2U3276800000 0

EOF1AprilData 0000150001000100400 94265000000 000001

EOF2U3276800000 0

```
rm LabelData
```

Background execution: -I
(on page 30)

Create a new generation for each month. The Generation number will match the month number. The Name of the file (the *fileid*) stored in the file header HDR1 will be the Month Name followed by "Data."

```
rlaccess -I -RW -V 000015 tlink
```

```
Request ID: 1177
```

The user could go off and do other things while the tape is being mounted. If the user completes other work and wants to pend until the tape is mounted, he can use the rlstat command.

```
rlstat
User Request Summary

    User: joeuser
     Key:

     Reqnum  Prio  Status  User      Key       Dev
     ------  ----  ------  ----      ---       ---
     1177    1     POSNW   joeuser             d.3480

rlstat > foo
set reqnum = `awk -e '/^[ \
]*[0-9][0-9][0-9][0-9]/ {print $1}' foo`
rlstat -w -r 1177
Request ID: 1177
Device 1: /mnt/UG/examples/tlink
cp /usr/adm/log/tapelog.4 tlink
cp /usr/adm/log/tapelog.5 tlink
cp /usr/adm/log/tapelog.6 tlink
rlrelease
```

Verify the data:

```
rlaccess -RW -V 000015 tlink
Device 1: /mnt/UG/examples/tlink
diff tlink /usr/adm/log/tapelog.4
diff tlink /usr/adm/log/tapelog.5
diff tlink /usr/adm/log/tapelog.6
rlrelease tlink
rm foo
```

Resource key: -k reskey
(on page 31)

Access to separate devices under different resource keys. This allows mounts by the same uid to be tracked separately (log files). You can dismount groups of requests by resource key.

```
rlaccess -k TapeLg -RW -V 000015 \
-F:fAprilData:g4:A -F:fMayData:g5:A \
-F:fJuneData:g6:A tlink

Device 1: /mnt/UG/examples/tlink
                            \
rlaccess -k Config -RW -V 000016 \
-F:fpasswd:A -F:fgroup:A tlink2

Device 1: /mnt/UG/examples/tlink2

rlstat -R -a

Outstanding Resource Requests: Thu Sep 22 16:01:06 1994


    User: joeuser

     Key: ALL



Request: 1180:


      User: joeuser         Key: TapeLg       Twait: 01:57

      Prio: 1             Status: POSNW      Timeout: 360000


    Device: /mnt/UG/examples/tlink

      Opts: -d 3480  -f 3480:W  -v silo



Request: 1181:


      User: joeuser         Key: Config       Twait: 00:53

      Prio: 1             Status: POSNW      Timeout: 360000


    Device: /mnt/UG/examples/tlink2
```

```
                    Opts: -d 3480  -f 3480:W  -v silo
```

rldev

```
Current device status;

Device   Model  Status   Tape INTLBL EXTLBL       ReqID  Dn User     Key
------   -----  ------   ---- ------ ------       -----  -- ----     ---
tc7      3480   cfgdown  strt ------ ------
tc6      3480   orion    user 000015 000015       1180   1  joeuser  TapeLg
tc5      3480   cfgdown  strt ------ ------
tc4      3480   cfgdown  strt ------ ------
tc3      3480   cfgdown  strt ------ ------
tc2      3480   orion    user 000016 000016       1181   1  joeuser  Config
tc1      3480   cfgdown  strt ------ ------
tc0      3480   cfgdown  strt ------ ------
mt1      3420   cfgdown  strt ------ ------
mt0      3420   cfgdown  strt ------ ------
dat0     DAT    cfgdown  strt ------ ------
```

ls -l /usr/lib/REEL/logs/J_joeuser

```
total 34
-r--------  1 joeuser      751 Sep 21 19:24 Config
-r--------  1 joeuser      683 Sep 21 19:23 TapeLg
-r--------  1 joeuser    30131 Sep 20 10:20 nokey
```

rlrelease -k Config

rlrelease -k TapeLg

Glossary

A

ACL
Access Control List; A list of user and group IDs and access permissions associated with a ConvexTMR object. The ACL of a ConvexTMR object determines what kind of access specific users and groups have to the object. Viewed via `rllsacl`; controlled via `rlchacl`.

access permissions
A string of alphabetic characters associated with a user or group ID that controls access to a ConvexTMR object. An element of the access control list (ACL).

active state
The state of a volume that contains unexpired and wanted data.

administrator groups
A group of administrators assigned a specified authority level, name, and password. An administrator must be enrolled in a group to perform administrative tasks.

authority levels
A hierarchical division of powers; administrator groups are always assigned an authority level.

B

BLP
Bypass Label Processing; a processing mode that ignores the volume and file tape labels. Tape positioning and navigation occur as under standard label processing, but the labels are ignored. A ConvexTMR privilege.

C

catalog
A database that tracks all of the objects in the ConvexTMR library. Catalog records are maintained on ownership, contents, pools, vaulting, status, and many other details.

command line interface
The interface to the ConvexTMR mount request system that accepts ConvexTMR commands at the command line.

container
The container in which a volume is shipped and/or stored. Often included in the *vol_def*.

ConvexTMR objects
Objects that can be referenced or manipulated with the ConvexTMR software. These objects include files, volumes, volumesets, pools, and rotations.

D

database key
A system-generated string of characters that uniquely identifies a ConvexTMR file, volume, volumeset, pool, or rotation.

dataset
See tape file.

device
A tape drive. Specific devices are referenced by *dev_name*.

dev_name
A name for a specific tape device.

display panel
The eight-character LED display panel on IBM 3480/90 devices. ConvexTMR utilizes this panel to inform library operators of pending requests.

domain
A set of tasks and devices defined by the ConvexTMR administrator. ConvexTMR has a global domain that covers all requests. Multiple, custom domains are also supported.

domain server
A server that controls a specific operator domain.

E

EOV
End-of Volume; the physical end of a tape.

external label (*ext_lbl*)
The user-defined, unique string of from 4 to 12 characters that identifies the volume.The external label name is often recorded on a sticker on the outside of the volume.

ESL
Exceeding site resource limits for the number of devices that can be allocated simultaneously to one user or allocated under one resource key; a ConvexTMR privilege.

F

file
See tape file.

file access mode
Accessing tape data on a file basis only. When a tape is accessed in this manner, no other portion of the tape can be accessed.

file section
The segment of a tape file which resides on a single volume. If a tape file spans three tape volumes, then it has three file sections.

fileset
A set of associated, adjacent files residing on a volume or volumeset.

file_spec
A very specific method of file referencing recognized by the ConvexTMR catalog. For an expanded definition, see the `rlflc`(1) manpage.

fingerprint
A string of characters, calculated from label data and recorded in the ConvexTMR catalog, used to electronically identify volumes when they are mounted.

full-screen interface
An easy-to use alternative to the command-line interface, comprised of three windows: the request queue window, the device list window, and the message window.

G

generation
An instance of a file or volumeset. Generations allow multiple instances of a file or volumeset to exist at the same time and to be referenced independently. Newer instances are given higher generation numbers. A file with the name `payroll:G2:V1`, is the second instance, or generation, of the file payroll. The

number proceeded by "G" is the generation number; the number proceeded by "V" is the version number (see version).

H

header labels
Electronic labels at the beginning of volumes (volume header labels) and tape files (file header labels) that contain identification and other data.

hold state
A volume that is no longer part of a volumeset, that is reserved for future use.

I

implicit pools
A site configuration that creates private user pools for all library users. Indicated by the tag `dpool=IMPOOL` in the site constants report.

initialization
The process by which labels are written to ConvexTMR tape volumes; accomplished via `rlinit`.

internal label (*`int_lbl`*)
The string of characters recorded in the VOL1 label of all IBM and ANSI volumes; often referred to as the volume serial number (VSN). In many tape libraries, internal and external labels are identical. Maximum length for an internal label is 6 characters.

K

key
See database key.

L

labels
See header, trailer, user, internal, or external labels.

LIBDIR
The RLLIBDIR value specified in the file `reelenv`. `reelenv` resides in the directory specified for REELENV in your environment profile.

low water mark
A number that, if set, is the lowest number of scratch volumes ConvexTMR will allow in a pool without sending warning messages to the pool owner. This number is set via `rlpoolc` or `rlpoole`.

M

mask
An ACL entry that specifies the maximum permissions allowed to all users and groups. The mask entry overrides all specific user and group entries, but does not override the entry for other users.

N

NSL
Non-Standard Labels; labels other than IBM or ANSI standard labels. NSL tapes are processed as standard label tapes and accessed via BLP mode. NSL processing is a ConvexTMR privilege.

O

OBO
Initiating and controlling tape sessions "on behalf of" other users; a ConvexTMR privilege.

OSH
Off-line storage hierarchy; the ConvexTMR catalog directory structure. The OSH is not specific to a particular UNIX shell or process; do not confuse the OSH with your UNIX directory structure.

operator groups
A group of operators assigned a specified authority level, name, and password. An operator must be enrolled in a group to perform operator tasks.

P

pathname
A UNIX directory or file name. Examples: /usr, /etc/passwd, ./passwd.

permissions
See access permissions.

PDA
Accessing specific physical devices by name; a ConvexTMR privilege.

physical volume
A single volume, (see volume).

pools
A group of volumes. Tape pools partition the tape library into sub-libraries.

R

receipt number
A five-digit, random number proceeded by "R." Receipts are generated by the ConvexTMR software when volumes are submitted to or retrieved from the tape library. Receipt numbers can be used to reference volumes and volumesets.

request ID (*reqid*)
A five-digit, sequential number assigned to a `rlaccess` or `rlnext` request. Request IDs appear on the request monitor and can be used to reference specific ConvexTMR requests.

request monitor
The monitor through which the operator views and services ConvexTMR requests, messages, and device status; command-line and full-screen interfaces are available.

request queue
The list of all outstanding operator requests that displays on the request monitor.

resource key (*reskey*)
A user-defined alpha-numeric sequence of up to six characters that distinguishes a tape session from other sessions under the same user ID. Tape requests made under the same resource key affect the same tape session. Tape requests made by the same user ID under a different resource keys affect different sessions.

retension
To re-wrap tape on its storage spool to avoid loose spooling; a maintenance procedure to safeguard against tape breakage.

rewind
To position to the beginning of a tape.

role groups
See operator or administrator groups.

rotations
A list of vault locations and durations assigned to a volumeset. Rotations are ConvexTMR database objects.

S

scratch state
A volume that is not in current use, and on which any data has expired. Scratch volumes are available for assignment to a volumeset.

session
The period during which a user accesses a tape or tapes on one or more drives using the same resource key.

slot
The unique address of a volume in the storage vault. At some sites tapes are organized by external label rather than slot. Often included in the *vol_def*.

T

tape file
A set of related bits written to tape. Tape files can span multiple volumes within a volumeset.

tape mark
A delimiter used to indicate end of tape files.

trailer labels
Electronic labels at the end of volumes (volume trailer labels) and tape files (file trailer labels) that contain identification and other data.

U

user labels
Electronic labels reserved for user or application data; user labels follow the file header labels.

V

vault
The storage location for a group of volumes; ConvexTMR supports multiple vaults.

version
An instance of a generation of a file or volumeset; versions allow another level of object subscripting beyond generations. Newer versions are assigned higher numbers. A file with the name `payroll:G2:V1`, is the first instance, or version, of the second generation of the file `payroll`. The number proceeded by "V" is the version number; the number proceeded by "G" is the generation number (see generation).

vol_def
A method of volume referencing that may include any or all of the following values: internal label, external label, slot, container. *vol_def* is used to reference uncataloged volumes.

vol_spec
A very specific method of volume referencing recognized by the ConvexTMR catalog. For an expanded definition, see the `rlvsc`(1) manpage.

vs_spec
A very specific method of volumeset referencing recognized by the ConvexTMR catalog. For an expanded definition, see the `rlvsc`(1)
manpage

volume
A single tape.

volume access mode
Accessing an entire volumeset. When a volume is accessed in this manner, any and all files on the volumeset can be accessed.

volumeset
A logical volume consisting of one or more physical volumes.

Index

P

R

S

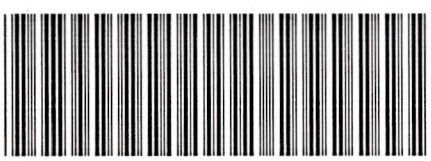
T

U

V

ORDER NUMBER
DSW-481

DOCUMENT NUMBER
710-029830-000



CONVEX PRESS