TMF User’s Guide

007–3969–006 Version 1.4

New Features in This Guide

Support for the Linux operating system has been added. Miscellaneous technical and editing changes were also made.

Record of Revision

Version	Description
1.0	December 1998 Original printing to support the Tape Management Facility (TMF) release 1.0, for SGI 64-bit systems running on the IRIX 6.4.1 or IRIX 6.5.2m operating system.
1.1	July 1999 Incorporates information in support of the TMF release 1.1 for the 64-bit SGI Challenge XL, SGI Challenge L, SGI Origin 200, and SGI Origin 2000 systems running the IRIX 6.4.1, IRIX 6.5.2m, or later operating system and for the 32-bit SGI Challenge S system running the IRIX 6.5.2m or later operating system.
003	November 1999 Incorporates information in support of the TMF release 1.2 for the 64-bit SGI Challenge XL, SGI Challenge L, SGI Origin 200, and SGI Origin 2000 systems running the IRIX 6.4.1, IRIX 6.5.2m, or later operating system and for the 32-bit SGI Challenge S system running the IRIX 6.5.2m or later operating system. Requires OpenVault release 1.4. The version entry on the Record of Revision page has been changed from the product revision number to the document revision number (the last three digits of the part number).
004	August 2000 Incorporates information in support of the TMF release 1.3 for the 64-bit SGI Challenge XL, SGI Challenge L, SGI Origin 200, and SGI Origin 2000 systems running the IRIX 6.5.2m or later operating system and for the 32-bit SGI Challenge S system running the IRIX 6.5.2m or later operating system. Requires OpenVault release 1.4.
005	April 2002 Supports TMF release 1.3.5.
006	March 2003 Supports TMF release 1.4.

Contents

Figures

Tables

Examples

Procedures

About This Guide

This user's guide documents the characteristics and capabilities of the Tape Management Facility (TMF). It explains the ways in which you can work with TMF and provides many examples of commonly used commands. It provides information on using tape formats, performing basic tape procedures, and writing C tape applications. It includes the following topics:

- Software overview and basic terminology
- Structure of tape formats
- Commands that access tapes by using TMF as well as the tape status and information commands
- Use of TMF from C programs
- TMF system messages
- TMF user man-page list

Related Publications

This guide is one of a set of manuals that describes TMF. The following manuals are also in the set:

- *TMF Administrator's Guide*
- *TMF Release and Installation Guide*

The following publication contains additional information that may be helpful:

- *Application Programmer's I/O Guide*

TMF Man Pages

In addition to printed and online documentation, several online man pages describe aspects of TMF. Each man page includes a general description of one or more commands, system calls, or other topics, and provides usage details (command syntax, parameters, and so on). Man pages exist for the user commands, devices

(special files), file formats, miscellaneous topics, and administration commands. Man page section identifiers appear in parentheses after man page names, as follows:

User commands	(1)
Devices (special files)	(4)
File formats	(5)
Miscellaneous topics	(7)
Administration commands	(8)

You can access these man pages by using the `man`(1) command as shown in the following example:

```
% man tmstat
```

You can print copies of online man pages by using the pipe symbol with the `man`(1), `col`(1), and `lpr`(1) commands. In the following example, these commands are used to print a copy of the `tmstat`(1) man page:

```
% man tmstat | col -b | lpr
```

Obtaining Publications

You can obtain SGI documentation in the following ways:

- See the SGI Technical Publications Library at: `http://docs.sgi.com`. Various formats are available. This library contains the most recent and most comprehensive set of online books, release notes, man pages, and other information.
- If it is installed on your SGI system, you can use InfoSearch, an online tool that provides a more limited set of online books, release notes, and man pages. With an IRIX system, select **Help** from the Toolchest, and then select **InfoSearch**. Or you can type `infosearch` on a command line.
- You can also view release notes by typing either `grelnotes` or `relnotes` on a command line.
- You can also view man pages by typing `man` *title* on a command line.

Conventions

The following conventions are used throughout this document:

Convention	Meaning
`command`	This fixed-space font denotes literal items such as commands, files, routines, path names, signals, messages, and programming language structures.
variable	Italic typeface denotes variable entries and words or concepts being defined.
`user input`	This bold, fixed-space font denotes literal items that the user enters in interactive sessions. (Output is shown in nonbold, fixed-space font.)
[ ]	Brackets enclose optional portions of a command or directive line.
...	Ellipses indicate that a preceding element can be repeated.

Reader Comments

If you have comments about the technical accuracy, content, or organization of this publication, contact SGI. Be sure to include the title and document number of the publication with your comments. (Online, the document number is located in the front matter of the publication. In printed publications, the document number is located at the bottom of each page.)

You can contact SGI in any of the following ways:

- Send e-mail to the following address:

  techpubs@sgi.com

- Use the Feedback option on the Technical Publications Library Web page:

  `http://docs.sgi.com`

- Contact your customer service representative and ask that an incident be filed in the SGI incident tracking system.

- Send mail to the following address:

  Technical Publications
  SGI
  1600 Amphitheatre Parkway, M/S 535
  Mountain View, California 94043–1351

- Send a fax to the attention of “Technical Publications” at +1 650 932 0801.

SGI values your comments and will respond to them promptly.

Software Overview

This chapter provides a brief software architecture overview, and covers the following topics:

- Architecture
- Tape label support
- Resource management
- Volume mounting and unmounting
- Tape positioning
- Front-end servicing
- User end-of-volume processing
- Multifile volume allocation
- Concatenated tape files
- Tape message log file
- Terminology

For details of specific releases, hardware platform support, and tape device and library support, see the *TMF Release and Installation Guide*.

1.1 Architecture

TMF is a subsystem that supports processing of ANSI and IBM labeled tape, including multifile volumes and multivolume sets. These capabilities are most important to customers who run production tape operations where tape label recognition and tape security are requirements.

The basic elements of TMF are the TMF daemon and TMF tape device driver. TMF provides operating personnel with a means to view and manage the tape resources configured within TMF. It also is the backbone for the operation of the Data Migration Facility (DMF), and for the operation of the `xfsdump`(1m) and `xfsrestore`(1m) commands.

TMF is started by the system operator or the system administrator, or it is started automatically as part of the system startup. It can communicate directly with the TMF driver and the SCSI tape device driver to process your requests as shown in Figure 1-1, page 3.

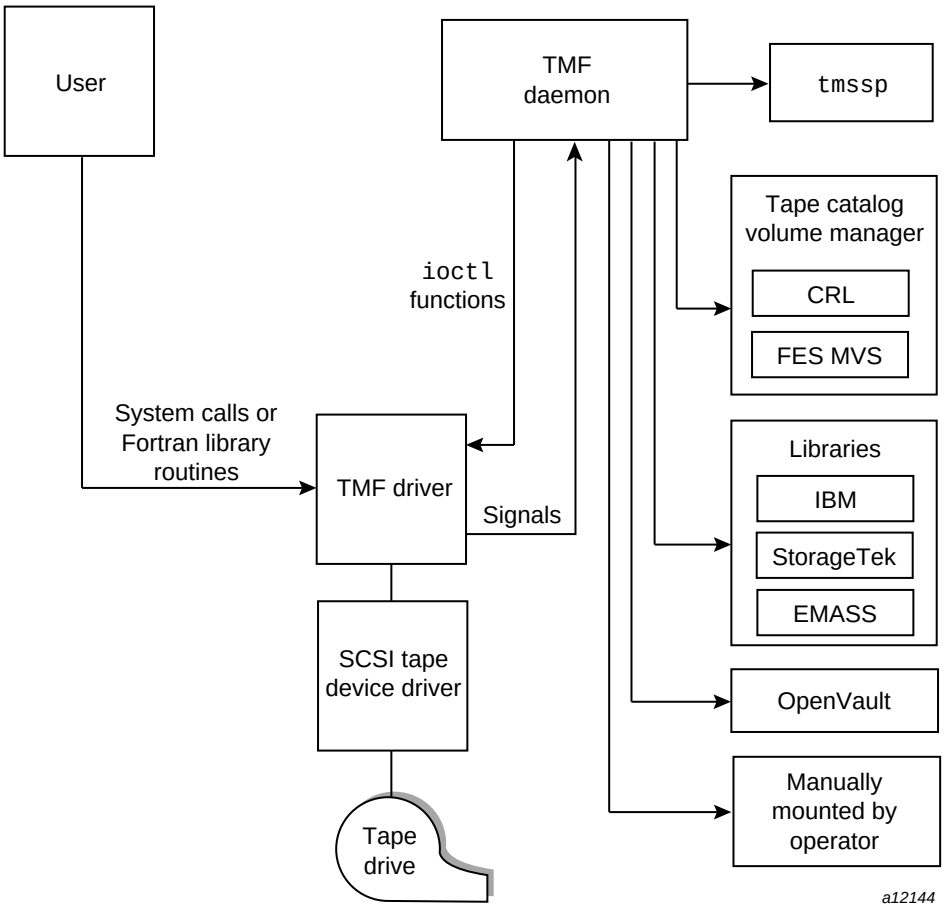

Figure 1-1 TMF Architecture

1.1.1 Communications

The TMF driver signals the TMF daemon if any of the following conditions occur:

- You issue an `open`(2) request to the tape path name.
- You issue a `close`(2) request to the tape path name.
- You issue the first I/O request to the tape path name.

- An I/O error occurs.
- A filemark is read.
- An end-of-tape is detected during a `write`(2) operation.
- An end-of-file is detected, requiring filemark processing.

If any of these conditions occur, your job is suspended until the TMF daemon finishes processing. At this point, the TMF daemon requests that the TMF driver either issue you an error message or allow you to continue. Figure 1-2 illustrates this process.

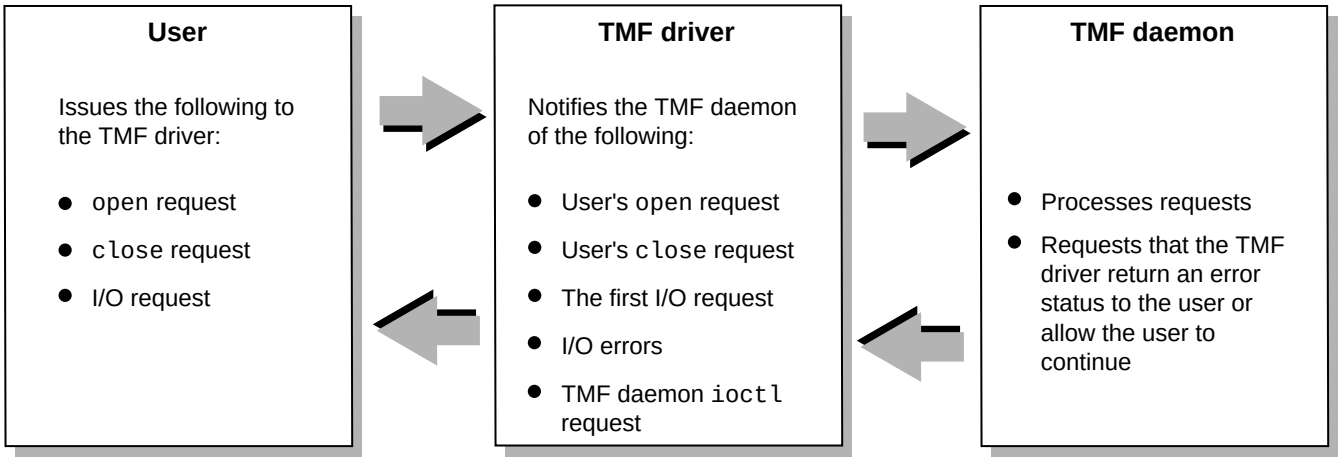

Figure 1-2 Communication between the User, TMF Driver, and TMF Daemon

Most vendors support only a character-special functionality, defined simply as the ability to open, to read from and write to, and to close a device that is recognized by the system software.

1.1.2 Functionality

SGI offers the following set of advanced functionality on all devices that TMF supports:

- Dynamic resource control
- Standard label support
- Nonlabeled tape support
- Bypass labeling
- Dynamic configuration control
- Multivolume and multifile support
- Embedded filemarks
- Distributed operator control
- Loader domains
- User end-of-volume processing
- Front-end servicing
- Absolute positioning
- OpenVault support
- Automatic volume recognition

When device vendors introduce new products, the standard marketing line is "this product is designed and operates within the boundaries of the *xxx* specification." While this may be true in the vendor engineering and test labs, when the device is introduced to the real world, reality sets in.

SGI has found that almost every device that vendors have produced can be made to fail when introduced into an environment with SGI systems configured. We have found that our software drives the devices to their limits. Most failures that have been discovered and fixed by the device vendor have led to a more stable and better performing product in the field.

While most enterprise vendors delegate device driver support to the peripheral vendors, SGI has accepted development and support as its role.

1.2 Tape Label Support

TMF supports ANSI standard labels, IBM standard labels, single filemark format tapes, and nonlabeled tapes. Single filemark format tapes do not have labels and are terminated by a single filemark at the end of volume, whereas a normal nonlabeled tape is terminated by two filemarks at the end-of-volume. Also, bypasslabel processing is available to tape administrators. Bypasslabel processing lets users read or write tape labels as regular files.

1.3 Resource Management

TMF monitors all of the tape resources configured within the system. It reads a TMF configuration file that contains a description of the tape configuration and then constructs a data-structure complex that contains information about each one of the tape drives. TMF enables system administrators to configure tape devices up or down. It also contains several commands to monitor its activities.

TMF allocates tape drives upon request, and ensures sure that such an allocation does not result in a deadlock condition. (A *deadlock* condition is one in which a task is locked in a state from which it cannot proceed.)

TMF creates and maintains wait queues for requests that cannot be satisfied at the time of the request. After a user has finished using a tape drive, that resource can be assigned to another user who has been queued in one of the wait queues.

1.4 Volume Mounting and Unmounting

TMF issues messages to either operating personnel in plain text or to a library in a data-structure format. These messages request the mounting of tapes on tape drives.

TMF supports several different families of libraries, including those from IBM, StorageTek, and EMASS. It also supports OpenVault, a storage library management facility. The automatic volume recognition (AVR) feature allows the operator to premount tapes prior to use and to direct the mounting of tapes to specific devices.

1.5 Tape Positioning

Tape positioning lets you position to the beginning of a tape block. Tape movement may be forward or backward; however, tape positioning directives cannot be used to

circumvent normal tape label processing or label checking unless you are `root` and use an absolute track address positioning request. You can position the tape file relative to a filemark, tape block, or volume; or you can position the tape file to an absolute track address.

1.6 Front-End Servicing

TMF provides a means of using a tape management system: front-end servicing for MVS (FES MVS available from SGI) that allows TMF messages and catalog requests to be processed by an IBM MVS system. Alternately, user exits let you use a local implementation for catalog services.

1.7 User End-of-Volume Processing

User end-of-volume (EOV) processing lets you gain control at the end of a tape volume. For EOV processing or positioning to a tape block, it is necessary to know that the file being processed is a tape file. You may request to be notified when end-of-volume is reached.

In addition, you can request special user EOV processing, which includes the reading, writing, and positioning of the volume before and after a volume switch. After special processing has completed, you must request that TMF resume normal processing.

1.8 Multifile Volume Allocation

Multifile volume allocation lets you process a multifile volume tape without the need for the system to unload and load tapes between files. A volume is a physical unit or storage medium, usually synonymous with a reel of magnetic tape.

1.9 Concatenated Tape Files

The concatenated tape file feature lets you read multiple tape files as though they were one tape file. An EOV status is returned for all of the concatenated files read, until the last file and its end-of-file is encountered.

1.10 Tape Message Log File

TMF maintains a log file in a user directory in which it records key events in its processing of requests on behalf of the user. This enables you to issue a batch job to process tape volumes and have a record of the activities that took place. Statistical data is recorded in this file as well.

1.11 Terminology

Table 1-1 describes TMF terminology that is used throughout this manual.

Table 1-1 TMF Terminology

Term	Definition
block size	The block size specifies the size (in bytes) of a data block on a tape.
device group	Each tape device belongs to a device group. The device group name is the generic device name in the TMF configuration file. Also referred to as a *resource*.
device name	Each tape device is identified by a device name, which is defined by a device name entry in the TMF configuration file.
device type	Each device has a device type, which is specified by a number. The different tape devices are available on operating systems.
external VSN	The external VSN is the human readable label applied to the tape's container. It is also called the *external volume identifier*.

Term	Definition
file identifier	The file identifier is the name of the file recorded in the `HDR1` label of a labeled tape. If specified in lowercase, the file identifier is converted to uppercase, per ANSI standard.
label type	The label type may be one of the following: nonlabeled, IBM standard, ANSI standard, or single filemark format.
path name	Each tape file is defined by a path name. You can specify the path name of the tape file by using the `tmmnt`(1) command. The system creates an entry in the directory specified by the path name. The tape device assigned to the tape file may change during volume switching. While a tape device is assigned to a tape file, you may not remove the path name of that tape file; the path name is removed when the tape device is released.
record length	The record length specifies the maximum length of a logical record (in bytes).
session identifier	The session identifier is the process identification number unique to the shell or batch job currently in use.
volume identifier	The volume identifier is a character string that consists of 1- to 6-alphanumeric characters identifying a tape. The volume identifier may also be referred to as the *volume serial number (VSN)*, the *internal VSN*, or the *internal volume identifier*.

Tape Formats

TMF supports both IBM compatible tape and ANSI formats. This chapter describes and illustrates these formats and the label fields if applicable.

2.1 IBM Compatible Tape Format

This section briefly describes and illustrates the IBM tape format. Tape format is determined by the presence or absence of labels and the number of files on a tape volume or number of volumes for a tape file.

System labels and filemarks are accessible to a user process only through the use of the `tmmnt`(1) command.

In the following figures, the character b represents the beginning of the tape and the character * represents a filemark (HDR2, EOV2, and EOF2 labels are optional for input). TMF always creates these labels for labeled tapes; other systems may not.

2.1.1 Nonlabeled Tapes

Nonlabeled tapes are of two formats, determined by the number of filemarks that indicate end-of-volume.

2.1.1.1 Two Filemark Tapes

Nonlabeled tapes with two filemarks, implemented by the `-l nl` option of the `tmmnt`(1) command, may consist of a single-volume file; a multivolume file; or multifile, multivolume file formats. Figure 2-1, page 12, illustrates these formats. For tapes with multiple files, a single filemark separates files on the same volume. End-of-volume is reached when two consecutive filemarks are encountered, and there is another tape to read.

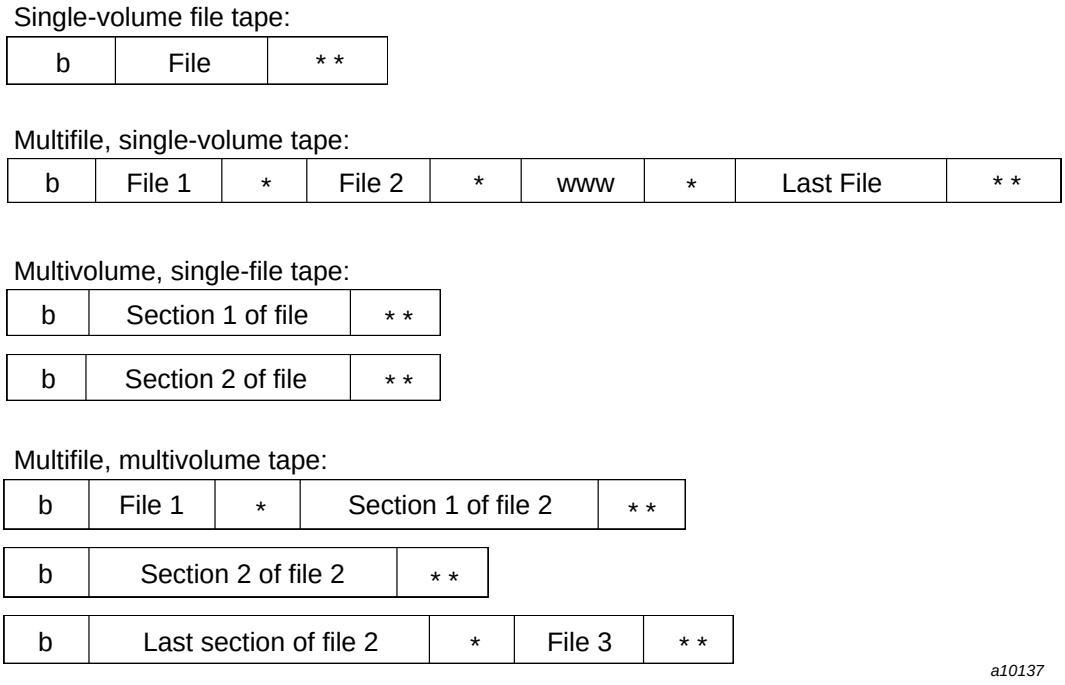

Figure 2-1 Nonlabeled, Two Filemark Formats

2.1.1.2 Single Filemark Tapes

For nonlabeled, single filemark format, implemented by the `-l st` option of the `tmmnt`(1) command, a single filemark indicates end-of-volume. When using one filemark tape as an input tape, the system reads only to the first filemark encountered.

When using a single filemark tape as an output tape, the system terminates the tape with three filemarks, allowing it to be read as a nonlabeled tape later on. Note that because the system processes only the data blocks and the first filemark, you cannot have multifiles on a single filemark tape. That is, you cannot use the `-l st` option with the `-q` option of the `tmmnt`(1) command.

Figure 2-2 illustrates nonlabeled, single filemark formats.

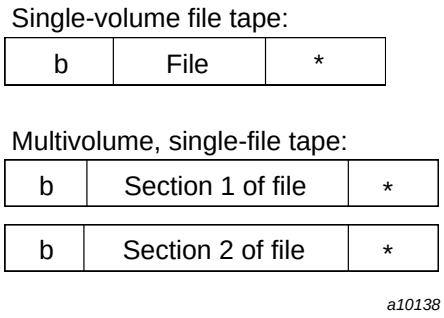

Figure 2-2 Nonlabeled, Single Filemark Formats

2.1.2 Labeled Tapes

Labeled tapes are implemented by the `-l al` (ANSI standard label) and the `-l sl` (IBM standard label) options of the `tmmnt`(1) command. ANSI standard labels and IBM standard labels are similar, with the exception that in IBM standard labels, the character fields are represented by EBCDIC characters while in ANSI standard labels, the character fields use ASCII characters.

Labeled tapes use the following labels in TMF (see Section 2.2, page 16, for a description of these labels):

- Volume header label (VOL1)
- First file header (HDR1)
- First end-of-volume (EOV1)
- First end-of-file (EOF1)
- Second file header (HDR2)
- Second end-of-volume (EOV2)
- Second end-of-file (EOF2)

Figure 2-3 illustrates labeled tape formats.

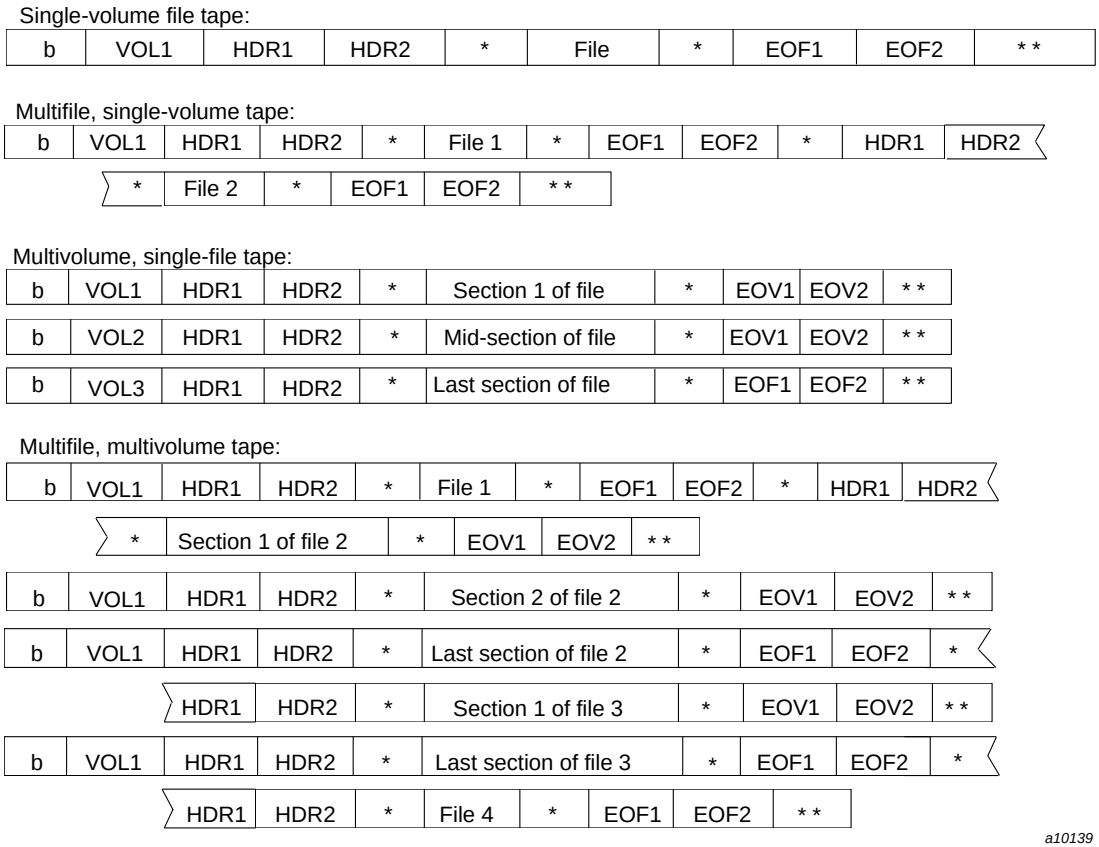

Figure 2-3 Labeled Tape Formats

2.1.3 IBM Compatible Tape Format Summary

The formats described previously are illustrated in Figure 2-4, page 15, through Figure 2-7, page 16, grouped by number of files and number of volumes. Figure 2-4 shows a single-volume file; Figure 2-5 shows a multifile, single-volume tape; Figure 2-6 shows a multivolume, single-file tape; and Figure 2-7 shows a multifile, multivolume tape. For each format type, the figures show both labeled (ANSI or IBM) and nonlabeled tapes.

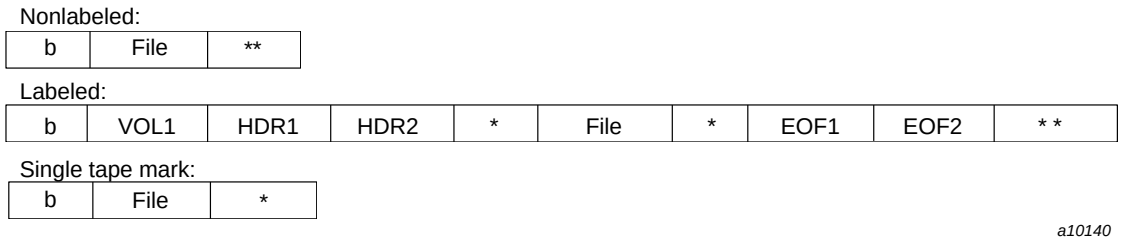

Figure 2-4 Single-Volume File

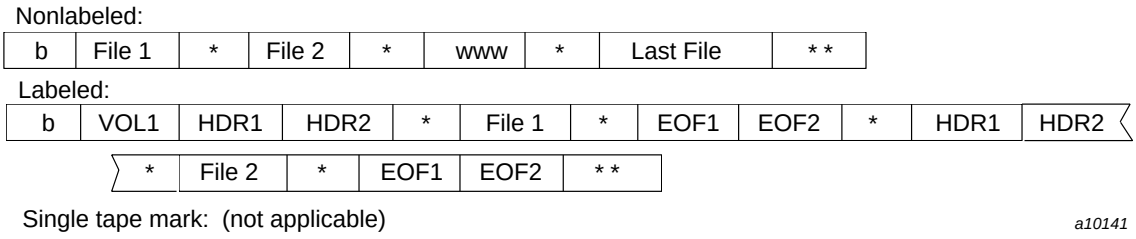

Figure 2-5 Multifile, Single-Volume Tape

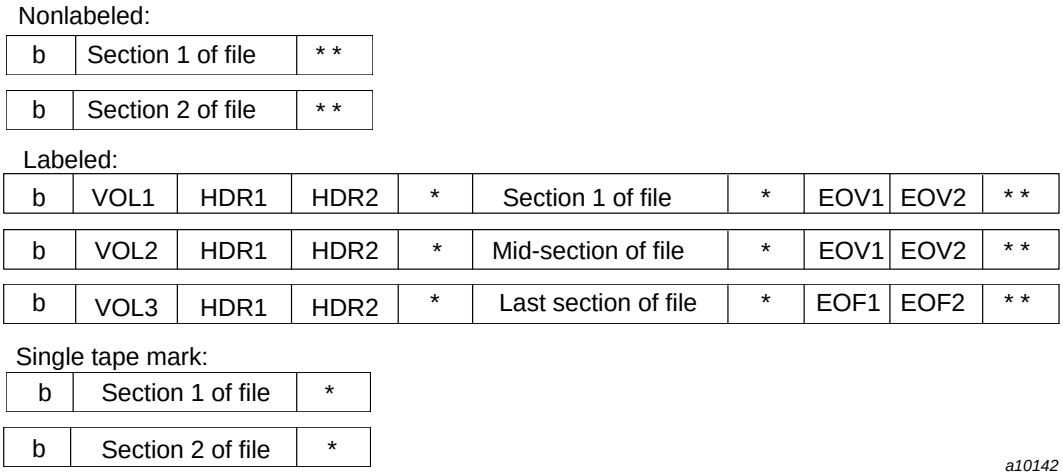

Figure 2-6 Multivolume, Single-File Tape

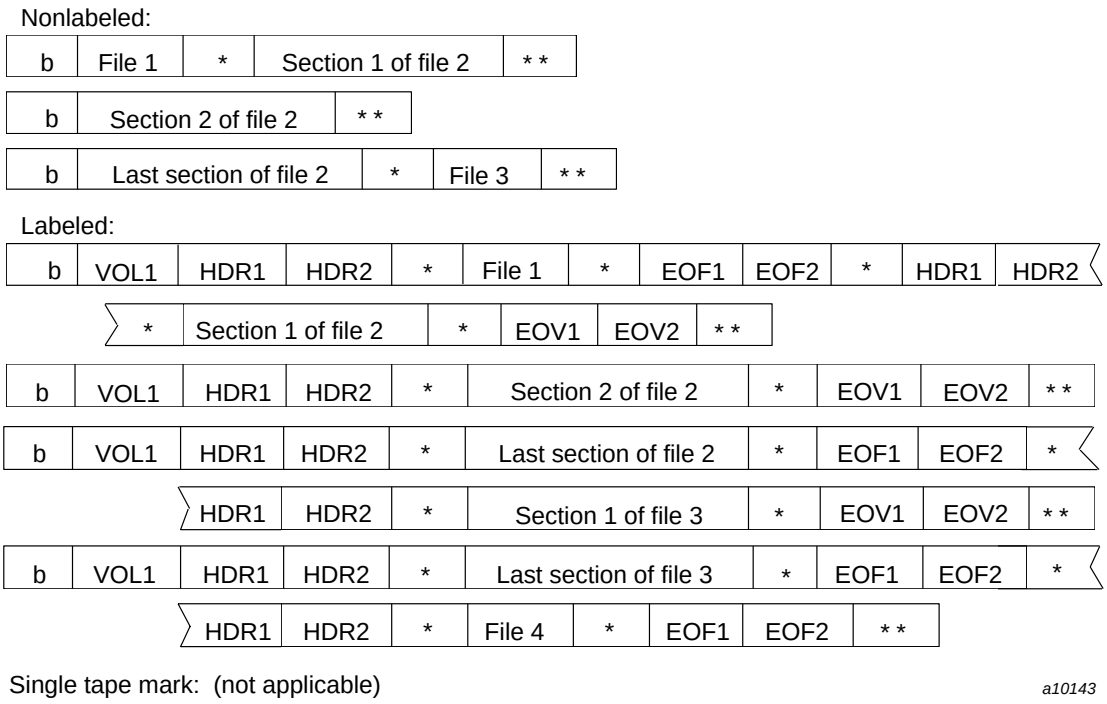

Figure 2-7 Multifile, Multivolume Tape

2.2 Tape Label Fields

This section describes the various tape label fields for ANSI standard and IBM standard labels. Specifically, it describes the fields in which label types are supported. These are checked by the system when reading or writing a tape and those that are filled in with parameter values when you use the `tmmnt`(1) command to create a labeled tape.

In IBM standard-label character fields, the characters are represented by EBCDIC characters. ANSI standard labels use ASCII characters.

2.2.1 VOL1 Label

The VOL1 label is the first block on a labeled tape. Table 2-1 describes the fields for an ANSI standard label. Figure 2-8, page 18, shows the format of the VOL1 label.

Table 2-1 VOL1 Label Values

Field	Starting byte	Length in bytes	Contents	Description
`label id`	1	4	VOL1	VOL1 label; required system-supplied character string.
`volume id`	5	6	*vi*	Volume identifier of the tape; it is specified with the `-v` option or contained in the file specified with the `-V` option of the `tmmnt`(1) command. It is checked on all labeled tapes and contains up to 6 alphanumeric characters.
`owner id`	38	14	*owner_id*	User ID of the tape owner.
`standard level`	80	1	*label standard version*	ANSI standard version number for label and data formats. For SGI systems, the version number is 4.

The fields of the ANSI standard VOL1 label are the same as the IBM standard VOL1 label, with the following exceptions:

- The `owner id` field of the IBM standard VOL1 label starts at byte 42 and has a length of 10 bytes.
- The standard level field is not used in the IBM standard VOL1 label.

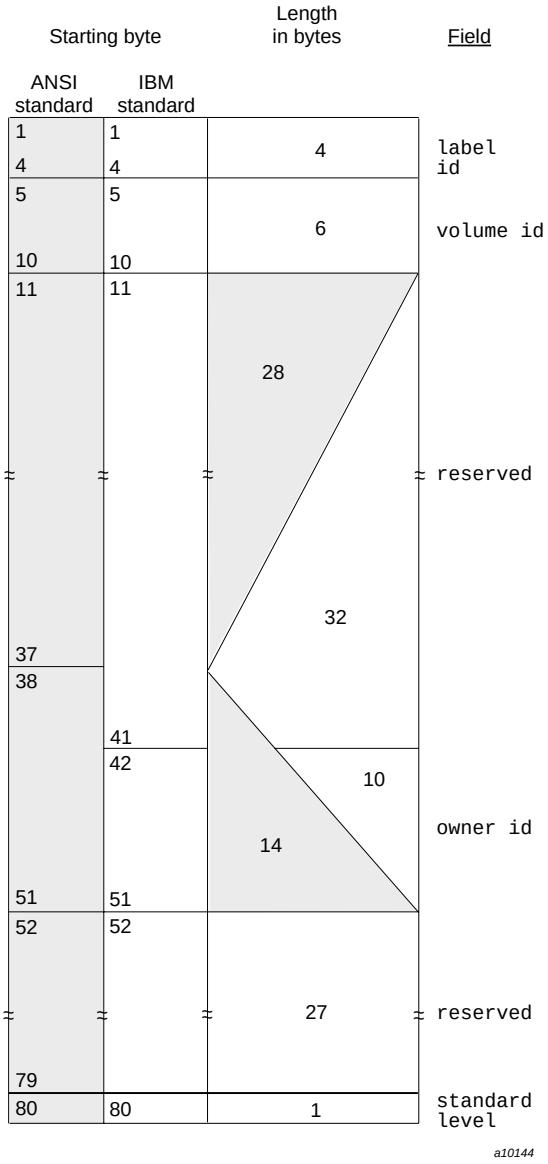

Figure 2-8 VOL1 Label

2.2.2 HDR1, EOV1, and EOF1 Labels

The HDR1 label is located before each file or section of a file on a tape volume. If a file is not completed on a tape volume and extends to the following tape volume, the data in the file is followed by an EOV1 label. If a file or file section is completed on a tape volume, the data in the file is followed by an EOF1 label.

The fields of the HDR1, EOV1, and EOF1 labels are the same in both the ANSI standard label format and the IBM standard label format. Table 2-2 describes the specified fields. Figure 2-9, page 21, shows the format of the HDR1/EOV1/EOF1 labels.

Table 2-2 HDR1/EOV1/EOF1 Labels

Field	Starting byte	Length in bytes	Contents	Description
`label id`	1	4	HDR1 EOV1 EOF1	Label type; required system-supplied character string.
`file id`	5	17	*file_id*	File identifier; 1- through 17-alphanumeric character field specified by the `-f` option of the `tmmnt`(1) command. If the `-f` option is not specified, the file identifier is taken from the path name of the `-p` or `-P` option of `tmmnt`. The level of checking on `file id` is installation specified.
`sequence`	28	4	*number*	Order of this volume in a multivolume set; it is specified by a decimal number (1 through 9999) on the `-O` option of `tmmnt`.
`file sequence`	32	4	*number*	File order within a multifile tape; it is specified by a decimal number (1 through 9999) on the `-q` option of `tmmnt`. The system uses the specified value to position the tape volume to the proper file.
`creation date`	42	6	*cyyddd*	Creation date (pseudo-Julian format) of a new tape; *c* = century (blank = 19, 0 = 20, 1 = 21...), *yy* = year (00-99), and *ddd* = day (001-366).

Field	Starting byte	Length in bytes	Contents	Description
`expiration date`	48	6	*cyyddd*	Expiration date (pseudo-Julian format) at which time a tape may be scratched or overwritten. Normally, it is specified in the *cyyddd* format by using the `-x` option of `tmmnt`. Otherwise, you can specify the number of days on the `-t` option by using `tmmnt`(1). The specified number is added to the `creation date`, thus creating the `expiration date`.
`block count`	55	6	*number*	Number of data blocks in the preceding file section or file on the current tape volume for EOV1 and EOF1 labels. The block count in the HDR1 label contains a value of 000000. In EOV and EOF labels for standard labels (`sl`), if the block count is greater than 999,999, the `block count` field will represent the block count as mod 1,000,000. The overflow (`block count` / 1000000) will be stored in bytes 76 through 80. This is the `extended block count` field. For ANSI standard labels (`al`), if the block count is greater than 999,999, the `block count` field will represent the block count as mod 1,000,000.
`extended block count`	76	5	*number*	For standard labels (`sl`), if the block count is greater than 999,999, the `block count` field will represent the block count as mod 1,000,000. The `extended block count` field will contain the overflow (`block count` / 1000000).

Starting byte	Length in bytes	Field
1 4	4	label id
5 21	17	file id
22 27	6	reserved
28 31	4	sequence
32 35	4	file sequence
36 41	6	reserved
42 47	6	creation date
48 53	6	expiration date
54	1	reserved
55 60	6	block count
61 ≈ 75	≈ 14 ≈	reserved
76 80	6	extended block count

a10145

Figure 2-9 HDR1/EOV1/EOF1 Labels

2.2.3 HDR2, EOV2, and EOF2 Labels

An HDR2 label immediately follows an HDR1 label, and it is followed by a filemark. An EOV2 label immediately follows an EOV1 label, and it is followed by two filemarks. An EOF2 label immediately follows an EOF1 label, and if more files follow this file, it is followed by one filemark. If the EOF2 label is the last file on the tape volume, it is followed by two filemarks.

ANSI standard does not specify a format for these labels, except for the first 4 bytes. IBM standard labels use the HDR2, EOV2, and EOF2 labels to store additional information concerning the file they bracket. TMF automatically writes these labels when you use the `-l sl` or `-l al` options of `tmmnt`(1). These labels follow the IBM standard format. Table 2-3 describes the specified fields. Figure 2-10, page 24, shows the format of the HDR2/EOV2/EOF2 labels.

Table 2-3 HDR2/EOV2/EOF2 Labels

Field	Starting byte	Length in bytes	Contents	Description
`label id`	1	4	HDR1 EOV1 EOF1	Label type; required system-supplied character string.
`record format`	5	1	*format*	Record format; 1-character filed specified by the `-F` option of `tmmnt`(1).
`block length`	6	5	*number*	Maximum block length, in bytes, for the associated file; specified by a decimal number (1 through 99999) on the `-b` option of `tmmnt`. If the block length is greater than 100000, the block length field will represent the block length as mod 100000.

Field	Starting byte	Length in bytes	Contents	Description
`record length`	11	5	*number*	Record length, in bytes; specified by the `-L` option of `tmmnt`.
`density`	16	1	*number*	Tape density; specified by the `-d` option of `tmmnt`. TMF supports 1600 bpi (this field contains the value 3) and 6250 bpi (this field contains the value 4).

Starting byte	Length in bytes	Field
1 4	4	label id
5	1	record format
6 10	5	block length
11 15	5	record length
16	1	density
17 ≈ 54	≈ 38 ≈	reserved
55 57	3	security level
58	1	reserved
59 74	17	compartments
75 80	4	reserved

a10146

Figure 2-10 HDR2/EOV2/EOF2 Labels

TMF Tutorial

This tutorial is designed to give an overview of various TMF capabilities. It introduces the following aspects of TMF usage and provides examples:

- Using tapes

  Using tapes generally consists of four parts. You reserve a tape device, mount a tape, manipulate the tape in some way, and lastly release the tape device.

  Additional detail is provided through basic usage examples, a description of how to specify block size with the `tmmnt` command, and an overview of OpenVault usage.

- Obtaining tape status and information

  You can obtain status information about your tape usage from various commands and files. You can also send messages to the operator.

- Using standard commands

  You can manipulate tapes with standard commands like `cp`(1), `dd`(1m), `tar`(1), `cpio`(1), and `tmmnt`(1).

Note: Details concerning the commands and parameters used in this chapter are described in the individual man pages. To see an online description of a particular command or routine, use the `man`(1) command.

3.1 Using Tapes

Procedure 3-1, page 26 provides an overview of the steps involved in using TMF. The examples in the following sections illustrate these steps.

Procedure 3-1 Basic TMF Tape Usage

To use a TMF tape, you follow four basic steps:

1. Reserve tape resources by using the `tmrsv`(1) command.

   The `tmrsv`(1) command allows you to reserve a device for your specific need. Reserving a device does not guarantee you immediate access to the drive; it simply alerts TMF that you will eventually request a tape.

2. Request a tape mount for a tape file by using the `tmmnt`(1) command.

   The `tmmnt`(1) command allows you to specify the various aspects of the tape, such as the VSN (volume serial number), FID (file identifier), and block size.

3. Process the tape file information by using whatever commands or programs you need to accomplish what you want to do.

   Once the tape is physically mounted on the drive, you can then manipulate it by referring to the path (`-p`) parameter that was used in the `tmmnt`(1) command. There are many ways to manipulate the tape including the `cp`(1), `dd`(1m), `tar`(1), and `cpio`(1) commands as well as through a C language program.

   Section 3.2, page 29, contains examples showing various ways to manipulate tapes. Chapter 4, page 37, describes the ways in which you may work with tapes from within C programs.

4. Release the tape resources reserved by using the `tmrls`(1) command.

   After you are done using the tape, you should use the `tmrls`(1) command to physically disassociate the tape from the tape drive. You can include parameters on the `tmrls`(1) command that allow you to continue using the tape drive with another tape or to release the tape drive from your control.

Note: Under TMF, your tape stays mounted unless you specifically issue a `tmrls`(1) command. If you neglect to release a tape explicitly, a user with `root` permission must issue a `tmfrls`(8) command to unload the tape.

3.1.1 Basic Usage Examples

The following series of examples illustrates the four steps. Example 3-1, page 27 shows how you request that a tape to be mounted. Example 3-2, page 27, shows how you mount an exiting tape and read it. Example 3-3, page 27, shows how you add a new tape file. For additional information about the commands and descriptions of the parameters, see the individual man pages.

Example 3-1 Creating a Tape

This tape has an IBM standard label with a volume serial number (VSN) of `000001` and a file name of `tf`. The contents of the disk file named `data` is copied to `tf`. After the tape file is created, the tape is unloaded and the allocated tape drive is released.

```
$ tmrsv CART
$ tmmnt -l sl -v 000001 -P tf -n -g CART
$ cp data tf
$ tmrls -a
```

Example 3-2 Reading an Existing Tape File

This example shows how you mount the previous tape, indicating that it is an old tape, and read the data from the tape into a disk file named `old.data`.

```
$ tmrsv CART
$ tmmnt -l sl -v 000001 -P tf -o -g CART
$ cp tf old.data
$ tmrls -a
```

Example 3-3 Adding a New File to an Existing Tape

This example shows how to add a new tape file (file sequence 2) to the tape `000001` and then copy the disk file named `new.data` into the new tape file.

```
$ tmrsv CART
$ tmmnt -l sl -v 000001 -q 2 -P tf2 -n -g CART
$ cp new.data tf2
$ tmrls -a
```

3.1.2 Specifying Block Size with `tmmnt`

Under TMF, you must specify the same block size on a command that you do on the `tmmnt`(1) command. Thus, if you enter a `tmmnt`(1) command with the `-b` option set

to 32768 and, then, read or write the tape with a `dd`(1m) command, the `bs` parameter must be set to 32768. Since a block size cannot be specified on a `cat`(1) or `cp`(1) command, you cannot use either command in this situation.

When you dump files to tape using the `-d` option of `xfsdump`(1m), it uses 262144 as the block size. As a result, you must issue the `tmmnt`(1) command with the `-v` option set to the number of volumes needed and the `-b` option set to 262144, which is 2^18.

Note: The administration command, `xfsdump`(1m), is documented in the *TMF Administrator's Guide*.

3.1.3 OpenVault Usage

When TMF requests OpenVault to mount or unmount a cartridge (physical tape), TMF uses the OpenVault *physical cartridge label* (PCL) as the external volume identifier for the `-v` option in the `tmmnt`(1) command.

For OpenVault users, a volume is an application's view of a partition on a cartridge. A cartridge has at most one volume on a partition. Each volume has a *volume name*.

If you omit the external volume identifier in the `-v` option of a `tmmnt`(1) command, TMF assumes the external volume identifier (the OpenVault PCL) is the same as the internal volume identifier (OpenVault volume name). Figure 3-1, page 28 shows this terminology.

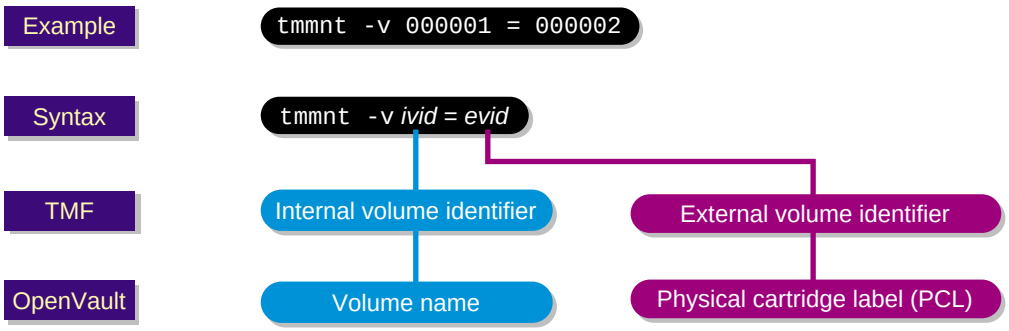

Figure 3-1 OpenVault `tmmnt` Terminology

Example 3-4 uses the same tmmnt(1) command as used in Example 3-2; however, this example is for a specific OpenVault environment instead of general tape usage.

Example 3-4 Reading an Existing OpenVault Tape File

In this tmmnt(1) command, CART is a drive group of OpenVault drives and a device group in TMF, and 000001 is the OpenVault volume name that TMF uses as the internal volume identifier in the -v option:

```
$tmrsv CART
$tmmnt -l -v 000001 -P tf -o -g CART
```

TMF is assuming -v 000001 = 000001; that is, the external volume identifier is the same as the internal volume identifier (000001). TMF then sends OpenVault a message that requests OpenVault to mount the PCL called 000001.

3.2 Obtaining Tape Status

You can use the commands and files described in this section to obtain tape status and to send messages to the operator.

3.2.1 Tape Status Commands

Users can check reserve status of tape devices by using the tmrst(1) command. For example, enter the tmrst(1) command to display the reserved-tape status resource name, number of reserved devices, number of used devices, and number of devices available for use as shown in Example 3-5.

Example 3-5 tmrst Status Display

One CART and two DLT devices have been reserved, and one DLT device is currently in use:

```
$ tmrst
Resource w  rsv  used avail
CART          1    0    1
DLT           2    1    1
SD3           0    0    0
3590          0    0    0
```

Note: The information display in the `Resource` column is determined by the system administrator in the TMF configuration file. You can use the `tmstat`(1) command to display available tape resources. The resource name is displayed in the `group` column. The system administrator specifies the name of each tape resource in the TMF configuration file.

The `tmstat`(1) command allows users to check the status of various tape devices. You can display the user identifier, session identifier of the user, device group name, device status, device name, stream ordinal, VSN, and other values by using the `tmstat`(1) command as shown in Example 3-6.

Example 3-6 `tmstat` Status Display

Two `STK9490` devices, `s9490s1` and `s9490s3`, and two `CART` devices, `tape100` and `tape200`, are idle and available for use. One `STK9490` device, `s9490s2`, is assigned to user `edh`, session identifier`402`, with a volume identifier of `004014`. It is on tape block `4`.

One `CART` device, `cart200`, is in a `cnfp` (configuration pending) state, which generally means that the drive is waiting on a response from a front-end controlled media loader before it can be configured completely up. All other devices are down and unavailable.

```
tmstat

user     sess   group   a stat device   stm rl ivsn   evsn   blks  NQSid
               STK9490 - idle s9490s1
edh      402    STK9490 - assn s9490s2    2 o  004014 004014    4
               STK9490 - idle s9490s3
               STK9490 - down s9490s4
               STK9490 - down s9490s5
               STK9490 - down s9490s6
               CART    - down cart100
               CART    - cnfp cart200
               CART    + idle tape100
               CART    + idle tape200
```

Note: The resource names are displayed in the `group` column. The system administrator specifies the name of each tape resource in the TMF configuration file.

3.2.2 Tape Log File

When you issue a `tmrsv`(1) command, a log file called `tape.msg` is created in your current working directory. This log file keeps track of messages TMF issues concerning your tape job. Example 3-7 is a tape message log file.

Example 3-7 `tape.msg` file

All informative and error messages are appended to this file.

```
Sep 23 12:48:41.733795 TM000 - Tape resource(s) have been reserved for you
Sep 23 12:48:48.170742 TM048 - Tape stream '/ptmp/ptc/tmp/x' assigned to device
dlt2
Sep 23 12:48:48.339957 TM046 - Mount volume DLT100(sl) ring-in, on device dlt2 for ptc (23896)
(null) or reply cancel / device name
Sep 23 12:50:43.564636 TM049 - /ptmp/ptc/tmp/x : DLT100(sl) : open : blocks = 0
Sep 23 12:50:52.514214 TM049 - /ptmp/ptc/tmp/x : DLT100(sl) : bov : write : blocks = 0
Sep 23 12:50:52.515612 TM049 - /ptmp/ptc/tmp/x : DLT100(sl) : bof : write : blocks = 0
Sep 23 12:51:01.547197 TM049 - /ptmp/ptc/tmp/x : DLT100(sl) : eot : write : blocks = 449
Sep 23 12:51:01.572432 TM049 - /ptmp/ptc/tmp/x : DLT100(sl) : close : blocks = 449
Sep 23 12:51:13.325245 TM050 - Tape device dlt2 has been released
Sep 23 12:51:13.328480 TM029 - All tape resources have been released
```

3.2.3 Messages to Operator

The `msgr`(1) command lets you send messages to the operator. For example, the following command line sends an informative message to the operator:

```
msgr  "Is tape ABC on the system?" &
```

You should always put messages to the operator in the background as a command will wait for an operator response before completing.

3.3 Using Standard Commands

This section describes some of the ways in which you may work with TMF by using standard commands.

Before you can issue commands to TMF for tape file processing, you must reserve the required number of tape drives for each device type needed. After you have reserved the tape drives, you may specify the tape volume in which the files to be processed

are located. After you have the volumes mounted, you can begin processing the tape files. When processing is complete, release the reserved tape drives.

The procedures in this section describes how you can copy a file from disk to tape (Procedure 3-2, page 32), copy a disk file to tape (Procedure 3-3, page 32), read and write to tape using the `tar`(1) command (Procedure 3-4, page 33), read and write to tape using the `cpio`(1) command (see Procedure 3-5, page 34), and read files to and write files from a multifile tape (Procedure 3-6, page 35).

Procedure 3-2 Using the `cp`(1) Command

To copy a file from disk to tape, use the `cp`(1) command:

1. Reserve a tape by using the `tmrsv`(1) command. In this example, the device group name is `CART`, and the number of devices requested is `1`:

   ```
   tmrsv CART 1
   ```

2. Request a tape mount by using the `tmmnt`(1) command. In this example, the tape has IBM standard labels, a volume identifier of `MYTAPE`, and a path name of `/tmp/tapefile`.

   You must create a path name. Do not remove, rename, or move this file:

   ```
   tmmnt -v MYTAPE -l sl -p /tmp/tapefile -g CART -b 32768 -n -r in
   ```

3. Copy file `myfile` by using the `cp`(1) command-line syntax, as follows:

   ```
   cp myfile /tmp/tapefile
   ```

   The `cp`(1) command copies bytes of data from the disk file to the tape file. It does not format any data, but blocks it into tape records of size 32768 bytes for IBM compatibles devices.

4. Release the reserved tape. The code in this example releases all resources. The tape device is allocated to you until you issue the `tmrls`(1) command with the `-a`, `-d`, or `-p` option. When you issue the `tmrls`(1) command, TMF deletes the associated file, `/tmp/tapefile`.

   ```
   tmrls -a
   ```

Procedure 3-3 Using the `dd`(1m) Command

Use the `dd`(1m) command to copy a disk file to tape.

1. Reserve a tape:

   ```
   tmrsv CART 1
   ```

2. Request a tape mount by using the `tmmnt`(1) command. In this example, the tape has an IBM standard label, the volume identifier is `MYFILE`, it is a new file, and writing is permitted to the tape via the use of the `-r in` parameter:

   ```
   tmmnt -b 4096 -l sl -v MYFILE -p /tmp/tapefile -n -r in -g CART
   ```

3. Use the `dd`(1m) command to copy file `mydisk` to tape, specifying a block size of 4096 bytes:

   ```
   dd if=mydisk of=/tmp/tapefile bs=4096
   ```

4. You can also use the `dd`(1m) command to read the tape file back into file `newfile`:

   ```
   dd if=/tmp/tapefile of=newfile bs=4096
   ```

5. Release the tape resources:

   ```
   tmrls -a
   ```

Procedure 3-4 Using the `tar`(1) Command

The examples in this procedure show you how to read or write to tape by using the `tar`(1) command. You must select the `-f` option of the `tar`(1) command and specify the device path name you used in the `tmmnt`(1) command.

Use the `tar`(1) command to write to tape.

1. Reserve a tape using the default values of the `tmrsv`(1) command:

   ```
   tmrsv CART 1
   ```

2. Request a tape mount by using the `tmmnt`(1) command:

   ```
   tmmnt -l sl -p /tmp/tapefile -b 4096 -v MYFILE -n -r in -g CART
   ```

3. Copy the subtree to tape, starting at the current working directory:

   ```
   tar -cvfb /tmp/tapefile 8 *
   ```

4. Change to a new directory:

   ```
   cd /tmp/newdir
   ```

5. To read the tape back in, copy the tar subtree back from tape to your current working directory:

   ```
   tar -xvfb /tmp/tapefile 8
   ```

6. Release the reserved tape:

   ```
   tmrls -a
   ```

Read a tape that was created in the first part of Procedure 3-4, page 33, or on another UNIX system. In this example, the contents of the tape are read into your current working directory.

1. Reserve a tape:

   ```
   tmrsv CART 1
   ```

2. Request a tape mount by using the `tmmnt`(1) command:

   ```
   tmmnt -l sl -p /tmp/tapefile -v MYFILE -g CART -o
   ```

3. Read the tape by using the `tar`(1) command:

   ```
   tar -xvf /tmp/tapefile
   ```

4. Release the reserved tape:

   ```
   tmrls -a
   ```

Procedure 3-5 Using the `cpio`(1) Command

Use the `cpio`(1) command to read and write to a tape. In this example, data is written to tape; then the `cpio`(1) command is used to read the data from tape:

1. Reserve a tape with a device group name of `CART`:

   ```
   tmrsv CART 1
   ```

2. Request a tape mount using the `tmmnt`(1) command:

   ```
   tmmnt -l sl -v MYTAPE -p /tmp/tapefile -g CART -n -r in
   ```

3. Copy the subtree to tape, starting with the current working directory:

   ```
   find . -print | cpio -Bcov > /tmp/tapefile
   ```

4. Change to a new directory:

   ```
   cd /tmp/newdir
   ```

5. To read the tape back in, copy the `cpio`(1) subtree into your current working directory:

   ```
   cpio -civd < /tmp/tapefile
   ```

6. Release the reserved resources:

   ```
   tmrls -a
   ```

Procedure 3-6 Using the `tmmnt`(1) Command

Multifile volume allocation allows you to read or write different files on a tape without the need for the tape volume to be unloaded and remounted. To accomplish this you must use the same VSN (`-v` parameter) on the `tmmnt`(1) command for each file you wish to process. After the first `tmmnt`(1) command is processed, subsequent `tmmnt`(1) commands with the same VSN do not get sent to the operator.

When using multifile volume allocation, you can use only one file on a tape at a time; that is, you must open a specified file, process the file, and then close it before you can open another file on the same multifile tape volume. You must also reserve a device for each multifile volume tape requested. For example, if the tape files reside on several volumes, you can have files on different volumes opened at the same time; however, you must have reserved enough devices to hold all of the volumes.

The following procedures show you how to use the `-v` parameter of the `tmmnt`(1) command for multifile volume allocation.

Read three files from the same tape without having the operator mount the tape three times.

1. Reserve a tape:

   ```
   tmrsv CART 1
   ```

2. Request a tape mount using the `tmmnt`(1) command. In this example, the tape has an ANSI standard label, the volume identifier is `MYFILE`, and the path names are `one`, `two`, and `three`, with file sequence numbers of the files to be processed as `1`, `2`, and `3`:

   ```
   tmmnt -l al -v MYFILE -p one -q 1 -r in -g CART
   tmmnt -l al -v MYFILE -p two -q 2 -r in -g CART
   tmmnt -l al -v MYFILE -p three -q 3 -r in -g CART
   ```

3. Read the accessed tape files into disk files:

```
cat one > firstfile
cat two > scndfile
cat three > thrdfile
```

4. Release the reserved tape:

```
tmrls -a
```

Use a multifile tape to write three files to the same tape:

1. Reserve a tape:

```
tmrsv CART 1
```

2. Request a tape mount using the `tmmnt`(1) command. In this example, the tape has an ANSI standard label, the volume identifier is `MYFILE`, and the path names are `one`, `two`, and `three`, with file sequence numbers of the files to be processed as `1`, `2`, and `3`. The `-u` option is specified so that the tape will not be unloaded when the process terminates. This option is useful when a tape is used repeatedly, and it minimizes operator time spent mounting tapes:

```
tmmnt -u -l al -v MYFILE -p one -q 1 -n -g CART
tmmnt -u -l al -v MYFILE -p two -q 2 -n -g CART
tmmnt -u -l al -v MYFILE -p three -q 3 -n -g CART
```

3. Write the disk files to the specified tape files:

```
cat file1 > one
cat file2 > two
cat file3 > three
```

4. Release the reserved tape:

```
tmrls -a
```

Writing C Applications Using Tapes

This chapter describes the ways in which you can work with TMF from within C programs.

Before you can access TMF for file processing, you must reserve the required number of tape drives for each device type needed. After you have reserved the tape drives, you may specify the tape volume in which the files to be processed are located.

After you have the volumes mounted and positioned, you can begin processing the tape files. When processing is complete, release the reserved tape drives.

There are two levels of access to TMF. The recommended and easiest to use is the C library level, using flexible file I/O (FFIO) library routines. The second level is to use system calls, which requires much greater detail than the C library level. At this level, access is via the `read`(2) and `write`(2) system calls or the `ioctl`(2) system call.

This chapter contains the following sections:

- Using FFIO
- Using System Calls: `read` and `write`
- Using System Calls: `ioctl`

4.1 Using FFIO

The flexible file I/O (FFIO) routines provide another way to perform tape I/O with the ease of use of system calls. The FFIO routines automatically recognize tape devices and use the appropriate buffering.

The C library routines `ffopen`(3), `ffread`(3), `ffwrite`(3), `ffseek`(3), `ffbksp`(3), `ffclose`(3), and `ffweof`(3) provide the capability to read and write records to tape, rewind the tape and backspace records, and read and write filemarks.

For IBM compatible devices, the `ffread`(3) and `ffwrite`(3) routines provide an interface that is sensitive to block boundaries and that returns information on tape block boundaries on request. With the FFIO layer, a rewind operation can be performed simply with a call to `ffseek`(3). Filemarks can be written with `ffweof`(3), and filemarks can be read with `ffread`(3). A call to `ffwrite`(3) can write a tape block of a designated number of bytes on a tape. A call to `ffread`(3) can read up to

one tape block from a tape. Explicit information about tape block boundaries and the ability to read and write partial tape blocks is available through the use of optional parameters on `ffread`(3) and `ffwrite`(3).

When you use byte-stream mode, end-of-volume processing, user filemarks, and some positioning functionality are not available with the FFIO tape layer. When you use block mode and the FFIO tape layer, each record written must be the same size as specified on the `-b` option of the `tmmnt`(1) command. An exception to this rule is the last record written before a filemark or the end-of-file.

Example 4-1 illustrates how you can use C library routines in a program and Example 4-2, page 40, shows how you execute the program. For more information on the C library routines, see the *MIPSpro Application Programmer's I/O Guide*.

Example 4-1 C Library Routine Usage

The program, called `cexam.c.`, demonstrates how C library routines can be used.

```
#include <fcntl.h>
#include <sys/types.h>
#include <foreign.h>
#include <errno.h>
main()
{
   int ffd;
   int i,j;
   int buf[2000];
   int ret;
   ffd = ffopen("mytape", O_RDWR);
   if (ffd<0){
      printf("open failed, error = %d\n",errno)
      exit(1);
   }

    /************** Write 10 records, a filemark, and 10 more records to tape */

    for (j = 0; j < 2; j++){
      for (i = 0; i < 10; i++){
         ret = ffwrite(ffd, buf, 800);
         if (ret < 800){
         printf("ffwrite returned %d\n",ret);
         printf("error = %d\n",errno);
         }
```

```
      }

    /************** Write a filemark */

      ret = ffweof(ffd);
      if (ret < 0)
         printf("ffweof failed, error = %d\n",errno);
   }

    /************** Rewind the tape */

    ret = ffseek(ffd,0,0);
    if (ret != 0)
      printf("ffseek failed, error = %d\n",errno);

   /************** Read the tape until the first filemark is reached. */

    for (;;)   {
      ret = ffread(ffd, buf, 16000);
      if (ret < 0) {
         printf("ffread failed, error = %d\n",errno);
         break;
      }
      else if (ret == 0)
         break;

   /* Just read a filemark */

      else
         printf("We read %d bytes\n",ret);
   }

    /************** Close the file */

    ffclose(ffd);
}
```

Example 4-2 Executing `cexam.c`

The following commands show how to execute the `cexam.c` program.

```
cc cexam.c -lffio
tmrsv CART 1
tmmnt -v ISCSL -l sl -p mytape -g CART -r in -n -T
assign -F tmf mytape
./a.out
tmrls -a
```

4.2 Using System Calls: `read` and `write`

TMF supports two kinds of unbuffered, synchronous I/O:

- Variable-block I/O, which is supported for all devices with this capability
- Fixed-block I/O, which is supported for all devices

I/O requests are issued with the `read`(2) and `write`(2) system calls.

4.2.1 Variable-block I/O

Variable-block I/O is the default I/O type. Each `read`(2) request transfers one block into your I/O buffer, and each `write`(2) transfers one block from your I/O buffer.

The size specified on the I/O request can vary. However, the request size must be within the driver limits, which can be obtained with the `MTIOCGETBLKINFO ioctl` request (see Section 4.3.2.2, page 78). The `write` size is also limited by the maximum block size specified on the `tmmnt`(1) command with option `-b`. On a `write`(2) system call, the request size cannot exceed this value. On a `read`(2) system call, the request size must be larger than or equal to the size of the next block on tape.

Procedure 4-1 and Procedure 4-2, page 41, show you write and read to a tape file, respectively.

Procedure 4-1 Writing to a Tape File

Shows you how to write a tape file using variable-block I/O.

1. Specify the maximum block size as 98304 bytes on your `tmmnt`(1) command:

```
tmrsv CART 1
tmmnt -v 004141 -l sl -p tapefile -n -g CART -b 98304
```

2. Issue the `write`(2) requests:

```
#include <fcntl.h>

void
main( int argc, char **argv ) {

      char     buf[98304];
      int      fd;
      int      bytes;

      fd = open( "tapefile", O_WRONLY );
      bytes = write( fd, buf, 98304 );
      bytes = write( fd, buf, 32768 );
}
```

Procedure 4-2 Reading a Tape File

Read 98304 bytes followed by another `read`(2) request of 98304 bytes from a tape file that has a maximum block size of 98304 bytes. If you are reading the data just written in the previous example, the first `read` request will return 98304 bytes and the second 32768 bytes.

1. Specify the maximum block size as 98304 bytes on your `tmmnt`(1) command:

```
tmrsv CART 1
tmmnt -v 004141 -l sl -p tapefile -g CART -b 98304
```

2. Issue the `read`(2) requests:

```
#include <fcntl.h>

void
main( int argc, char **argv ) {

      char     buf[98304];
      int      fd;
      int      bytes;
```

```
        fd = open( "tapefile", O_RDONLY );
        bytes = read( fd, buf, 98304 );
        bytes = read( fd, buf, 98304 );
}
```

4.2.2 Fixed-block I/O

To request fixed-block I/O, specify the `-B` option on the `tmmnt`(1) command. The block size is specified with the `-b` option on the `tmmnt`(1) command or with `MTSPECOP ioctl` request (see Section 4.3.2.10, page 88).

TMF sets the block size when processing the `tmmnt`(1) command. If a block size other than what is specified on the `tmmnt`(1) command is required, a `MTSPECOP ioctl` request must be used to modify the block size. The tape must be positioned at the beginning of the tape or at the beginning of the file to be able to modify the block size. The block size must be within the drive limits and cannot exceed the maximum block size specified on the `tmmnt`(1) command.

Multiple blocks can be requested on each `read`(2) or `write`(2) request. The `write` request size must be a multiple of the block size. The `read` request size must be at least as large as the block size. Procedure 4-3 and Procedure 4-4, page 43, show how you write and read to a tape file, respectively.

Procedure 4-3 Writing to a Tape File

Write a tape file using fixed-block I/O. The first `write`(2) request writes 3 blocks each of size 32768 bytes. The second `write` request writes one 32768 byte block. The block size is set by TMF to 32768 bytes when `tmmnt`(1) request is processed.

1. Specify the maximum block size as 32768 bytes on your `tmmnt`(1) command:

   ```
   tmrsv CART 1
   tmmnt -v 004141 -l sl -p tapefile -n -g CART -b 32768
   ```

2. Issue the `write`(2) requests:

   ```
   #include <fcntl.h>

   void
   main( int argc, char **argv ) {

           char   buf[98304];
           int    fd;
   ```

```
        int    bytes;

        fd = open( "tapefile", O_WRONLY );
        bytes = write( fd, buf, 98304 );
        bytes = write( fd, buf, 32768 );
}
```

Procedure 4-4 Reading a Tape File

Read 3 blocks, each 32768 bytes in length, followed by another `read`(2) request which returns one block of size 32768 bytes.

1. Specify the maximum block size as 32768 bytes on your `tmmnt`(1) command:

```
tmrsv CART 1
tmmnt -v 004141 -l sl -p tapefile -g CART -b 32768
```

2. Issue the `read`(2) requests:

```
#include <fcntl.h>

void
main( int argc, char **argv ) {

     char    buf[100000];
     int     fd;
     int     bytes;

       fd = open( "tapefile", O_RDONLY );
       bytes = read( fd, buf, 100000);
       bytes = read( fd, buf, 32768 );
}
```

4.2.3 Status

Information is available about end-of-file, end-of-data, and error status.

4.2.3.1 End-of-File Status

If a filemark is detected on a `read`(2) request, the request returns a short count if data was read and the filemark is processed on the next I/O request. If no data was read, TMF determines if the filemark is embedded in data or indicates an end of the dataset.

If the filemark detected is embedded in data and the user has permission to process these filemarks, the tape is left positioned after the filemark and zero is returned to notify you of the filemark status. Permission to process filemarks is requested with the `-T` option on the `tmmnt`(1) command.

Otherwise, if the user does not have permission to process filemarks embedded in data, the user's request is terminated and no further requests other than a `close`(2) request is accepted. On the `read`(2) request, -1 is returned to notify you of the error.

4.2.3.2 End-of-Data Status

If a filemark is detected on a `read`(2) request and the filemark terminates the dataset, the tape is left positioned before the filemark. All further `read` requests return the filemark status; that is, the zero byte count.

To distinguish between an end-of-file and an end-of-data status, use a `ioctl`(2) system call with a `TMFC_EOD` request (see Section 4.3.1.1, page 45). This request returns 0 if the filemark indicates the end of the data, and -1 if the filemark is a user filemark. Otherwise, since a filemark status will continue to be returned once the end-of-data is detected, a count of the number of consecutive filemarks detected can be used.

4.2.3.3 Error Status

If an error is detected on a `read`(2) request or `write`(2) request, the I/O request returns a short count or -1 if no data was input or output. If a short count is returned, the next I/O request returns -1 to notify you of the error.

If a negative byte count is returned, `errno` is set to identify the error. Additional information relating to the error is also available in the user's message file, which is specified when the user reserves resources with the `tmrsv`(1) command.

4.3 Using System Calls: `ioctl`

TMF supports `ioctl`(2) requests, which perform tape device operations, set certain device attributes, provide TMF status information and device information, and perform user end-of-volume (EOV) special processing. The `ioctl`(2) request codes, structures, and field values are defined in the TMF `ioctl` definition file, `/usr/include/tmf/sys/tmfctl.h`, and the tape `ioctl` definition file, `/usr/include/`

sys/mtio.h for IRIX systems and /usr/include/tmf/mtio.h for Linux systems (see Section 4.3.2, page 76). Table 4-1, page 45, describes these files:

Table 4-1 ioctl Definition Files

File	Description
/usr/include/tmf/sys/tmfctl.h	The ioctl(2) requests defined in the tmfctl.h file either return TMF status information or perform tape device operations via the TMF daemon and its child processes.
/usr/include/sys/mtio.h (IRIX systems); /usr/include/tmf/mtio.h (Linux systems)	The ioctl(2) requests defined in this file are primarily for users who are not using TMF to access tape devices, but are instead using the tpsc tape interface (see the tpsc(7M) man page. TMF users can access mtio.h requests that do no not involve tape movement or the setting of attributes. TMF controls the other tape operations, which users can request using the TMF ioctl interface defined in tmfctl.h.

4.3.1 tmfctl.h ioctl Requests

The following ioctl(2) requests are defined in the tmfctl.h file:

- Status ioctl(2) requests
- TMF daemon request or reply ioctl(2) requests

4.3.1.1 Status ioctl Requests

Using the ioctl(2) system call with the TMFC_DAEMON and TMFC_EOD requests provides status information.

TMFC_DAEMON returns TMF status. The ioctl(2) system call returns an exit code of 0 if TMF is up; otherwise, an exit status of -1 is returned.

TMFC_EOD returns the state of a tape file. If the tape is positioned at the end-of-data, 0 is returned; otherwise, -1 is returned.

4.3.1.2 TMF Daemon Request or Reply `ioctl` Requests

The `ioctl`(2) system call supports the `TMFC_DMNREQ` and `TMFC_DMNREP` requests, which provide positioning and related status information.

`TMFC_DMNREQ` issues requests to TMF daemon. These requests include tape positioning requests, user end-of-volume (EOV) special processing requests, filemark writes, and informational requests. The `TMFC_DMNREQ` request can be sent synchronously or asynchronously.

`TMFC_DMNREP` obtains the status of an asynchronous TMF daemon request.

On synchronous requests, a reply is not sent until TMF daemon completes the request.

On asynchronous requests, a reply to the `TMFC_DMNREQ` request is returned after TMF daemon has been forwarded the request. Once the TMF daemon has completed the request, the status of the request can be obtained with the `TMFC_DMNREP` request. If the `TMFC_DMNREP` request is issued before the request completes, TMF waits for the request to complete before replying to the user.

The TMF daemon request consists of a header and, for those requests which require additional information, a request body. The header has the following format:

```
typedef struct  tmfreqhdr {
        int     request;
        int     length;
        int     reply;
        int     residual;
        int     async;
} tmfreqhdr_t;
```

Structures for requests which require more information than what is provided in `tmfreqhdr_t` are also defined in `tmfctl.h`. These structures consist of the request header, `tmfreqhdr_t`, followed by information specific to the request.

The following `tmfreqhdr_t` values must be set in the request:

Value	Description
`request`	TMF request type
`length`	Size of the request excluding the header
`async`	Asynchronous processing enabled

The following TMF daemon requests are supported:

Request	Description
TR_CLV	Closes the current tape volume
TR_EOV	Selects or deselects user end-of-volume (EOV) processing
TR_INFO	Returns TMF stream information
TR_PABS	Performs absolute positioning
TR_PBLKS	Performs block positioning
TR_PFMS	Performs file positioning
TR_PVOL	Performs volume positioning using an index
TR_PVSN	Performs volume positioning using a volume name
TR_RWD	Rewinds the tape file
TR_WFM	Writes a filemark

On synchronous requests, TMF returns the status of the request in the `reply` field, and a residual in the `residual` field. The reply status values are defined in the `/usr/include/tmf/tmferr.h` file. `TMFC_DMNREQ` and `TMFC_DMNREP` return the following `tmferr.h` values.

Value	Description
ETBOF	The beginning of the file was detected.
ETBOT	The beginning of the tape was detected.
ETBRQ	An invalid request was sent to the tape device.
ETEOD	The end of recorded data was detected.
ETEOF	The end-of-file was detected.
ETEOM	The end-of-media was detected.
ETFMNA	Filemark processing is not permitted.
ETFMS	The filemark was detected.
ETIVSN	An invalid VSN was specified.
ETLBL	An invalid label structure or sequence was used.
ETLOADF	A volume load failure occurred.
ETMEDIA	A media error occurred.
ETNDY	The device is not ready,

ETNOP	The device is not operational.
ETNVS	The VSN was not found.
ETOFF	An invalid VSN offset was specified.
ETRST	The device was reset.
ETUDE	An unrecoverable data error occurred.

The `residual` value is that portion of the request which did not complete. This field is valid only for block and file positioning requests and write filemark requests.

`TMFC_DMNREP` is used to obtain the status of a TMF daemon request that was issued asynchronously. The argument to the `TMFC_DMNREP` request is a pointer to the `tmfrep_t data` structure:

```
typedef  struct   tmfrep {
         int      reply;
         int      residual;
         int      datalen;
         void     *databuf;
} tmfrep_t;
```

For requests that return data, you set the following `tmfrep_t` values in the requests:

Value	Description
`databuf`	Pointer to a data buffer to which information gathered by TMF will be copied
`datalen`	Length of the data buffer

The `reply` and `residual` fields are returned by TMF and return the same values as in the `TMF_DMNREQ` request.

TMF supports the following types of requests:

- Positioning requests
- User end-of-volume (EOV) positioning requests
- Write filemark requests
- Information requests
- Positioning requests

- User end-of-volume (EOV) positioning requests
- Write filemark requests
- Information requests

4.3.1.3 Positioning Requests

The following requests control positioning:

- Rewind positioning
- Block positioning
- File positioning
- Absolute positioning
- Volume index positioning
- Volume name positioning

4.3.1.3.1 Rewind Positioning

The rewind request positions a tape to the beginning of the current file as shown in Example 4-3.

Example 4-3 Synchronous Rewind Request

If the beginning of the file is on a volume which is not currently mounted, TMF mounts the correct volume before positioning to the beginning of file. If the file is a concatenated file, TMF positions to the beginning of the first file of the concatenated set. If this request is issued after a `write`(2) request, the volume is terminated before positioning the tape.

The argument to the rewind request is a pointer to the `tmfreqhdr_t` structure with the request code set to `TR_RWD`.

```
#include <errno.h>
#include <sys/fcntl.h>
#include <tmf/sys/tmfctl.h>

void
main( int argc, char **argv ) {
```

```
    tmfpblk_t  req;
    int fd;
    int c;

    /*
     *   Open the TMF path.  "tmfpath" is the path name specified on
     *   the tmmnt command.
     */

    fd = open( "tmfpath", O_RDWR );
    if ( fd < 0 ) {
         printf( "Unable to open file tmfpath (errno = %d)\n",
                                                    errno );
         exit(errno);
    }

    /*
     *   Create and send the TMF rewind request.
     */

     req.request = TR_RWD;
     req.async   = 0;
     req.length  = 0;
     req.reply   = 0;
     c = ioctl( fd, TMFC_DMNREQ, &req );
     if ( c < 0 ) {
         printf( "Unable to issue the TMF rewind request (errno=%d)\n",
         errno );
         exit(errno);
     }

    /*
     *   Check the status of the rewind request.
     */

     if (req.reply ) {
         printf( "Error %d on the TMF rewind request\n", req.reply);
         exit(EIO);
     }
     exit(0);
}
```

4.3.1.3.2 Block Positioning

The block positioning request positions forward or backward by blocks.

Block positioning can span volumes for multivolume or concatenated files. TMF mounts the next volume in a volume set if the end-of-volume (EOV) is detected on forward positioning requests. Positioning then continues at the beginning of the file section on the next volume. TMF mounts the previous volume if the beginning-of-volume (BOV) is detected on a backward positioning request. Positioning then continues at the end of the file section on this volume.

Example 4-4 illustrates a synchronous block positioning request. If this request is issued after a `write`(2) request, the volume is terminated before positioning the tape.

Example 4-4 Synchronous Block Positioning Request

The argument to the block positioning request is a pointer to the `tmfpblk_t` structure with the request code set to `TR_PBLKS`. The `count` field of the `tmfpblk_t` structure specifies the number of blocks to position.

A positive `count` value, N, positions the tape forward. The tape is left positioned after the Nth block on the end-of-tape (EOT) side of the block.

A negative count value, N, positions the tape backwards. The tape is left positioned on the beginning-of-tape (BOT) side of the Nth block. A count value of zero, positions the tape backward one block and then forward one block, leaving the tape at the same position.

The error status is returned in the `reply` field of the `tmfreqhdr_t` or `tmfrep_t` structure. If one of the following error conditions is detected, block positioning is terminated and the error returned.

```
#include <errno.h>
#include <stdlib.h>
#include <sys/fcntl.h>
#include <tmf/sys/tmfctl.h>

void
main( int argc, char **argv ) {

    tmfpblk_t  req;
    int  fd;
    int  c;
```

```
    /*
     *   Open the TMF path.  "tmfpath" is the path name specified on
     *   the tmmnt command.
     */

    fd = open( "tmfpath", O_RDWR );
    if ( fd < 0 ) {
       printf( "Unable to open file tmfpath (errno = %d)\n", errno );
                exit(errno);
    }

    /*
     *   Create and send a request to position forward 5 blocks.
     */

    req.rh.request  = TR_PBLKS;
    req.rh.length   = sizeof(tmfpblk_t) - sizeof(tmfreqhdr_t);
    req.rh.async    = 0;
    req.rh.reply    = 0;
    req.rh.residual = 0;
    req.count       = 5;
    c = ioctl( fd, TMFC_DMNREQ, &req );
    if ( c < 0 ) {
       printf("Unable to issue the block position request (errno=%d)\n",
                                            errno );
       exit(errno);
    }

    /*
     *   Check the status of the block positioning request.
     */

    if ( req.rh.reply ) {
       printf("Error %d on the block position request\n", req.rh.reply );
       printf("%d blocks of the requested %d blocks were positioned\n",
                abs(req.count) - req.rh.residual, abs(req.count) );
       exit(EIO);
    }
    exit(0);
}
```

4.3.1.3.3 File Positioning

The file positioning request positions forward or backward by filemarks.

File positioning can span volumes for multivolume or concatenated files. TMF mounts the next volume in a volume set if the end-of-volume (EOV) is detected on forward positioning requests. Positioning then continues at the beginning of the file section on the next volume. TMF mounts the previous volume if the beginning-of-volume is detected on a backward positioning request. Positioning then continues at the end of the file section on this volume.

If this request is issued after a `write`(2) request, the volume is terminated before positioning the tape.

To perform file positioning, you enable filemark processing with the `-T` option on the `tmmnt`(1) command.

Example 4-5 illustrates a synchronous file positioning request.

Example 4-5 Synchronous File Positioning Request

The argument to the block positioning request is a pointer to the `tmfpfm_t` structure with the request code set to `TR_PFMS`. The `count` field of the `tmfpfm_t` structure specifies the number of files to position.

A positive `count` value, N, positions the tape forward. The tape is left positioned after the Nth file on the end-of-tape (EOT) side of the filemark.

A negative `count` value, N, positions the tape backwards. The tape is left positioned on the beginning-of-tape (BOT) side of the Nth filemark.

A `count` value of zero does not change the tape position.

```
#include <errno.h>
#include <stdlib.h>
#include <sys/fcntl.h>
#include <tmf/sys/tmfctl.h>

void
main( int argc, char **argv ) {

   tmfpfm_t  req;
   int  fd;
   intc;
```

```
    /*
     *  Open the TMF path.  "tmfpath" is the path name specified on
     *   the tmmnt command.
     */

   fd = open( "tmfpath", O_RDWR );
   if ( fd < 0 ) {
        printf( "Unable to open file tmfpath (errno = %d)\n", errno );
        exit(errno);
   }

   /*
    *   Create and send a request to position backward 7 filemarks.
    */

   req.rh.request  = TR_PFMS;
   req.rh.length   = sizeof(tmfpfm_t) - sizeof(tmfreqhdr_t);
   req.rh.async    = 0;
   req.rh.reply    = 0;
   req.rh.residual = 0;
   req.count       = -7;
   c = ioctl( fd, TMFC_DMNREQ, &req );
   if ( c < 0 ) {
        printf("Unable to issue the file position request (errno=%d)\n",
                                                          errno );
        exit(errno);
   }

   /*
    *   Check the status of the file positioning request.
    */

   if ( req.rh.reply ) {
        printf("Error %d on the file position request\n", req.rh.reply );
        printf("%d files of the requested %d files were positioned\n",
                 abs(req.count) - req.rh.residual, abs(req.count) );
        exit(EIO);
   }
   exit(0);
}
```

4.3.1.3.4 Absolute Positioning

The absolute positioning request moves the tape to a specified address. This request can only be performed by tape administrators.

If this request is issued after a `write`(2) request, the volume is terminated before positioning the tape.

Example 4-6 illustrates a synchronous absolute positioning request.

Example 4-6 Synchronous Absolute Positioning Request

The argument to the absolute positioning request is a pointer to the `tmfpabs_t` structure with the request code set to `TR_PABS`. The `blkaddr` field of the `tmfpabs_t` structure specifies the address of the block to position to. You can obtain the address of a block with a `MTIOCGET_SGI` or `MTIOCGETEXT` `ioctl` call (see Section 4.3.2.3, page 78). The address is returned in the `mt_blkno` field of these requests.

```
#include <errno.h>
#include <sys/fctnl.h>
#include <tmf/sys/tmfctl.h>

void
main( int argc, char **argv )
{
   tmfpabs_t  req;
   intfd;
   intc;

   /*
    *   Open the TMF path.  "tmfpath" is the path name specified on
    *   the tmmnt command.
    */

   fd = open( "tmfpath", O_RDWR );
   if ( fd < 0 ) {
        printf( "Unable to open file tmfpath (errno = %d)\n", errno );
         exit(errno);
   }

   /*
    *   Create and send a request to position to the specified block.
    *   The block location is obtained with the MTIOCGET or MTIOCGETEXT
```

```
     *   ioctl.
     */

    req.rh.request = TR_PABS;
    req.rh.length  = sizeof(tmfpabs_t) - sizeof(tmfreqhdr_t);
    req.rh.async   = 0;
    req.rh.reply   = 0;
    req.blkaddr    = block-address;
    c = ioctl( fd, TMFC_DMNREQ, &req );
    if ( c < 0 ) {
         printf("Unable to issue the absolute positioning request (errno=%d)\n",
                                                          errno );
         exit(errno);
    }

    /*
     *   Check the status of the absolute positioning request.
     */

    if ( req.rh.reply ) {
         printf("Error %d on the absolute positioning request\n",
                                                         req.rh.reply );
         exit(EIO);
    }
    exit(0);
}
```

4.3.1.3.5 Volume Index

The volume index positioning request positions to the beginning-of-volume (BOV) of the volume with the specified offset in the volume identifier list as shown in Example 4-7. It illustrates a synchronous volume positioning request using an index.

Example 4-7 Synchronous Volume Positioning Request

The argument to the volume index positioning request is a pointer to the `tmfpvol_t` structure with the request code set to `TR_PVOL`. The `index` field of the `tmfpvol_t` structure specifies the index of the volume to position to. 1 specifies the first volume in the volume list, 2 specifies the second volume, and so on.

You specify the volume list with the `tmmnt`(1) command. In this example, if the volume list on the `tmmnt`(1) command is specified as follows, then an index of 4 positions to the beginning-of-volume 004143.

```
tmmnt -v 00410:004141:004142:004143 -1 sl -p tmfpath ...
```

```
#include <errno.h>
#include <sys/fcntl.h>
#include <tmf/sys/tmfctl.h>

void
main( int argc, char **argv )
{
    tmfpvol_t  req;
    int     fd;
    int     c;

    /*
     *   Open the TMF path.  "tmfpath" is the path name specified on
     *   the tmmnt command.
     */

    fd = open( "tmfpath", O_RDWR );
    if ( fd < 0 ) {
         printf( "Unable to open file tmfpath (errno = %d)\n", errno );
         exit(errno);
    }

    /*
     *   Create and send a request to position to the beginning of the
     *   volume with an index of 2 in the volume list.
     */

    req.rh.request  = TR_PVOL;
    req.rh.length   = sizeof(tmfpvol_t) - sizeof(tmfreqhdr_t);
    req.rh.async    = 0;
    req.rh.reply    = 0;
    req.index       = 2;
    c = ioctl( fd, TMFC_DMNREQ, &req );
    if ( c < 0 ) {
         printf("Unable to issue the volume positioning request (errno=%d)\n",
                                                        errno );
```

```
        exit(errno);
    }

    /*
     *   Check the status of the volume positioning request.
     */

    if ( req.rh.reply ) {
        printf("Error %d on the volume positioning request\n",
                                                    req.rh.reply );
        exit(EIO);
    }
}
```

4.3.1.3.6 Volume Name

The volume name positioning request positions to the beginning-of-volume of the volume with the specified external volume identifier. Example 4-8 illustrates a synchronous volume positioning request using a volume identifier.

Example 4-8 Synchronous Volume Positioning Request (Volume Identifier)

The argument to the volume index positioning request is a pointer to the `tmfpvsn_t` structure with the request code set to `TR_PVSN`. The `evsn` field of the `tmfpvsn_t` structure specifies the external volume identifier of the volume to position to. The `fvsn` field of the `tmfpvsn_t` structure specifies the format identifier of the volume to position to.

Note: The `fvsn` field is only applicable to devices which support format identifiers and support for these devices is deferred.

```
#include <errno.h>
#include <sys/fcntl.h>
#include <tmf/sys/tmfctl.h>

void
main( int argc, char **argv )
{
   tmfpvsn_t  req;
   intfd;
   intc;
```

```
   /*
    *   Open the TMF path.  "tmfpath" is the path name specified on
    *   the tmmnt command.
    */

   fd = open( "tmfpath", O_RDWR );
   if ( fd < 0 ) {
        printf( "Unable to open file tmfpath (errno = %d)\n", errno );
        exit(errno);
   }

   /*
    *   Create and send a request to position to the beginning of the
    *   volume with the external volume identifier of "004141".
    */

   req.rh.request  = TR_PVSN;
   req.rh.length   = sizeof(tmfpvsn_t) - sizeof(tmfreqhdr_t);
   req.rh.async    = 0;
   req.rh.reply    = 0;
   strcpy( req.evsn, "004141" );
   c = ioctl( fd, TMFC_DMNREQ, &req );
   if ( c < 0 ) {
        printf("Unable to issue the volume positioning request (errno=%d)\n",
                                                        errno );
        exit(errno);
   }

   /*
    *   Check the status of the volume positioning request.
    */

   if ( req.rh.reply ) {
        printf("Error %d on the volume positioning request\n",
                                                        req.rh.reply );
        exit(EIO);
   }
   exit(0);
}
```

4.3.1.4 User End-of-Volume Processing Requests

User end-of-volume (EOV) processing allows a user to close a volume or gain control when the end-of-tape (EOT) or EOV is detected. The following are EOV processing requests:

- Selection and deselection
- Close volume

Example 4-9 illustrates a synchronous user EOV selection and deselection request as well as a close volume request.

4.3.1.4.1 Selection and Deselection

The user end-of-volume (EOV) selection request controls when the EOT is detected on a `write`(2) request or when the EOV is detected on a `read`(2) request. The default action is to terminate the current volume and then mount the next volume when the EOT is detected or to mount the next volume when the EOV is detected.

Obtaining control at the EOT or EOV allows a user to determine how much data is written at the EOT by either reading back data or writing additional data, to terminate a volume with the user's own volume termination data, to write the user's own volume header information after the next volume is mounted, or to perform any other additional processing before the next volume is mounted.

4.3.1.4.2 Close Volume

The close volume request closes the current volume and mounts the next volume of the volume set. If data was being output, the request terminates the current volume before mounting the next volume. If user EOV processing is selected and the EOT is detected before the close volume request is issued, the volume is terminated regardless of the type of I/O taking place between the EOT detection and the close volume request.

For example, if, after the EOT is detected, the tape is positioned backward five blocks before a close volume request is issued, the file is terminated at this point and the five blocks are lost.

Example 4-9 EOV Selection and Deselection Request, and Close Request

The argument to the TMF user EOV selection or deselection request is a pointer to the `tmfeov_t` structure with the request code set to `TR_EOV`. The `select` field of the

`tmfeov_t`structure is set to a nonzero value to select user EOV processing and is set to zero to deselect user EOV processing.

The argument to the close volume request is a pointer to the `tmfreqhdr_t` structure with the request code set to `TR_CLV`.

```
#include <errno.h>
#include <stdlib.h>
#include <sys/fcntl.h>
#include <tmf/sys/tmfctl.h>

void
closev( int );
void
eov( int, int );
int  wrttape( int, char *, int *, int * );

*define BUFSIZE  98304

void
main( int argc, char **argv ) {

   charbuf[BUFSIZE];
   int  count=0
   int  datalen;
   int  fd, c, i;

   /*
    *   Open the TMF path. "tmfpath" is the path name specified on
    *   the tmmnt command.
    */

   fd = open( "tmfpath", O_RDWR );
   if ( fd < 0 ) {
        printf( "Unable to open file tmfpath (errno = %d)\n", errno);
        exit(errno);
   }

   /*
    *   Create and send a request to select user end-of-volume processing.
    */
```

```
   eov(fd,1);

   /*
    *   Write until EOT.
    */

   while ( !wrttape( fd, buf, &count, &datalen ) );

   /*
    *   Complete the last block.
    */

   if ( datalen ) {
        c = write( fd, buf+datalen, BUFSIZE-datalen );
        if ( c != BUFSIZE-datalen )
                 exit(EIO);
   }

   /*
    *   Close the volume.
    */

   closev(fd);

   /*
    *   Deselect user end-of-volume processing.
    */

   eov(fd,0);

   /*
    *   Output additional data on the next volume.
    */

   for ( i=0; i , 5; i++ ) {
        wrttape( fd, buf, &count, &datalen );
   }
   exit(0);
}

/*
```

```
 *   closev
 *
 *   Terminate the current volume and mount the next.
 *
 */

void
closev( int fd ) {

   tmfreqhdr_t  req;
   int  c;

   req.request = TR_CLV;
   req.length  = 0;
   req.async   = 0;
   req.reply   = 0;
   c = ioctl( fd, TMFC_DMNREQ, &req );
   if (c < 0 ) {
        printf( "Unable to issue the close volume request (errno=%d)\n",
                                                        errno );
        exit(errno);
   }

   /*
    *   Check the status of the close volume request.
    */

   if ( req.reply ) {
        printf( "Error %d on the close volume request\n", req.reply );
        exit(EIO);
   }
}

/*
 *  eov
 *
 *  Select or deselect user end-of-volume processing.
 *
 */

void
```

```
eov( int fd, int value ) {
    tmfeov_t req;
    int c;

    req.rh.request = TR_EOV;
    req.rh.length  = sizeof(tmfeov_t) - sizeof(tmfreqhdr_t);
    req.rh.async   = 0;
    req.rh.reply   = 0;
    req.select     = value;
    c = ioctl( fd, TMFC_DMNREQ, &req );
    if ( c < 0 ) {
         printf( "Unable to issue the eov request (errno=%d)\n", errno );
         exit(errno);
    }

    /*
     *   Check the status of the eov request.
     */

    if ( req.rh.reply ) {
         printf('Error %d on the eov request\n", req.rh.reply );
         exit(EIO);
    }
}

/*  wrttape
 *
 *  Returns;
 *    0        No error
 *    1        EOT
 *
 */

int
wrttape( int fd, char *buf, int *count, int *datalen ) {

    char *bp = buf;
    int  *bpw = (int *)buf;
    int  nbytes, len;

    len    = BUFSIZE;
```

```
    bpw[0] = *count;
    while ( len ) {
         nbytes = write( fd, bp, len );
         if ( nbytes < 0 ) {

/* Error on write */

              if ( errno == ENOSPC )
                   return(1);

                       /* At End-of-Tape */

              printf( "Unrecoverable write error (errno = %d)\n", errno );
              exit(1);
         }
         *datalen = len - nbytes;
         bp  += nbytes;
         len -= nbytes;
     }
     (*count)++;
     return(0);
}
```

4.3.1.5 Write Filemark Requests

The write filemark requests allow you to output one or more filemarks to tape. You enable filemark writes with the `-T` option on the `tmmnt`(1) command. Example 4-10 illustrates a synchronous write filemark request.

Example 4-10 Synchronous Write Filemark Request

The argument to a write filemark request is a pointer to the `tmfwfm_t` structure with the request code set to `TR_WFM`. The `count` field of the `tmfwfm_t` structure specifies the number of filemarks to write.

```
#include <errno.h>
#include <sys/fcntl.h>
#include <tmf/sys/tmfctl.h>

void
main( int argc, char **argv )
{
```

```
tmfwfm_t  req;
intfd;
intc;

/*
 *   Open the TMF path.  "tmfpath" is the path name specified on
 *   the tmmnt command.
 */

fd = open( "tmfpath", O_RDWR );
if ( fd < 0 ) {
     printf( "Unable to open file tmfpath (errno = %d)\n", errno );
     exit(errno);
}

/*
 *   Create and send a request to write 2 filemarks.
 */

req.rh.request  = TR_WFM;
req.rh.length   = sizeof(tmfwfm_t) - sizeof(tmfreqhdr_t);
req.rh.async    = 0;
req.rh.reply    = 0;
req.rh.residual = 0;
req.count       = 2;
c = ioctl( fd, TMFC_DMNREQ, &req );
if ( c < 0 ) {
     printf("Unable to issue the write filemark request (errno=%d)\n",
                                                        errno );
     exit(errno);
}

/*
 *   Check the status of the write filemark request.
 */

if ( req.rh.reply ) {
     printf("Error %d on the write filemark request\n", req.rh.reply );
     printf("%d filemarks of the requested %d filemarks were written\n",
                  req.count - req.rh.residual, req.count );
     exit(EIO);
```

```
    }

    exit(0);
}
```

4.3.1.6 Information Requests

The information requests return tape stream information. These requests return information such as the current block size, maximum block size, label type, device type, device name, current VSN, file expiration date, VSN list, and the last device function and status.

The argument to an information request is a pointer to the `tmfinfo_t` structure with the request code set to `TR_INFO`. The details differ for synchronous and asynchronous requests:

TMF returns the stream information data in the `tsdata_t` structure followed by a list of 8 character volume names. `tsdata_t` is defined in the `/usr/include/tmf/tmfreq.h` include file; this structure contains the following fields:

Field	Description
`ts_ba`	Block attribute of the tape dataset. On new files, you define this value with the `-F` option on the `tmmnt`(1) command. On old files, you obtain this value from the tape label. The following are valid values:

Value	Description
`B`	Blocked
`S`	Spanned, for variable-length records
`S`	Standard, for fixed-length records
`R`	Blocked and spanned records, if variable length
`R`	Blocked and standard records, if fixed length

`ts_block`	File block number. This value includes user filemarks as well as blocks. If absolute positioning takes place, this value is no longer valid.

`ts_blocksize` File block size. This value is valid only for fixed block-length files.

`ts_bnum` Block number relative to the last user filemark written.

`ts_cvsn` Index in the volume identifier list of the current volume. 0 is the first volume in the volume list; 1 is the second volume and so on.

`ts_day` Current day in the year. Valid values are 1 through 366.

`ts_den` Requested density value. The following are valid values:

Value	Description
`DEN_800`	800 bpi
`DEN_1600`	1600 bpi
`DEN_3200`	3200 bpi
`DEN_4000`	4000 bpi
`DEN_7000`	7000 bpi
`DEN_8200`	8200 bpi
`DEN_8500`	8500 bpi

`ts_dev` Major or minor device number of the assigned tape device.

`ts_dgn` Resource name of the assigned device.

`ts_dst` Last device status represented by a set of flags. The following device status flags are in the `/usr/include/sys/tpsc.h` file (IRIX systems) and the `/usr/include/tmf/tpsc.h` file (Linux systems):

Flag	Description
`CT_EOM`	At end-of-media.
`CT_BOT`	At beginning-of-tape (BOT).
`CT_WRP`	Write protected volume.
`CT_EW`	At end-of-tape.

CT_GETBLKLEN	Block length request is pending.
CT_MOTION	Last command moved the tape.
CT_ONL	Device is online.
CT_QIC24	Low density tape on Viper 150 device.
CT_QIC120	High density tape on Viper 150 device.
CT_OPEN	Tape device is open.
CT_READ	Last tape movement request was a read(2) request.
CT_WRITE	Last tape movement request was a write(2) request.
CT_CHG	Unit attention occurred
CT_DIDIO	A tape movement request has been done since the device is open.
CT_EOD	At end of recorded data.
CT_FM	At filemark.

ts_dty — Type of the assigned device. Valid device types are defined in the /usr/include/sys/invent.h file (IRIX systems) and the /usr/include/tmf/invent.h file (Linux systems) beginning with the TPUNKNOWN value.

ts_dvn — Name of the assigned device.

ts_eov — User end-of-volume selection state. The following are valid values:

Value	Description
0	User end-of-processing is not selected.

	1	User end-of-processing is selected.

`ts_eovproc` User end-of-volume processing state. The following are valid values:

Value	Description
0	No user end-of-processing.
1	In user end-of-processing; that is, the EOT or EOV was detected, and a new volume has not yet been mounted.

`ts_erreg` Last device error status. The device status flags can be found in the `/usr/include/sys/tpsc.h` file (IRIX systems) and the `/usr/include/tmf/tpsc.h` file (Linux systems).

The `ts_erreg[0]` error status is represented by the upper 16 bits of the following flags:

Flag	Description
`CT_NEEDBOT`	Positioning is required before I/O.
`CT_LOADED`	Tape has been loaded.
`CT_ANSI`	I/O is permitted after EOT.
`CT_SMK`	Drive is at a setmark.
`CT_AUDIO`	Drive is in audio mode.
`CT_AUD_MED`	Media is recorded in audio format.
`CT_MULTPART`	Multi-partitioned volume is loaded.
`CT_SEEKING`	Seek request is pending.
`CT_HITFMSHORT`	Filemark was detected, but the status has not yet been returned to the user.
`CT_INCOMPAT_MEDIA`	Media is incompatible with the device.

CT_CLEANHEADS	Device needs to be cleaned.
CT_COMPRESS	Device is in compression mode.
CT_MEDIA_ERR	Media error has been detected.
CT_EOD	At end of recorded data.
CT_FM	At filemark.

The ts_erreg[1] error status is represented by the upper 16 bits of the following flags:

Flag	Description
CT_BAD_REQT	An invalid request was issued to the device.
CT_LARGE_BLK	The block on tape is larger than the requested read byte count.
CT_LOAD_ERR	Volume load failed.
CT_HWERR	Hardware error occurred.
CT_NOT_READY	Device is not ready.
CT_BLKLEN	Actual recorded block size differs from the set size.

The remaining ts_erreg statuses are undefined.

ts_fcn

Last user request issued. The following user request codes are defined in the /usr/include/sys/mtio.h file (IRIX systems) and the /usr/include/tmf/mtio.h file (Linux systems):

Code	Description
MTR_READ	read(2) request
MTR_WRITE	write(2) request
MTR_WFM	Write filemark(s)

	MTR_SRB	Skip records backward
	MTR_SRF	Skip records forward
	MTR_SFB	Skip filemarks backward
	MTR_SFF	Skip filemarks forward
	MTR_SEOD	Space to the end-of-data
	MTR_SEOM	Space to the end-of-media
	MTR_FORMAT	Volume format
	MTR_PART	Position to a partition
	MTR_SSM	Skip filemarks
	MTR_WSM	Write filemarks
	MTR_MODEAUD	Enable or disable audio mode
	MTR_REW	Rewind
	MTR_ERASE	Erase from current position to the EOT
	MTR_RETEN	Retention
	MTR_UNLOAD	Unload
	MTR_PABS	Position to an absolute address
	MTR_PAUDIO	Audio position
	MTR_GAUDIO	Get audio position
	MTR_RDLOG	Read log
	MTR_ATTR	Set attribute
	MTR_SPOS	Set vendor specific position
	MTR_GPOS	Get vendor specific position
ts_ffseq	File sequence number of first file on a volume. 1 is the first file of a volume set, 2, the second, and so on.	
ts_fid	File identifier.	

`ts_first` Index of a volume in the volume identifier list of the volume in which a file begins. 1 is the first volume in the volume identifier list.

`ts_fmdir` Direction from the last user filemark. The following are valid values:

Value	Description
0	After filemark
1	Before filemark

`ts_fsec` File section number, that is, the number of the volume of the dataset. 1 is the first volume of the volume set.

`ts_fseq` File sequence number. 1 is the first file residing on the volume set.

`ts_fst` File status. The following are valid values:

Value	Description
`FST_NEW`	New file status
`FST_OLD`	Old file status
`FST_APP`	Append file status

`ts_h1` HDR1 label.

`ts_h2` HDR2 label.

`ts_lb` Label type. The following are valid values:

Value	Description
`LBL_NS`	Label type is not specified on the `tmmnt`(1) command.
`LBL_NL`	Nonlabeled.
`LBL_AL`	ANSI labeled tape.
`LBL_SL`	IBM standard label.
`LBL_BLP`	Bypass label processing.
`FST_ST`	Single filemark format.
`FST_ULP`	User label processing.

`ts_mbs` Maximum block size.

`ts_numvsn` Number of volumes in a volume set.

`ts_ord` Stream ordinal.

`ts_path` Stream path.

`ts_rf` Record format of the tape dataset. On new files, you define this value with the `-F` option on the `tmmnt`(1) command. On old files, you obtain it from the tape label. The following are valid values:

Value	Description
`D`	Variable length
`F`	Fixed length
`S`	Spanned
`U`	Undefined length

`ts_ring` Write ring status. The following are valid values:

Value	Description
0	Ring out
1	Ring in

`ts_rl` Record length is defined with the `-L` option on the `tmmnt`(1) command.

`ts_urwfm` Set if user read or write of filemarks is permitted.

`ts_vl` VOL1 label.

`ts_vsnoff` Offset to the VSN list from the beginning of the `tsdata` structure.

`ts_xday` File expiration day. Valid values are 1 through 366.

`ts_xyear` File expiration year. Valid values are 00 through 99.

`ts_year` Current year. Valid values are 00 through 99.

Example 4-11 illustrates a synchronous information request.

Example 4-11 Synchronous Information Request

```
include # <errno.h>
#include <sys/fcntl.h>
#include <tmf/sys/tmfctl.h>
#include <tmf/tmfreq.h>

void
main( int argc, char **argv )
{

   tmfinfo_t  req;
   tsdata_t   *ts;
   cha     *vsnp;
   int     fd;
   int     c;
   int     i;

   /*
    *   Open the TMF path.  "tmfpath" is the path name specified on
    *   the tmmnt command.
    */

   fd = open( "tmfpath", O_RDWR );
   if ( fd < 0 ) {
        printf( "Unable to open file tmfpath (errno = %d)\n", errno );
        exit(errno);
   }

   /*
    *   Create and send a request for tape stream information.
    */

   req.rh.request  = TR_INFO;
   req.rh.length   = sizeof(tmfinfo_t) - sizeof(tmfreqhdr_t);
   req.rh.async    = 0;
   req.rh.reply    = 0;
   req.datalen     = sizeof(tsdata_t) + MAXVSN * L_MAXVSN;
   req.databuf     = (void *)calloc( 1, req.datalen );
   c = ioctl( fd, TMFC_DMNREQ, &req );
   if ( c < 0 ) {
        printf("Unable to issue an information request (errno=%d)\n",
```

```
                                                    errno );
        exit(errno);
    }

    /*
     *   Check the status of the stream information request.
     */

    if ( req.rh.reply ) {
        printf("Error %d on the stream information request\n",
                                                    req.rh.reply );
        exit(EIO);
    }

    /*
     *   Output selected fields from the returned information.
     */

    ts = (tsdata_t *)req.databuf;
    printf( "Stream Ordinal ........ %d\n",   ts->ts_ord    );
    printf( "Stream Path ........... %s\n",   ts->ts_path   );
    printf( "File Identifier ....... %s\n",   ts->ts_fid    );
    printf( "Device Name ........... %s\n",   ts->ts_dvn    );
    printf( "Device Type ........... %d\n",   ts->ts_dty    );
    printf( "Label Type ............ %d\n",   ts->ts_lb     );
    printf( "Maximum Block Size .... %d\n",   ts->ts_mbs    );
    printf( "Number of VSNs ........ %d\n",   ts->ts_numvsn );

    vsnp = (char *)req.databuf + ts->ts_vsnoff;
    printf( "VSN List\n");
    for ( i=0;  i < ts->ts_numvsn;  i++, vsnp+=L_MAXVSN ) {
        printf("  %s\n", vsnp );
    }

    exit(0);
}
```

4.3.2 `mtio.h` `ioctl` Requests

TMF supports the tape `ioctl`(2) requests defined in the `/usr/include/sys/mtio.h` file for IRIX systems and the `/usr/include/tmf/mtio.h` file for Linux

systems. This file does not involve tape movement or the setting of certain attributes that must be controlled by TMF. It supports the following `mtio.h` requests:

Request	Description
`MTCAPABILITY`	Returns the device type and capabilities for a device
`MTIOCGETBLKINFO`	Returns device block size information: the minimum, maximum, current, and recommended block size
`MTIOCGET_SGI`	Returns tape status
`MTIOCGET`	Returns Linux tape status
`MTIOCGETEXT`	Returns extended tape status
`MTIOCGETEXTL`	Returns the last tape status
`MTSCI_RDLOG`	Returns statistical information maintained by a device
`MTSCISI_SENSE`	Returns the sense data from the last device command that terminated with a SCSI Check Condition or Command Terminated status
`MTSCSIINQ`	Returns device parameter information
`MTSPECOP`	Sets the block size for fixed-length I/O

4.3.2.1 `MTCAPABILITY`

The `MTCAPABILITY` `ioctl` request returns the device type and capabilities for a device. The argument to this request is a pointer to the `mt_capablity` structure defined in the `/usr/include/sys/mtio.h` file for IRIX systems and the `/usr/include/tmf/mtio.h` file for Linux systems.

On the reply, the `mt_subtype` field returns the device type. Valid device types are defined in the `/usr/include/sys/invent.h` file (IRIX systems) and the `/usr/include/tmf/invent.h` file (Linux systems) beginning with value `TPUNKNOWN`. The `mt_capablity` field returns the set of capabilities for a device. The capabilities are a set of flags, the `MTCAN*` status flags, which are also defined in the `/usr/include/sys/mtio.h` file for IRIX systems and the `/usr/include/tmf/mtio.h` file for Linux systems.

Note: The Linux location for all of the `.h` files is in `/usr/include/tmf`, rather than in `/usr/include/sys` (which is the IRIX location).

4.3.2.2 MTIOCGETBLKINFO

The MTIOCGETBLKINFO ioctl request returns device block size information: the minimum, maximum, current, and recommended block size. The argument to this request is a pointer to the mtblkinfo structure, which is defined in the /usr/include/sys/mtio.h file for IRIX systems and the /usr/include/tmf/mtio.h file for Linux systems.

Note: The Linux location for all of the .h files is in /usr/include/tmf, rather than in /usr/include/sys (which is the IRIX location).

4.3.2.3 MTIOCGET_SGI

The MTIOCGET_SGI ioctl request returns tape status. The argument to this request is a pointer to the mtget_sgi structure. The MTIOCGET_SGI ioctl returns the following values:

Field	Description
mt_type	Controller type. This value is always MT_ISSCI.
mt_dposn	Tape position status. The following state flags are in the /usr/include/sys/mtio.h file (IRIX systems) and the /usr/include/tmf/mtio.h file (Linux systems):

Flag	Description
MT_EOM	At end-of-media.
MT_BOT	At beginning-of-tape (BOT).
MT_WPROT	Write protected volume.
MT_EW	At end-of-tape.
MT_ONL	Device is online.
MT_EOD	At end of recorded data.
MT_FMK	At filemark.

`mt_dsreg`

Last device status represented by a set of flags. The device status flags are in the `/usr/include/sys/tpsc.h` file (IRIX systems) and the `/usr/include/tmf/tpsc.h` file (Linux systems):

Flag	Description
`CT_EOM`	At end-of-media.
`CT_BOT`	At beginning-of-tape (BOT).
`CT_WRP`	Write protected volume.
`CT_EW`	At end-of-tape (EOT).
`CT_GETBLKLEN`	Block length request is pending.
`CT_MOTION`	Last command moved the tape.
`CT_ONL`	Device is online.
`CT_QIC24`	Low density tape is on Viper 150 device.
`CT_QIC120`	High density tape is on Viper 150 device.
`CT_OPEN`	Tape device is open.
`CT_READ`	Last tape movement request was a `read`(2) request.
`CT_WRITE`	Last tape movement request was a `write`(2) request.
`CT_CHG`	Unit attention occurred.
`CT_DIDIO`	Tape movement request has been done since device open.
`CT_BOD`	At end of recorded data.
`CT_FM`	At filemark.

`mt_erreg` Last device error status. The device status flags are in the `/usr/include/sys/tpsc.h` file (IRIX systems) and the `/usr/include/tmf/tpsc.h` file (Linux systems):

Flag	Description
`CT_NEEDBOT`	Positioning is required before I/O.
`CT_LOADER`	Tape has been loaded.
`CT_ANSI`	I/O after end-of-tape (EOT) is permitted.
`CT_SMK`	Drive is at a setmark.
`CT_AUDIO`	Drive is in audio mode.
`CT_AUD_MED`	Media is recorded in audio format.
`CT_MULTPART`	Multi-partitioned volume is loaded.
`CT_SEEKING`	Seek request is pending.
`CT_HITFMSHORT`	Filemark was detected, but the status has not yet been returned to the user.
`CT_INCOMPAT_MEDIA`	Media is incompatible with the device.
`CT_CLEANHEADS`	Device needs to be cleaned.
`CT_COMPRESS`	Device is in compression mode.
`CT_MEDIA_ERR`	Media error has been detected.
`CT_EOD`	At end of recorded data.

`mt_resid` Residual count. This field contains the difference between what was requested and what has completed, in bytes, blocks, files, or records, depending on the command.

mt_fileno	Not supported.
mt_blkno	Block number (returned from the device).

Note: The Linux location for all of the .h files is in /usr/include/tmf, rather than in /usr/include/sys (which is the IRIX location).

4.3.2.4 MTIOCGET (Linux only)

The MTIOCGET ioctl request returns tape status on Linux systems. The argument to this request is a pointer to the Linux mtget structure. The MTIOCGET ioctl returns the following values:

Field	Description
mt_type	Controller type. This value is either MT_ISSCSI1 or MT_ISSCSI2.
mt_resid	Residual count. This field contains the difference between what was requested and what has completed, in bytes, blocks, files, or records, depending on the command.
mt_dsreg	Drive status register, containing block size and density status.
mt_gstat	Generic (device independent) status, represented by a set of flags defined in the /usr/include/linux/mtio.h file, as follows:

Flag	Description
GMT_EOF	At end-of-file.
GMT_BOT	At beginning-of-tape.
GMT_EOT	At end-of-tape.
GMT_SM	Drive is at a setmark.
GMT_EOD	At end of recorded data.
GMT_WR_PROT	Write protected volume.
GMT_ONLINE	Device is online.
GMT_D_6250	Density is 6250 bpi (0x3).

	GMT_D_1600	Density is 1600 bpi (0x2).
	GMT_D_800	Density is 800 bpi (0x1).
	GMT_DR_OPEN	Device has door open (no tape).
	GMT_IM_REP_EN	Device is in immediate report mode.
	GMT_CLN	Cleaning requested.
mt_erreg	Error register. Contains the last device error status.	
mt_fileno	File number. Contains the number of the current file on the tape.	
mt_blkno	Block number. Contains the number of the current block on the tape.	

4.3.2.5 MTIOCGETEXT

The MTIOCGETEXT ioctl returns the tape status. The argument to this request is a pointer to the mtgetext structure. The MTIOCGETEXT ioctl returns the following values:

Field	Description
mt_type	Controller type. This value is always MT_ISSCI.
mt_dposn	Tape position status. The status flags are in the /usr/include/sys/mtio.h file (IRIX systems) and the /usr/include/tmf/mtio.h file (Linux systems). See Section 4.3.2.3, page 78, for a list of the mt_dposn values.
mt_dsreg	Last device status represented by a set of flags. The device status flags are in the /usr/include/sys/tpsc.h file (IRIX systems) and the /usr/include/tmf/tpsc.h file (Linux systems). See Section 4.3.2.3, page 78, for a list of the mt_dposn values.

`mt_erreg`

Last device error status. The device status flags can be found in the `/usr/include/sys/tpsc.h` file (IRIX systems) and the `/usr/include/tmf/tpsc.h` file (Linux systems).

The `ts_erreg[0]` error status is represented by the upper 16 bits of the following status flags:

Flag	Description
`CT_NEEDBOT`	Positioning is required before I/O.
`CT_LOADER`	Tape has been loaded.
`CT_ANSI`	I/O after end-of-tape (EOT) is permitted.
`CT_SMK`	Drive is at a setmark.
`CT_AUDIO`	Drive is in audio mode.
`CT_AUD_MED`	Media is recorded in audio format.
`CT_MULTPART`	Multi-partitioned volume is loaded.
`CT_SEEKING`	Seek request is pending.
`CT_HITFMSHORT`	Filemark was detected, but the status has not yet been returned to the user.
`CT_INCOMPAT_MEDIA`	Media is incompatible with the device.
`CT_CLEANHEADS`	Device needs to be cleaned.
`CT_COMPRESS`	Device is in compression mode.
`CT_MEDIA_ERR`	Media error has been detected.
`CT_EOD`	At end of recorded data.

The `ts_erreg[1]` error status is represented by the status following flags:

Flag	Description
CT_BAD_REQT	An invalid request was issued to the device.
CT_LARGE_LBLK	The block on tape is larger than the requested read byte count.
CT_LOAD_ERR	Volume load failed.
CT_HWERR	Hardware error occurred.
CT_NOT_READY	Device is not ready.
CT_BLKLEN	Actual recorded block size differs from the set size.

The remaining `ts_erreg` statuses are undefined.

`mt_resid` Difference between what was requested and what has completed, in bytes or blocks depending on the command.

`mt_fileno` Not supported.

`mt_blkno` Block address (returned from the device).

`mt_partno` Partition number for those devices supporting partitions.

`mt_cblkno` Block number calculated by the driver. This value includes filemarks as well as blocks and becomes invalid if absolute positioning occurs.

`mt_lastreq` Last user request issued. The user request codes are defined in the `/usr/include/sys/mtio.h` file (IRIX systems) and the `/usr/include/tmf/mtio.h` file (Linux systems):

Flag	Description
MTR_READ	Specifies `read`(2) request.
MTR_WRITE	Specifies `write`(2) request.
MTR_WFM	Write filemark(s).

MTR_SRB	Skip records backward.
MTR_SRF	Skip records forward.
MTR_SFB	Skip filemarks backward, position after filemark.
MTR_SFF	Skip filemarks forward, position after filemark.
MTR_SEOD	Space to the end-of-data.
MTR_SEOM	Space to the end-of-media.
MTR_FORMAT	Specifies volume format.
MTR_PART	Position to a partition.
MTR_SSM	Skip setmarks.
MTR_WSM	Write setmarks.
MTR_MODEAUD	Enable or disable audio mode.
MTR_REW	Rewind.
MTR_ERASE	Erase from current position to the EOT.
MTR_RETEN	Specifies retention.
MTR_UNLOAD	Unload.
MTR_PABS	Position to an absolute address.
MTR_PAUDIO	Specifies audio position.
MTR_GAUDIO	Get audio position.
MTR_RDLOG	Read log.
MTR_ATTR	Set attribute.
MTR_SPOS	Set vendor specific position.
MTR_GPOS	Get vendor specific position.
MTR_LOAD	Load.

	MTR_SFBM	Skip filemarks backward, position at filemark.
	MTR_SFFM	Skip filemarks forward, position at filemark.
mt_ilimode	State of illegal length mode reporting for variable-block I/O. If mt_ilimode is set and a read(2) request is less than the size of the block on tape, an error is returned.	
mt_buffmode	State of filemark buffering. If mt_buffmode is set, filemarks are buffered.	
mt_subtype	Device type. Valid device types are defined in the /usr/include/sys/invent.h file (IRIX systems) and the /usr/include/tmf/invent.h file (Linux systems) beginning with the TPUNKNOWN value.	
mt_capability	Device capabilities. The capabilities are set of flags, the MTCAN* status flags, which are defined in the /usr/include/sys/mtio.h file (IRIX systems) and the /usr/include/tmf/mtio.h file (Linux systems).	

Note: The Linux location for all of the .h files is in /usr/include/tmf, rather than in /usr/include/sys (which is the IRIX location).

4.3.2.6 MTIOCGETEXTL

The MTIOCGETEXTL ioctl request returns the last tape status. The argument to this request is a pointer to the mtgetext structure. The MTIOCGETEXTL ioctl request returns the same information as the MTIOCGETEXT ioctl request. It differs from the MTIOCGETEXT ioctl request in that it does not seek updated status information from the device.

Note: The Linux location for all of the .h files is in /usr/include/tmf, rather than in /usr/include/sys (which is the IRIX location).

4.3.2.7 MTSCI_RDLOG

The MTSCI_RDLOG ioctl request returns statistical information maintained by a device. The argument to this request is a pointer to the mtscsi_rdlog structure. The following values are set on the request:

Field	Description
mtppc	Parameter pointer control
mtsp	Save parameters
mtpc	Page control
mtpage	Page code of the requested page
mtparam	Parameter from which to begin transferring information
mtlen	Size of the buffer to receive the device log
mtarg	Pointer to a buffer to receive the device log

For a more detailed explanation of the mtppc, mtsp, mtpage, and mtparam fields, see the device product manual for the device from which log information is required.

Note: The Linux location for all of the .h files is in /usr/include/tmf, rather than in /usr/include/sys (which is the IRIX location).

4.3.2.8 MTSCISI_SENSE

The MTSCISI_SENSE ioctl request returns the sense data from the last device command that terminated with a SCSI Check Condition or Command Terminated status. The argument to this request is a pointer to a buffer large enough to receive the sense data, at least MAX_SENSE_DATA bytes. MAX_SENSE_DATA is defined in the /usr/include/sys/tpsc.h file (IRIX systems) and the /usr/include/tmf/tpsc.h file (Linux systems). For an explanation of the sense data, see the device product manual for the device from which the sense information is required.

Note: The Linux location for all of the .h files is in /usr/include/tmf, rather than in /usr/include/sys (which is the IRIX location).

4.3.2.9 MTSCSIINQ

The MTSCSIINQ ioctl request returns device parameter information, such as the vendor identification, product identification, and revision level. The argument to this request is a pointer to a buffer large enough to receive the inquiry information, at least as large as the ct_g0ing_data_t structure, defined in the /usr/include/sys/tpsc.h file (IRIX systems) and the /usr/include/tmf/tpsc.h file (Linux systems). For an explanation of the inquiry data, see the device product manual for the device from which the device information is required.

Note: The Linux location for all of the .h files is in /usr/include/tmf, rather than in /usr/include/sys (which is the IRIX location).

4.3.2.10 MTSPECOP

The MTSPECOP ioctl request sets the block size for fixed-length I/O. You request fixed-length I/O with the -B option on the tmmnt(1) command. The argument to the MTSPECOP ioctl request is a pointer to the mtop structure. The mt_op field must be set to MTSCSI_SETFIXED and mt_count to the block size.

Note: The Linux location for all of the .h files is in /usr/include/tmf, rather than in /usr/include/sys (which is the IRIX location).

Interpreting System Messages

This appendix lists and describes the system messages, error or informative, that you may encounter while you are working with Tape Management Facility (TMF). These system messages are found in either the `tape.msg` file or your standard output file. For system messages indicating that a TMF error has occurred, contact your system administrator. For internal problems, contact your system support staff.

Each message description is followed by a label that signifies the message recipient. The following labels are used with TMF system messages:

USER	Individual (end user) using TMF to manage tapes
ADM	TMF administrator
USER/ADM	Individual (end user) or TMF administrator

`TM000 -- Tape resource(s) have been reserved for you`
Tape resources have been reserved for you by means of the `tmrsv`(1) command. – USER

`TM001 -- Unable to open` *pipe_desc* `pipe` *pathname* `(errno =` *errno*`)`
TMF was unable to open the *pathname* pipe; the error returned is *errno*. This message may indicate that TMF is not running. – ADM

`TM002 -- Unable to write to` *path_desc pathname* `(errno =` *errno*`)`
A `write`(2) request to a file or pipe of type, *path_desc*, the *pathname* path name failed; the error returned is *errno*. – ADM

`TM003 -- Resource` *group_name* `is not available`
The requested number of resources, belonging to the *group_name* resource group, are not available. – USER

`TM004 -- The` *group_name* `resource count,` *count*`, is invalid`
The amount *count* of available resources of the *group_name* type, which is maintained by TMF, is invalid. This indicates an error in TMF. – ADM

`TM005 -- Exceeded the limit on the number of tape devices which can be specified,` *limit*

The number of devices specified in the `tmrls`(1) command exceeds the maximum, *limit*, allowed. – USER

`TM006 -- An invalid request was sent to the Tape Management Facility`

An invalid request was made to the Tape Management Facility. This indicates an error in TMF. – ADM

`TM007 -- The specified device does not support volume partitioning.`

A partition number was specified on a command issued for a device which does not support volume partitions. Reissue the command without the partition number or issue the command to a device supporting partitions. – USER

`TM008 -- Should volume` *vsn* `on device` *device* `switch from` *label_type1* `to` *label_type2* `for user` *userid*`? reply y/n`

The administrator must specify whether the tape named *vsn* on *device* may be switched from *label_type1* to *label_type2* for user *userid*. The administrator replies **y** for "yes" or **n** for "no". – USER/ADM

`TM009 -- Unable to clear the abnormal status flag in the TMF driver (errno =` *errno*`)`

A TMF stream process was unable to acknowledge an error detected by the TMF driver. – ADM

`TM010 -- Exceeded the limit of` *group_name* `resources which may be used`

The use of the *group_name* resource was requested with the `tmmnt`(1) command, but the request was denied because the user is at the user's limit for resources of this type. The limit was established with the `tmrsv`(1) command, and you can obtain the current status with the `tmrst`(1) command. – USER

`TM011 -- Enter the VSN for the volume mounted on device` *device_name*

You must specify the volume identifier of the volume on the *device_name* device . – ADM

`TM012 -- Unable to obtain memory for` *variable*

TMF could not acquire memory for the *variable* variable. This indicates an error in the TMF. When this message is issued, TMF will exit. The administrator should collect the trace files for examination by software product support. – ADM

`TM013 -- Volume` *vsn* `on device` *device name* `has not expired. Reply y/n for user` *userid* `to write on tape`

You must specify whether user *userid* may write on unexpired tape *vsn*. Reply **y** for "yes" or **n** for "no". – ADM

`TM014 -- Unable to give you ownership of path` *pathname* `(errno =` *errno*`)`

TMF issued a `chown`(2) command for the *pathname* path, and the *errno* error was received. – ADM

`TM015 -- Unable to obtain a reply to the` *request* `request (errno =` *errno*`)`

A TMF command was unable to obtain a reply to the TMF *request* request; the *errno* error was received. This indicates an error in TMF. – USER

`TM016 -- The write to` *type pathname* `failed after writing only` *m* `bytes of the requested` *n* `bytes.`

A `write`(2) request to a file or pipe described by *type* to the *pathname* path did not write all of the requested bytes. Only *m* bytes of the requested *n* bytes were written. – ADM

`TM017 -- The` *option* `specified,` *string*`, cannot exceed` *count* `characters.`

The number of characters in *string* is larger than *count* and is the value of the *option* option. – USER

`TM018 -- Option` *option* `cannot be specified more than once`

The *option* option is duplicated on your command line. – USER

`TM019 -- Volume remounts because an incorrect volume was mounted are not supported for volumes in an autoloader domain`

An incorrect volume was mounted by an library. Recovery from this error is not attempted for volumes serviced by an library. – USER

`TM020 -- A file sequence number cannot be specified for scratch mount requests`

A file sequence number cannot be specified with the `-q` option of the `tmmnt`(1) command if a volume was not also specified. – USER

`TM021 -- Exceeded the maximum number of volumes allowed,` *maxvsn*

The number of volume identifiers in the volume identifier list is greater than *maxvsn*. Use fewer volume identifiers. – USER

`TM022 -- Options` *option1* `and` *option2* `are mutually exclusive`
Options *option1* and *option2* are mutually exclusive. You may use only one of them. – USER

`TM023 -- A path name must be specified`
Specify the path name by using the `-p` or `-P` option on the `tmmnt`(1) command line or as an argument to the `tmcatalog`(1) request. – USER

`TM024 -- Unable to create` *desc_path*|*filename* `(errno =` *errno*`)`
TMF was unable to create either a file, directory, or process described by *desc_path*|*filename*. If the file was specified with a command option, check to see whether you have the correct permissions for creating the file. – USER/ADM

`TM025 -- Cannot reserve resources if resources are still held from a previous reservation request`
You have issued a `tmrsv`(1) command, but you must release all previously reserved resources by using the `tmrls`(1) command. – USER

`TM026 -- Unable to communicate with TMF daemon`
A command or child process was unable to communicate with the TMF daemon. This indicates an error in TMF. – USER/ADM

`TM027 -- The expiration date specified,` *date*`, must be of the format [c|@]yyddd`
The expiration date specified is invalid. It must be of the format `[c|@]yyddd`. – USER

`TM028 -- The request to mount volume` *volume* `was canceled`
The administrator canceled your mount request for volume *vsn*. – USER

`TM029 -- All tape resources have been released`
TMF has released all tape reservations held by the user. – USER

`TM030 -- The file sequence number must be numeric or the character n or u`

The file sequence number specified is invalid. It must be numeric or **n** to indicate that a new file should be created at the end of the tape, or **u** to indicate file positioning should be based on a file name. – USER

`TM031 -- Cannot write to volume` *volume*`, file` *path_name*`, because the volume was requested with write protection enabled`
You requested the `-r out` option on the `tmmnt`(1) command and issued a write operation to the *path_name* file, but the volume mounted is write protected. – USER

`TM032 -- The file status specified must be new when requesting a file be created at the end of tape`
You requested that a file be created at the end of tape by specifying the `-q` option with value *n* on the `tmmnt`(1) command, but you did not specify that the file is new with the `-n` option. – USER

`TM033 -- Unable to lock` *desc* `file` *pathname* `(errno =` *errno*`)`
TMF was unable to lock the *pathname* file, used to communicate information between a TMF process and the TMF daemon. The error returned was *errno*. This indicates an error in TMF. – ADM

`TM034 -- Waiting for a device from group` *device_group_name*
A device from the *device_group_name* group was requested with the `tmmnt`(1) command, but it was not available. TMF has queued the request and will assign a device from the *device_group_name* group as soon as one is available. – USER

`TM036 -- The volume offset cannot exceed the number of volumes.`
The value specified on the offset option is larger than the number of volume identifiers in the volume identifier list. – USER

`TM037 -- The specified user does not have tape resources reserved`
The user specified on a forced resource release request does not have any tape resources reserved. – USER

`TM038 -- Unable to send the request` *request* `to the TMF daemon`
The TMF command is unable to send the *request* request to TMF. This may indicate that the TMF is not running. – USER

`TM039 -- Path` *pathname* `is already in use by TMF`
Another tape file called *pathname* is being used by either you or another user. – USER

`TM040 -- Unable to unload volume` *volume* `(errno =` *errno*`)`
A TMF command was unable to unload the *volume* volume. The *errno* error was returned on the unload request. – USER

`TM041 -- Unable to send an action message to the message daemon`
TMF cannot communicate with the message daemon. – ADM

`TM042 -- Unable to find a stream owned by process` *pid*
The TMF daemon received a request from a child process for the stream owned by the *pid* process. However, TMF was not able to find a stream owned by this process. This indicates an error with TMF. – ADM

`TM043 -- The` *option* `specified,` *value*`, is invalid`
The value of *value* is invalid for the *option* option. – USER

`TM044 -- Unable to open tape the` *device* `device (errno =` *errno*`)`
TMF is unable to open the tape *device* device. – USER

`TM045 -- Unable to terminate the` *process_type* `process` *name*`, pid` *pid* `(errno =` *errno*`)`
TMF was unable to terminate one of its child processes. The *errno* error was received when attempting to terminate the *name* process, which has the process identifier, *pid*.

`TM046 -- Mount volume` *vsn* `(`*label_type*`)` *ring_option* `on device` *device_name* `for` *userid* `(`*session*`)` *NQSid reason* `or reply cancel / device name`
The administrator must mount the tape with the *vsn* volume identifier, a label of *label_type*, write ring in or out, on the *device_name* device, for *userid* user with session of *NQSid*. An optional *reason* may be given. The administrator may remount the tape on the specified drive, reply with a different device name, or reply **`cancel`**. If the administrator replies **`cancel`**, the tape mount is canceled and the user cannot continue with the tape processing. – USER/ADM

`TM047 -- Remount volume` *vsn* `(`*label_type*`)` *ring_option* `on device` *device_name* `for` *userid* `(`*session*`)` *NQSid reason* `or reply cancel / device name`
The administrator must remount the tape with the *vsn* volume identifier, a label of *label_type*, write ring in or out, on the *device_name* device , for *userid* user with session of *NQSid*. An optional *reason* may be given. The administrator may remount the tape on the specified drive, reply with a different device name, or reply **`cancel`**. If the administrator replies **`cancel`**, the tape remount is canceled and the user cannot continue with the tape processing. – USER/ADM

`TM048 -- Tape stream` *pathname* `assigned or reassigned to` *device_name*
The *pathname* file is assigned or reassigned to *device_name*. – USER/ADM

`TM049 --` *pathname* `:` *vsn*`(`*label_type*`) :` *state* `: blocks =` *number*
number blocks were read or written to the *pathname* file with *vsn* and *label_type* when at the *state* state. – USER

`TM050 -- Tape device` *device_name* `has been released`
The *device_name* tape name has been released. – USER

`TM051 -- Cannot request file positioning based on a file name with non-labeled tapes`
File positioning based on a file name uses the file name found in the tape label to position the tape. It cannot, therefore, be requested with nonlabeled tapes. – USER

`TM052 -- The TMF block count,` *number1*`, does not match the label block count,` *number2*`, for file` *pathname*
The block count for a file, calculated by TMF, *number1*, does not match the block count recorded in the tape label for this file, *pathname*. This indicates an error in TMF. – USER

`TM053 -- An unexpected signal, signal` *signo*`, was received by process` *process*
The *process* process received an unexpected signal (signal number *signo*). This indicates an error in TMF. – ADM

`TM054 -- Device` *device name* `has not been configured in the Tape Management Facility`
An invalid device name was specified on a TMF command. – USER

`TM055 -- An invalid device group name,` *group*`, was specified`
An invalid device group name was specified on a TMF command. – USER

`TM056 -- Tape resource` *group* `has not been reserved`
Either the device group name on the `tmmnt`(1) command does not match the device group name you used on the `tmrsv`(1) command, or you have not issued a `tmrsv`(1) command. – USER

`TM057 -- The path name specified for release,` *pathname*`, is not associated with a tape resource`

pathname, which was used in the `tmrls`(1) command, was not mounted with a `tmmnt`(1) command. – USER

`TM058 -- Command` *command* `was interrupted by signal` *signo*
The *command* (`tmrsv`(1) or `tmmnt`(1)) command has been interrupted by the *signo* signal. – USER

`TM059 -- Permission to write to volume` *vsn* `was denied`
Permission to write to the *vsn* volume was denied by the front end. – USER

`TM060 -- Waiting for device` *device_name*
The *device_name* device , which was requested with the `tmmnt`(1) command, was not available. TMF has queued the request and will assign the device as soon as it becomes available. – USER

`TM061 -- Permission to update directory` *dir* `is denied`
You cannot update the *dir* directory because you do not have write permission in the directory. – USER

`TM062 -- File` *pathname* `on volume,` *vsn*`, is protected`
The *pathname* file found on the *vsn* volume is volume protected. See the system administrator. – USER

`TM063 -- File` *pathname* `on volume` *vsn* `is not of the requested label type` *label_type*
The *pathname* file found on the *vsn* volume has an incorrect label type. Check your tape. – USER

`TM064 -- File` *filename* `could not be found on volume` *vsn*
The *vsn* volume identifier does not contain the specified file, *filename*. Check your tape. – USER

`TM065 -- File` *filename* `on volume` *vsn* `has not expired`
The *vsn* volume identifier does not contain the specified file, *filename*, in an expired state. – USER

`TM066 -- A volume must be mounted on the specified device`
The device specified with the TMF `tmlabel`(8) command must have a volume mounted. – ADM

`TM067 -- A tape volume must be specified for file` *filename*
There is no VSN list for the *filename* file. – USER

`TM068 -- Cannot request user label processing when creating a new file or appending to an existing file`
User label processing is only valid when accessing old files. – USER.

`TM069 -- An invalid request,` *request*`, was received from the TMF driver`
An invalid request was received from the driver. This indicates an error in TMF. – ADM

`TM070 -- Unable to complete the` *request* `request issued to device` *device* `(errno =` *errno*`)`
The *errno* error was encountered when performing the *request* request on the *device* device. – USER

`TM071 -- Unable to complete the` *request* `request issued to device` *device* `for file` *pathname* `(errno =` *errno*`)`
The *errno* error was encountered when performing the *request* request on the *device* device for the *pathname* file. – USER

`TM075 -- Cannot allow user label processing on a volume which is not write protected`
User label processing is valid only on write protected volumes. – USER

`TM076 -- An invalid label structure was detected on file` *filename* `on volume` *vsn*
The *vsn* volume containing the *filename* file has an invalid label structure. – USER

`TM077 -- A user and session identifier must be specified`
A user and session identifier must be specified on the `tmfrls`(1) command. – USER

`TM078 -- The Tape Management Facility is not available`
The TMF daemon is stopped. Either a `tmstop`(8) command has been issued, or an error has occurred. – USER/ADM

`TM079 -- An invalid reply was received from the loader daemon`

The reply to a loader request is invalid. TMF processing for this user will be terminated. – USER

`TM080 -- The` *message* `request failed because this message type is not supported by the front-end`
A message request sent to a front end was not accepted because the front end does not support messages of this type. – USER

`TM081 -- The file sequence number specified,` *number*`, for file` *filename*`, exceeds the number of files on the specified volume set`
The file indicated by the *number* file sequence number is not on the tape. – USER

`TM082 -- The default tape resource,` *resource*`, has not been reserved`
A request was made with the `tmmnt`(1) command to mount a volume on a device belonging to the default resource group. The request failed because the default tape resource, *resource,* had not been reserved with the `tmrsv`(1) command. – USER

`TM083 -- An invalid request,` *code*`, was received from a TMF child process`
The *code* request code is invalid. This indicates an error in TMF. – ADM

`TM084 -- Tape Management Facility internal error`
TMF returned an error indicating an internal TMF error. – USER/ADM

`TM085 -- The specified volume does not exist in the volume identifier list for this stream`
The volume identifier specified in a TMF request could not be found in the volume identifier list specified on the `tmmnt`(1) command for this stream. – USER

`TM086 -- Error` *error* `was encountered by the Tape Management Facility`
TMF returned the *error* error. – USER

`TM087 -- An invalid range,` *value1*`-`*value2*`, was specified`
The range specified is incorrect. *value1* exceeds *value2*. – ADM

`TM088 -- The path name specified,` *pathname*`, already exists`
You specified *pathname* on the `tmmnt`(1) command by using the `-p` option, and *pathname* exists. The `-p` option of the `tmmnt`(1) command does not delete the *pathname* file if it exists. You can either delete the *pathname* file or use the `-P` option. – USER

`TM089 -- Directory` *pathname* `was specified for the` *desc* `when a file name was expected`
You specified *pathname* on the TMF command rather than the expected file name; *pathname* is a directory. – USER

`TM090 -- Because the environment variable,` *variable*`, is not set, the current working directory will be used instead for the` *file_type*
It was requested that the `variable` environment variable be used by TMF command, but it is not set up correctly. – USER

`TM091 -- The path name of the` *desc* `cannot exceed` *number* `characters.`
The path name specified is larger than the maximum of *number* characters accepted by TMF. See the system administrator. – USER

`TM092 -- Unable to get the current working directory (errno =` *errno*`)`
TMF cannot get your current working directory. The `errno` is *errno*. – USER

`TM093 -- The` *desc* `specified must be either 'on' or 'off'`
The state specified for *desc* must be either **on** or **off**. – ADM

`TM094 -- Waiting for another request to complete`
The `tmmnt`(1) command was postponed to finish another request. – USER

`TM095 -- The limit on the number of tape users has been reached.`
The maximum number of tape users was exceeded. – USER

`TM096 -- The label type specified on multifile volume mount requests must match the type specified on previous mount requests for the volume`
The label type specified on a multifile volume mount request is incorrect. It must be identical to the type specified on the previous multifile volume mount requests. – USER

`TM097 -- The following tape users are deadlocked`
The following users are deadlocked during device allocation. – ADM

`TM098 -- Waiting for device` *device* `to avoid a possible deadlock`

Assigning the *device* device could lead to a possible system deadlock; therefore, allocation is delayed. – USER

`TM099 -- The tape stream open request specifies a file unknown to the TMF daemon`

The tape stream path name was not known to TMF when it processed the request code sent by the user. – USER

`TM100 -- The write of the` *request_type* `request to` *file type pathname* `failed after writing only` *number1* `bytes of the requested` *number2* `bytes`

The *pathname* file is used to communicate a request of the *request_type* type to a front end. TMF was unable to write the request to this temporary file. *number1* bytes were written instead of the expected *number2* bytes. – USER

`TM101 -- A device release is pending`

When TMF was processing a release request, the device could not be released immediately. It will be released as soon as possible. – USER

`TM102 -- Waiting for a resource release to complete`

TMF received your reserve request and is waiting for the release of devices from a previous release request. TMF delays the processing of your reserve request until all pending releases are completed. – USER

`TM103 -- Cannot mount a bypass label tape with the ring out`

TMF is installed with the option that `ring out` must be used when the `blp` label type is used. – USER

`TM104 -- Operator replied :` *reply_string*

The operator replied *reply_string* to an operator message about your tape. – USER

`TM105 -- Cannot send the` *request_type* `request because front-end` *fes* `is not logged on`

The *request_type* request, sent to the *fes* front end could not be processed because the *fes* front end is not logged on. – USER

`TM106 -- The` *request_type* `request failed because the request was created incorrectly`

An error was returned by the front end for the *request_type* request because TMF created the request incorrectly. – USER

`TM107 -- Unable to create a Tape Management Facility daemon (errno =` *errno*`)`
TMF could not be brought up successfully because a new session for TMF could not be created. – ADM

`TM108 -- The` *request_type* `request failed because of an unknown condition` *error*
The *request_type* request, which was sent to a front end, failed because of the *error* error. – USER

`TM110 -- Unable to register for signal` *signo* `(errno =` *errno*`)`
Registration for the *signo* signal could not be completed; the error returned is *errno*. – ADM

`TM111 -- Permission to write filemarks denied`
You attempted to read or write a filemark without using the `-T` option of the `tmmnt`(1) command. – USER

`TM112 -- Option` *option1* `requires option` *option2*
You must specify *option2* along with *option1* on the TMF command. – USER

`TM113 -- A` *label* `label was not found for file` *pathname*`, on volume` *vsn*
Your *pathname* tape file either does not have a valid *label* label or is missing this label for *vsn*. – USER

`TM114 -- Device` *device* `is already in use`
The device specified on the TMF command is already in use. – USER

`TM115 -- User end-of-volume processing has been` *action* `for tape file` *pathname*
This informational message is sent to your `tape.msg` file. *action* can be set to selected `on` or deselected `off` for *pathname* during a `TR_EOV` request. – USER

`TM116 -- Unable to set the block size for label I/O (errno =` *errno*`)`
TMF is unable to label a tape volume because it cannot set the block size to the label size; the error received is *errno*. – USER

`TM117 -- Permission to (re)catalog file` *filename* `was denied from the front end`

The servicing front end did not allow you to catalog or recatalog the *filename* file. – USER

`TM118 -- Permission to delete file` *filename* `from the front-end catalog was denied by the front-end.`

The servicing front end did not allow you to delete the *filename* file from the front-end catalog. – USER

`TM119 -- A request to position to a specific volume must specify a volume name`

TMF user request, `TR_PVSN`, did not include a volume name. – USER

`TM120 -- The storage server was unable to complete the` *request_type* `request`

The storage server encountered an error while attempting to complete the *request_type* request. – USER

`TM121 -- The record format specified must be one of the characters F, D, U, or V`

The record format specified with the `-F` option of the `tmmnt`(1) command is invalid. It must be one of the characters, F, D, U, or V. – USER

`TM122 -- Stream` *pathname* `is busy`

The stream specified for release with the `-R` option of the `tmmnt`(1) command cannot be released because the stream file is open. – USER/ADM

`TM123 -- Cannot request an unload of an assigned device,` *device*

You requested an unload of the *device* device with the `tmunld`(8) command. Your request failed because the device is assigned to a user. – USER

`TM124 -- AVR not active`

You tried to use the `tmuld`(8) command, but automatic volume recognition (AVR) is not active. – ADM

`TM125 -- Cannot issue an` *request_type* `request to a downed device,` *device*

You requested an unload of the *device* device with the `tmunld`(8) command. The request failed because the device is not configured up. – USER

`TM126 -- Cannot modify` *feature* `mode, feature in use`
You used the `tmset`(1) command to change a tape daemon option, but the tape subsystem is busy. The options (*feature*) available for type are automatic volume recognition (AVR), front-end servicing (FES), destination of tape operator messages, and tracing. – ADM

`TM128 -- The` *request_type* `failed because station message processing is not enabled`
The *request_type* request sent to a front end, failed because station message processing is not enabled. – USER

`TM129 -- An invalid TMF request,` *request* `was issued for file` *pathname*
You issued an invalid request, *request*, to TMF with the `TMFC_DMNREQ ioctl` system call for the *pathname* file. Check the `/usr/include/tmf/sys/tmfctl.h` include file for a list of valid requests. – USER

`TM130 -- File` *pathname* `is new and cannot, therefore, be read`
You issued a `read`(2) request to a file, *pathname*, which was accessed as new on the `tmmnt`(1) command. – USER

`TM131 -- Unable to send request` *request_type*`: error` *reason*
TMF received an error when it tried to send a station message to a servicing front end. *reason* was received as the reason for the error. – USER

`TM132 -- Permission to access file` *file_id* `denied by front-end` *feid*
The servicing front end, *feid*, denied access to the *file_id* file. – USER

`TM133 -- The` *request_type* `request cannot be issued to a front-end which is not secure`
The servicing front end is not secure. – USER

`TM134 -- File` *file_id* `cannot be cataloged because it already exists in the catalog`
The servicing front end returned an error because *file id* was specified as a new file, and it already exists in the catalog. – USER

`TM135 -- File` *file_id* `does not exist in the catalog`
The servicing front end returned an error because *file_id* was specified as an existing file and it does not exist in the catalog. – USER

`TM136 -- The data attribute specified must be one of the characters B, S, or R`
The data attribute portion of the record format field specified with the `-F` option of the `tmmnt`(1) command is invalid. It must be one of the following characters: B, S, or R – USER

`TM137 -- Unable to update file` *file_id* `in the front-end catalog`
The servicing front end returned an error because the catalog update failed. – USER

`TM138 -- Access to volume` *volume* `was denied`
The servicing front end denied access to the *volume* volume. – USER

`TM139 -- The requested volume,` *volume*`, is not in the volume catalog`
The servicing front end returned an error because the *volume* volume does not exist in the volume catalog. – USER

`TM140 -- Unable to update volume` *volume* `in the volume catalog`
The servicing front end returned an error because the volume catalog update failed. – USER

`TM141 -- Unable to create the front-end message requesting a` *request_type*
An error occurred during an attempt to build a station message. Contact your system support staff – USER/ADM

`TM142 -- An invalid station message type,` *request_type*`, was specified`
This is an invalid station message type. Contact your system support staff. – USER/ADM

`TM143 -- The size of the station message specified,` *size*`, differs from the size of the data written to the station message file`
The text of this station message is not of the size expected. Contact your system support staff. – USER/ADM

`TM144 -- Unable to read the station message reply header (errno =` *errno*`)`
An error occurred while a station message reply was read from a front–end. Contact your system support staff. – ADM

`TM145 -- Unable to read from` *desc* `pipe` *pathname* `(errno =` *errno*`)`
TMF was unable to read data from the *pathname* pipe; the error returned is *errno*. – ADM

`TM146 -- The size of the` *type* `read,` *bytes read* `bytes, differs from the expected size of` *bytes* `bytes`
A `read`(2) operation returned a count different from that expected. – USER

`TM147 -- The identifier specified for the front-end request,` *id*`, is invalid`
Front-end identifier*id* is invalid. – USER

`TM148 -- Unable to send a request to the front-end`
An error occurred during an attempt to send a station message. – USER

`TM149 -- The request was rejected by the front-end`
The servicing front end rejected your request. – USER

`TM150 -- The station message reply does not include the required table` *table*
Required table, *table,* is missing from a station message reply. Contact your system support staff. – USER/ADM

`TM151 -- A catalog request cannot be issued when front-end servicing is disabled`
Front-end servicing is turned off. – USER

`TM152 -- The last device within a range of devices was not specified`
The format of the device range is invalid. An upper range is required. – ADM

`TM153 -- The no unload option can only be specified when configuring a device up`
You specified the no-unload option on a request to configure a device down. – ADM

`TM154 -- A device configuration request must specify the device name(s) and state`
An invalid device configuration request was issued. The device configuration request requires a device name and state. – ADM

`TM155 -- The front-end did not reply to the` *request_type* `request`
Timed out while waiting for a reply from the servicing front end. – USER

`TM156 -- Front-end,` *feid*`, configured for the` *request_type* `request, is not enabled for station messages`
The specified servicing front end does not accept type 3 station messages. – USER

`TM157 -- The specified path,` *pathname*`, must be closed when issuing a catalog request`
The `tmcatalog`(1) command can be used only when the file is closed. – USER

`TM158 -- Unable to send` *message* `for` *user job_id*`,` *NQSid* `to front-end` *target_id*`(`*reason*`). Reply cancel or retry`
The *message* message to server or front-end identifier cannot be sent for *user* because *reason*. – ADM

`TM159 -- Option -`*option* `cannot be specified when front-end servicing is disabled.`
Front-end servicing must be enabled before requesting a front-end operation with the -*option* option. – USER

`TM160 -- User volume` *vsn* `closed`
The *vsn* volume identifier was closed. – USER

`TM161 -- Unable to open` *desc* `file` *filename* `(errno =` *errno*`)`

The *filename* file, could not be opened; the error returned is *errno*. Check to see if this file exists or whether you have the correct permissions for opening the *filename* file. – USER/ADM

`TM162 -- Extra parameters were specified at the end of command,` *string*
Extra characters (*string*) were specified in the TMF command. – USER

`TM163 --` *program_name* `(pid` *process_id*`): server` *server_name*`:` *error_text*`. Reply retry or cancel`
program_name encountered a problem communicating with the *server_name* server on behalf of one or more requests. The problem is described in *error_text*. A reply of **`cancel`** aborts all requests that have encountered this problem. A reply of **`retry`**

requeues all requests that encountered this problem and allocates more time for these requests to wait for the communication with the *server_name* server. – ADM

`TM164 -- Unable to read` *type* `file` *filename* `(errno =` *errno*`)`
An error occurred when attempting to read the *filename* file; the error returned is *errno*. Check to see if this file exists or whether you have the correct permissions for reading the *filename* file. – USER/ADM

`TM165 -- Either a loader status or a device list must be specified with option -m`
An invalid loader configuration request was issued via the `tmconfig`(8) command. When you specify the `-m` option, you must also specify a loader status or a device list or range. – ADM

`TM166 -- An invalid loader daemon request was created`
TMF did not correctly create a loader daemon request. – USER

`TM167 -- The ring status specified,` *status*`, must be either 'in' or 'out'`
An invalid ring status, *status*, was specified with the `-r` option of the `tmmnt`(1) command. The status must be `in` or `out`. – USER

`TM168 -- The function code specified on a loader daemon request,` *code*`, is not supported`
TMF created a loader daemon request incorrectly. The loader function specified, *code*, is not supported. – USER

`TM169 -- Unable to change the owner and group of` *file_type pathname* `(errno =` *errno*`)`
TMF was unable to change the owner or group of the *pathname* file; the error returned is *errno*. – ADM

`TM170 -- File` *pathname* `must be closed because of an unrecoverable error (errno =` *errno*`)`
The user received an error on a tape request. No further requests will be accepted except for a close request. Close and reopen the tape file. – USER

`TM171 -- User reads and writes of file marks cannot be performed on single tape mark format tape`
Single filemark format tapes are not allowed with the `-T` option. – USER

`TM176 -- Because multiple files are not supported for single tape mark format tapes, a file sequence number greater than one cannot be specified`

Single filemark format tapes are not allowed with the `-q` option of the `tmmnt`(1) command. – USER

`TM177 -- Unable to create a temporary file needed to communicate the` *request* `request to the TMF daemon (errno =` *errno*`)`

A TMF request was terminated because TMF was unable to create a file required to communicate the request to the TMF daemon; the error returned is *errno*. – USER/ADM

`TM178 -- Unable to provide the` *request* `request information to the TMF daemon indirectly`

A TMF request was terminated because TMF was unable to communicate the request to the TMF daemon; the error returned is *errno*. – USER/ADM

`TM179 -- An operator request cannot be issued for a device controlled by an unattended media loader`

With the tape loader in unattended mode, administrator requests are not valid; thus the request was aborted. – ADM

`TM180 -- Cannot write after a read`

`write`(2) requests cannot directly follow `read`(2) requests. – USER

`TM181 -- The specified` *path_type* `path,` *pathname*`, does not exist`

pathname specified with the TMF command does not exist. – USER

`TM182 -- Permission to read file filename was denied (errno =` *errno*`)`

The `filename` file does not have read access enabled. – USER

`TM183 -- Cannot request concatenation when creating a new file or appending to an existing file`

The `-c` option of the `tmmnt`(1) command was used to request that multiple tape files be read as though they were one tape file. This feature cannot be specified with either the `-n` or `-a` option. Correct the option specified and reissue the `tmmnt`(1) command. – USER

`TM184 -- Cannot issue a positioning request, other than a rewind request, to a concatenated file`
The position request was terminated because the tape file is a concatenated file. The only valid positioning request for concatenated files is the rewind request. – USER

`TM185 -- Session` *sid*`terminated but left stream file` *pathname* `open`
TMF allowed a stream file to remain open after the session owning the stream has terminated. – ADM

`TM186 -- Cannot read after a write`
`read`(2) requests cannot directly follow `write`(2) requests. – USER

`TM187 -- Cannot position past the beginning of file` *file*
A request to position backward by blocks was terminated because the beginning of the file was detected. – USER

`TM188 -- Cannot position past the end of file` *file*
A request to position forward by blocks was terminated because the end of the file was detected. – USER

`TM190 -- **WARNING** device` *device_name* `(autoloader:` *library_name*`, server:` *server_name*`) is in state:` *state*`.`
During the initialization of the *server_name* server for the *library_name* library, the *device_name* tape drive was reported by the server to be in the *state* state. Either check the state of the tape drive with the server and alter its state so that it can be used on the system or do not attempt to configured it up. – ADM

`TM191 -- Cannot issue a catalog request to a file,` *pathname*`, which has not been accessed`
The `tmcatalog`(1) command can be used only after the file has been opened and closed. – USER

`TM192 -- A servicing front-end has not been defined`
The `tmcatalog`(1) command cannot be used if a servicing front end is not being used. – USER

`TM193 -- Permission to` *operator* `file` *file* `denied by the front-end`
The servicing front end has not given permission to perform the specified operation. – USER

`TM194 -- The operator requested that the request be retried`
The message is being sent again to the front-end operator. – USER

`TM195 -- Unable to configure media loader` *loader* `to auto mode`
The *loader* media loader could not be modified to auto mode. – ADM

`TM196 -- Unable to configure media loader` *loader* `to manual mode`
The *loader* media loader could not be modified to manual mode. – ADM

`TM197 -- Unable to configure media loader` *loader* `to attended mode`
The *loader* media loader could not be modified to attended mode. – USER

`TM200 -- The device status specified,` *status*`, must be either 'up' or 'down'`
The status specified on a device configuration request must be either up or down. – ADM

`TM201 -- A hardware component must be specified`
Either a device name, loader name, or group name must be specified on a configuration request. – ADM

`TM202 -- The default file sequence number can be used for only one of the multi-file mount requests for a volume.`
Because unique files must be specified on multifile volume access requests, the default file sequence number can only be used by one of the mount requests. – USER

`TM203 -- The request to disable user end-of-volume processing was ignored because it is not enabled`
A request to deselect user end-of-volume processing was ignored because user end-of-volume processing is not currently selected. – USER

`TM204 -- The request to configure component name was ignored because another request is pending`
A `tmconfig`(8) command cannot be completed because a previous configuration command is still pending. – ADM

`TM205 -- The default file sequence number can be used for only one of the multi-file mount requests for a volume`

Multiple `tmmnt`(1) commands were entered without specifying the file sequence numbers. The default is 1, which indicates the first file on the tape. Reenter the commands using file sequence numbers for the `-q` option. – USER/ADM

`TM206 -- An invalid message type,` *type*`, was specified in the FES request`
A TMF child process issued a request to the TMF daemon to update file information for the *file* file, which could not be found. This indicates an error with TMF. – ADM

`TM209 -- Permission to write to file` *pathname* `denied`
You have attempted to write to a file that does not have `write` permission. The `write`(2) request has been aborted. – USER

`TM210 -- Media loader` *name* `has not been configured in the Tape Management Facility`
An invalid media loader *name* was specified. – USER

`TM211 -- A device list must be specified when requesting a device group reassignment with option -`*option*
An invalid device group reassignment request was issued. This request requires a device, device list, or device range. – USER

`TM212 -- The communication path,` *code*`, to loader,` *name*`, is invalid`
An invalid communication path was specified when a loader was defined in the configuration or parameter file. – ADM

`TM213 -- Media loader,` *name*`, is unable to change the ring status for volume` *vsn*
A tape volume in the loader does not have the correct ring status for the *vsn* volume. The loader is unable to correct the ring status. The tape request has been aborted. – USER

`TM214 -- A stream ordinal must be specified`
The TMF command required a stream ordinal. – USER

`TM215 -- Unable to obtain status information for loader` *name*
You requested status information for loader *name*. TMF was unable to obtain this information. – USER/ADM

`TM216 -- Unable to send` *msgtype* `for` *user sid*`,` *NQSid* `to front-end` *feid* (*reason*). `Reply cancel, retry, or ignore`

The TMF daemon was unable to send a message to the front end. The administrator should reply with **`cancel`** to abort the original request, **`retry`** to reissue the original request, or **`ignore`** to ignore the error condition. Multiple messages are generated when a tape is being mounted. If the administrator specifies **`ignore`**, it is assumed that the tape mount request will be satisfied as a result of one of the other messages issued. – ADM

`TM217 -- Scratch volume request denied.`

A request was made to mount a scratch tape to a loader that does not support the type of scratch tape specified in the request. The mount request has been terminated. – USER

`TM218 -- Error in file` *config_file* `, line` *line_number*`, offset` *offset* `with option` *option*`.`

A unique file name must be specified for the *option* option. The *pathname* file has already been specified for another file definition. – USER

`TM219 -- Cannot modify the status of loader` *loader* `because there are devices allocated to this loader which are assigned`

A request to change the configuration of a loader cannot be completed if devices allocated to the loader have been assigned. The request has been terminated. – USER

`TM220 -- The media loader for device,` *device*`, cannot be modified when the device is configured up`

The media loader for a device cannot be changed if the loader is not configured down. The `tmconfig`(8) request has been terminated. Configure the loader down, and reissue the loader change request. – USER

`TM221 -- Device` *device* `cannot be configured up when the loader for the device is down`

You attempted to change or configure up the media loader for a device, and the loader is not configured down. The `tmconfig`(8) request has been terminated. – USER

`TM222 -- Unable to notify the TMF driver of the maximum block size,` *size* `(errno =` *errno*`)`

TMF was unable to notify the TMF driver of the I/O type and maximum block size for a user when processing a user's open request; the error returned is *errno*. The user's open request was terminated. – ADM

`TM223 -- The devices belonging to device group,` *group* `are not all of the same device type`

The configuration file is invalid. All devices belonging to the *group* group must be the same device type. – ADM

`TM224 -- Volume remounts because of an incorrect label type are not supported for volumes in an autoloader domain`

A volume mounted by an library does not have the correct label type. Retrying the mount in an attempt to get a volume with the correct label type mounted is not supported for volumes controlled by an library. – USER

`TM225 -- The loader state specified,` *state*`, is invalid for loader` *loader*`.`

A request was made to configure a loader to a state that is invalid. The request has been terminated. – USER

`TM226 -- The` *type* `volume name specified,` *vsn*`, must consist of all alphanumeric characters`

An invalid volume was specified on the `tmmnt`(1) command. The volume specification must be alphanumeric. – USER

`TM227 -- The storage server was unable to mount the requested volume`

A mount request issued to a storage server could not be completed. – USER

`TM228 -- Waiting for volume` *vsn*

A request was made to mount the *vsn* volume. However, this volume is currently in use. TMF will place the `mount`(1m) request in a waiting state, and reissue the request when the volume is free. – USER

`TM229 -- Permission to mount the requested volume(s) was denied by the user exit.`

Your `mount`(1m) request was terminated because of your site's verification specifications. Check to see that you have permission to mount this volume. – USER

`TM230 --` *action* `<` *reason* `>? Reply y(yes), n(no), or q(requeue).`

The message requests that you import or export a volume. Reply **y** if you wish to import or export, **n** to abort the job that requested the volume, or **q** to queue the `mount`(1m) request. – ADM

`TM231 -- Reply y/n when import is complete.`

You replied **y** to an import or export request. When you complete the import or export, reply **y** to this message. – ADM

`TM232 -- Should volume` *vsn* `be ejected from loader` *loader* `to change the ring status? Reply y(yes) or n(no)`

The administrator must respond either **y** if the system is to eject the specified *vsn* and change the state of the tape ring before returning the tape to the loader domain or **n** if the administrator does not want these actions to occur. – ADM

`TM233 -- Volume` *vsn* `is not scratchable. Reply 'retry' or 'cancel'`

The mounted volume cannot be made into a scratch tape. Reply **retry** to mount another scratch tape or **cancel** to cancel the tape mount request. – ADM

`TM234 -- Embedded filemarks are not allowed on volume` *vsn*

Embedded filemarks are not allowed on the *vsn* volume. – USER

`TM235 -- Unable to get the pending stream request from the TMF driver (errno = errno)`

A user request is pending but the TMF daemon is unable to obtain the request from the TMF driver; the error returned is *errno*. – ADM

`TM236 -- Data compression is not supported for devices belonging to group,` *group*

Data compression was requested with the `-i` option of the `tmmnt`(1) command with a device group that does not support data compression. – USER

`TM237 -- The density value specified is invalid for devices belonging to group,` *group*

A density was specified with the `-d` option of the `tmmnt`(1) command with a device group that does not support this density. – USER

`TM238 -- Is volume` *vsn* `on device` *device* `a valid scratch volume for user` *userid*`, session` *sid*`? Reply y/n`

Request that the administrator verify the scratch volume mounted. Reply **y** if the user is permitted to use the specified volume as a scratch volume or **n** to deny the request. – ADM

`TM239 -- Error in file` *config_file*`, line` *line_number*`, offset` *offset*`, with option` *option*`. File` *pathname* `cannot include any device type suffixes`

The tape file specified for a device configuration contains a device type suffix. Eliminate the suffix from the definition. – ADM

`TM240 -- Cannot find host entry for sd=`*socket_descriptor*

The entry for *socket_descriptor* is set, indicating that it is expecting a reply from another system. No entry has been queued within the `tcpnet()` function expecting such a reply. Check the network for proper functionality or contact your system support staff. – ADM

`TM241 -- Error from socket` *operation*`, sd=`*socket_descriptor*`, host=`*host*`, rc=`*rc*`, errno=`*errno*`:` *error_description*

The *error_description* error has occurred while TMF was trying to do *operation* on *socket_descriptor* that is connected to *host*. Check the network or system *host* for proper functionality or contact your system support staff. – ADM

`TM242 -- Host name` *host* `not found`

An attempt to obtain the network host entry for system *host* from the `/etc/hosts` file or from the `/etc/host.bin` file has failed. Contact your system support staff to correct the network files or to correct the tape configuration file. – ADM

`TM243 -- Unable to modify the file status flags for pipe` *pipename* `(errno =` *errno*`)`

A file status setting required for TMF communication could not be set; the error returned is *errno*. – ADM

`TM244 -- The label write did not complete successfully`

The `tmlabel`(8) command was unable to complete a label write. Only part of the label could be written. – ADM/USER

`TM245 -- Unable to create a pipe required to receive the` *request* `reply`

The *request* request could not be completed because TMF could not create a pipe to receive the reply to the request. – USER

`TM252 -- Waiting for a device from group` *group* `to avoid a possible deadlock`

The request could lead to a possible system deadlock; so the device allocation is delayed. – USER

`TM253 -- Device` *device* `is not available`
You requested a specific device, but this device is not available. – USER/ADM

`TM255 -- Cannot use more tape resources than were reserved`
A required option, *option*, has not been specified. Reissue the command specifying this option. – USER/ADM

`TM257 -- Supplementary logfile message from` *request*`:`
This message issues a supplementary log file message from the front end. The message corresponds to the front-end request. – USER

`TM258 -- Unable to read the` *data* `from` *pipename* `pipe` *pathname* `(errno =` *errno*`)`
A child process was unable to obtain *data* from its parent. An error occurred when reading from the *pipename* pipe; the error returned is *errno*.– ADM

`TM259 -- The TMF device assigned,` *device_name*`, is being used by a non-TMF process`
The fact that a device configured as part of TMF has been opened by a process that was not supported by TMF has prevented TMF from accessing the device. – ADM

`TM260 -- Volume remounts to retry a failed label write is not supported for volumes in an autoloader domain`
TMF has attempted a label a volume, but the label write request failed. Recovery is not possible for volumes managed by an library. – USER/ADM

`TM261 -- The expected size,` *size1*`, of data element` *element*`, differs from the actual size,` *size2*
A TMF child process was unable to correctly read data from its parent. The size of the data expected for one element differs from what is expected. Check that the version of the parent is identical to that of the child. – ADM

`TM262 -- Unable to obtain the block information from device` *device* `for file` *pathname* `(errno =` *errno*`)`
A user's request was terminated because TMF was unable to obtain the block size used for fixed-block I/O. – ADM

`TM263 -- Variable block I/O is not supported for devices belonging to group,` *group*

You requested variable–block I/O via the `tmmnt`(1) command with a device group that does not support variable–block I/O. – USER

`TM264 -- Unable to acknowledge the stream request (errno =` *errno*`)`

An error was returned on a system call request because a TMF process was unable to communicate correctly with the TMF driver. The process could not acknowledge that the request was received; the error returned was *errno*. – ADM

`TM265 -- Unable to obtain the device status for device` *device* `for file` *file* `(errno =` *errno*`)`

Processing for a stream terminated because TMF was unable to obtain the status of a device following a device request; the error returned is *errno*. – ADM

`TM266 -- Unable to notify the TMF driver of the stream owner,` *owner* `(errno =` *errno*`)`

A mount request terminated because the child created to process the mount request could not communicate the stream ownership information to the TMF driver. – ADM

`TM267 -- Unable to open device` *device* `for the stream` *stream*`, file` *pathname*`(errno =` *errno*`)`

An error was returned while assigning the *device* device to the *stream* stream. TMF was unable to open the device for this stream; the error returned was *errno*. – ADM

`TM268 -- Unable to` *disable or enable* `writes past the EOT for file` *pathname* `(errno =` *errno*`)`

By default, data cannot be output after the end-of-tape has been detected. A request has to be made to enable this capability. TMF encountered an error on a request to reset this capability to the default state or on a request to enable this capability; the error returned is *errno*. – ADM

`TM269 -- A device range and device list cannot both be specified`

The format of a configuration request is invalid. – USER

`TM271 -- Unable to free device` *device* `from stream` *stream* `for file` *file* `(errno =` *errno*`)`

TMF is unable to free file. – ADM

`TM272 -- Unable to provide the configuration to the TMF driver (errno =` *errno*`)`

During TMF start-up processing, the TMF daemon was unable to notify the TMF driver of the configuration, and startup was terminated; the error returned is *errno*. – ADM

`TM273 -- Front-end servicing must be enabled`

A front end was defined as the mount message destination for the device chosen for the volume mount, but front-end servicing has not been enabled. Enable front-end servicing with the `tmset`(8) command or modify the mount message destination. – USER

`TM274 -- Unable to clear stream` *stream_number* `(errno =` *errno*`)`

TMF was unable to clear the *stream_number* stream; the error returned is *errno*. – USER/ADM

`TM275 -- Cannot export volume` *vsn* `from loader` *loader*

The requested *vsn* volume was not found in the domain of the preferred *loader* loader. The volume could not be exported to the *loader* loader because either the loader is unattended or the administrator denied the request to export the volume. – USER

`TM276 -- The system indicated that a TMF request was pending for stream` *stream* `but no request was found (errno =` *errno*`)`

Communication between the TMF daemon and a TMF child process failed. It was indicated that a TMF daemon request was pending, but no request was found. Processing for the *stream* stream is terminated. – ADM

`TM277 -- An invalid SSP request was received`

A TMF child process received an invalid request from the TMF daemon. Processing for the stream is terminated. – ADM

`TM279 -- The` *parameter* `specified,` *param*`, must be in the range` *min_value* `to` *max_value*

The user specified a value for *parameter* that is invalid. The value must in the range *min_value* to *max_value*. Correct the number specified and reissue the command. – ADM

`TM280 -- Terminating the user owning path` *pathname* `and using device` *device*

When processing has terminated for a stream, the user's processing is then killed. – USER

`TM281 -- A volume identifier must be specified`

The user did not specify any VSN with option `-v` on the TMF command. Correct the `-v` parameter and reissue the command. – USER

`TM282 -- An internal volume id must be specified`

The user specified an external VSN without also specifying an internal VSN (`-v =EXTID`) on a TMF command. Correct the VSN specified and reissue the command. – USER

`TM285 -- Cannot specify a negative or non-numeric value` *parameter* `for the` *partition_number*

The user specified a nonnumeric value, *value*, for a partition number, *partition_number*. Specify a numeric value for all partition numbers and reissue the command. – USER

`TM286 -- A volume description cannot begin with the '=' delimiter`

The user began the volume description with the = delimiter rather than with the internal volume identifier. Correct the volume description and reissue the command. – USER

`TM288 -- An external volume identifier must be specified after the first '=' delimiter`

The syntax of the volume description is incorrect. A VSN must follow the = delimiter. Correct the volume description and reissue the request. – USER

`TM292 -- Unable to lock the SSP process` *pid*`, managing file` *pathname*`, in memory (errno =` *errno*`)`

Process locking was requested with the `-l` option on the `tmdaemon`(8) command. TMF was unable to lock a Stream Service Program (SSP), created to manage the *pathname* stream in memory; the error returned is *errno*. – USER

`TM294 -- The` *reqt* `request has been interrupted`

The *reqt* request has been interrupted. – USER

`TM295 -- The` *reqt* `request failed because the message size exceeds the segment size`

An invalid front-end request was created. – USER

`TM296 -- Cannot export volume` *vsn* `from loader` *loader1* `and import to loader` *loader2*

The requested *vsn* volume was not found in the domain of the preferred *loader2* loader. The volume could not be exported from the *loader1* loader and exported to *loader2* because either the loader is unattended or the administrator denied the request to move the volume. – USER

`TM297 -- Unable to change the permissions mode of` *file type file name*`(errno =` *errno*`)`

TMF was unable to perform a `chmod`(8) on the named file. Contact your system support staff. – USER/ADM

`TM298 -- Cannot specify a negative value for the` *parameter*

An invalid parameter, *parameter*, was specified. The parameter cannot be negative. The command is terminated. Correct the parameter and reissue the command. – USER/ADM

`TM299 -- Unable to complete the` *cmd* `command (errno =` *errno*`)`

An unexpected error occurred when processing the *cmd* command. Contact your system support staff. – USER/ADM

`TM400 --` *program data_structure1* `V` *version_number1* `is incompatible with system` *data_structure2* `V` *version_number2*`, abort` *program*`.`

The *program* name has been created with *data_structure1* name version *version_number1*. This is incompatible with the *data_structure2* name version *version_number2* that was used to build the TMF daemon. Contact your system support staff to change either one of the components to create a matching set. – ADM

`TM401 -- The TMF child process expected` *n* `list elements from the TMF daemon but instead received` *m* `elements`

When TMF starts a child process, it delivers a data structure to this process. The data structure contains the number of data structures that will follow and the size of each structure. This message indicates that there is a discrepancy between the number delivered and the number the child process has calculated. Contact your system support staff to resolve the discrepancy. – ADM

`TM402 -- Either a device name or 'all' must be specified`

The `tmunld`(8) command requires a device name or `all`. – USER

`TM403 -- Unable to` *enable or disable* `signals for process` *pid*`, stream` *stream* `(errno =` *errno*`)`

The TMF driver communicates with the TMF daemon and its children using signals. When the TMF daemon or a child is ready to receive requests, it enables signals from the TMF driver. When it cannot accept signals, it disables the signals. This message is output if the TMF daemon or child cannot modify the signal state; the error returned is *errno*. – USER

`TM404 -- Unable to resume process` *pid*`, stream` *stream* `(errno =` *errno*`)`

The TMF driver disables further processing of user system call requests when the TMF daemon or its child has a request pending for a user. Once the TMF daemon or its child has completed its processing, it will resume the user request with an `ioctl`(2) request to the TMF driver. This message is output if the request fails. – USER

`TM405 -- The volume mounted on device` *device* `differs from the requested volume`

An incorrect volume was mounted on the *device* device. – USER

`TM407 -- The operator denied access to the scratch volume mounted on device` *device* `for file` *pathname*

A scratch volume was mounted, and scratch volume verification has been configured. The administrator did not allow the mounted volume to be scratched, and the user's request was terminated. – ADM

`TM408 -- The operator canceled the scratch volume mount request`

The mounted volume was not scratchable, and the scratch retry limit was reached. Permission to retry the scratch mount request was denied by the operator. – USER

`TM410 -- User exit function :` *function* `: returned :` *return_code*

The user exit function *function* returned *return_code*. This is a site-defined message. – USER

`TM411 -- Cannot read the labels from volume` *vsn*`, file` *pathname* `(errno =` *errno*`)`

TMF was unable to perform label processing. A label read failed with error *errno*. – ADM

`TM413 -- Permission to bypass label process denied`

Root privilege is required to bypass label processing. – USER

`TM414 -- Exceeded the limit on the number of tape groups which can be reserved,` *count*

You specified more tape groups on the `tmrsv`(1) command than what is allowed. – USER

`TM415 -- The` *option* `specified,` *value*`, must be greater than` *min*

The user specified a value for the `option` option, which is less than the minimal value required for this option. Reissue the command specifying an option value which meets this requirement. – USER

`TM416 -- Because directory` *dir*`, specified with environment variable,` *variable*`, is not writeable, the current working directory will be used instead for the` *file*

The user requested that the file be created in the directory specified with the *variable* environment variable. This directory is not writeable so that the current working directory is used instead. – USER

`TM417 -- Cannot import volume` *vsn* `to loader` *loader*`.`

The requested *vsn* volume was not found in the domain of the preferred *loader* loader. The volume could not be imported to the *loader* loader because either the loader is unattended or the administrator denied the request to import the volume. – ADM

`TM418 -- A resource must be specified`

A resource was not specified on the `tmrls`(1) command. Option `-a`, `-d`, or `-p` must be specified to indicate what resource or resources to release. – ADM

`TM419 -- A path or device name must be specified when requesting a resource privilege be kept`

The `-k` option was specified on the `tmrls`(1) command without specifying a path name or a device name. – USER

`TM420 -- Exceeded the limit on the maximum number of active tape streams.`

A mount request was terminated because TMF was at the limit of active tape streams. This limit was established by the *max_streams* value specified in the TMF configuration file or the default value specified in the `tmfdefaults.h` file. – ADM

`TM421 -- A media loader vendor address must be specified for device,` *device*`, controlled by media loader` *loader*

Specify the library address of the tape drive with the device name specified by the vendor supplied software for that library. For a StorageTek library this address is in the format (*acs,lsm,panel,drive*), and for the EMASS library this address is a single number. Contact your system support staff to specify the correct library address for drive name. – ADM

`TM422 -- The specified file,` *pathname*`, does not exist`

The *pathname* file does not exit. – ADM

`TM423 -- Error in line` *line_number*`, offset` *char_offset*`, with value` *value*`. Vendor address,` *value1*`, cannot exceed` *value2*

A value is specified for *value1* in the vendor address that is greater than the maximum boundary, *value2,* that can be specified for *value1* in the vendor address. Contact your system support staff to change the value for *value1* to a number that is less than *value2*. – ADM

`TM424 -- Error in line` *line_number*`, offset` *char_offset*`, with value` *value*`. The STK vendor address must be of the format, '(acs,lsm,panel,drive)'.`

The StorageTek library vendor address has not been specified correctly: more values were encountered than the four making up the StorageTek library vendor address. Contact your system support staff to specify a correct StorageTek library address. – ADM

`TM425 -- Error in file` *config_file_name*`, line` *line_number*`, offset` *char_offset*`, with value` *value* `:` *text*

Correct the error, *text,* and run again – ADM

`TM426 -- Error in file` *config_file_name*`, line` *line_number*`, offset` *char_offset*`, with parameter` *parameter*`. Exceeded the maximum number of` *items* `permitted,` *number*

More than *number* of *items* were specified. Reduce the number of *items* and reissue the command – ADM

`TM427 -- Error in file` *config_file_name*`, line` *line_number*`, offset` *char_offset*`, with value` *value*`. The server name specified,` *char_string*`, is invalid`

The server name specified by *char-string* is invalid. Specify a correct server name and reissue the command. – ADM

`TM428 -- Error in file` *config_file_name*`, line` *line_number*`, with statement preceding` *keyword_parameter*`. Parameter` *keyword_parameter* `is required in the preceding statement.`

keyword_parameter is required in the preceding statement. Specify *keyword_parameter* and reissue the command. – ADM

`TM429 -- Error in file` *config_file_name*`, line` *line_number*`, offset` *char_offset*`, with value` *value*`. The` *value* `specified,` *char_string*`, cannot exceed` *number* `characters.`

The name *char_string* is too long. The maximum length is *number*. Reduce length of name and reissue the command. – ADM

`TM430 -- Error in file` *config_file_name*`, line` *line_number*`, offset` *char_offset*`, with value` *value*`.` *value char_string* `has already been defined`

The parameter *char_string* has already been defined. Remove this instance of the parameter and reissue the command. – ADM

`TM433 -- Error in file` *config_file_name*`, line` *line_number*`, offset` *char_offset*`, with value` *value*`. value` *value* `has not been defined`

The item specified by the *value* value has not been defined. Add the definition of *value*. – ADM

`TM435 -- Unable to set the effective user ID of` *process_name* `to root (errno =` *errno*`)`

TMF process, *process_name,* must execute as root. The process was terminated because the effective user identifier could not be set to `root`; the error returned is *errno*. – ADM

`TM436 -- The device group for device,` *device_name*`, cannot be modified when the device is configured up`

The device cannot be reassigned a device group name unless it is in a down state. The `tmconfig`(8) request has been terminated. Configure the device down and reissue the command. – ADM

`TM437 -- Unable to obtain the status of the` *type*`,` *parameter* `(errno =` *errno*`)`

TMF was unable to obtain the file status of *pathname*; the error returned is *errno*. – ADM

`TM438 -- The binary configuration` *file* `not created because of a previous error`

The binary *file* file is not created because of a previous error. Correct the previous error and reissue the command. – ADM

`TM439 -- A servicing front-end identifier must be specified when mandatory front-end servicing is requested`

Mandatory front-end servicing was requested in the TMF configuration file with the `servicing_frontend_mandatory` option. The `servicing_frontend_id` option must also be specified with this option. – ADM

`TM440 -- The servicing front-end identifier is ignored when mandatory front-end servicing is not requested.`

The `servicing_frontend_id` option was specified in the TMF configuration file. This option is only required when the `servicing_frontend_mandatory` option is specified. Otherwise, it is ignored. – ADM

`TM441 -- The version of command tmconf is incompatible with the version of the TMF daemon`

The TMF daemon was unable to successfully use the `tmconf`(8) command because the version of the `tmconf`(8) command is incompatible with the version of the TMF daemon. – ADM

`TM442 -- A loader name is valid only with options -d and -s`

A loader name can only be specified on the `tmmls`(8) command with the `-d` or `-s` option. – ADM

`TM443 -- Exceeded the limit on the maximum number of loaders which can be specified,` *number*

You specified more loader names on the `tmmls`(8) command than what is allowed. – ADM

`TM447 -- Unable to open file,` *pathname*`, needed to configure the TMF subsystem (errno =` *errno*`)`

TMF start-up processing requires an open of the *pathname* file. TMF could not be started because this file could not be opened. The error returned is *errno*. – ADM

`TM448 -- The Tape Management Facility is already active`

The TMF daemon could not be started because the TMF daemon is already active. Shutdown TMF and then reissue the `tmdaemon`(8) command. – ADM

`TM449 -- Unable to complete the` *request* `request issued to device` *device* `for stream` *pathname* `because of a hardware error`

A nonrecoverable hardware error occurred while the device was processing the *request* request. – USER

`TM450 -- Unable to complete the` *req* `request issued to device` *device* `for stream` *pathname* `because the device is not ready`

The *device* device could not be accessed either because the device is offline or there is no volume in the drive. – USER

`TM451 -- Unable to complete the` *req* `request issued to device` *device* `for stream` *pathname* `because the actual block size differs from the set size`

A `read`(2) request issued to a fixed-length I/O device failed because the block size was not set to the correct length. – USER

`TM452 -- Unable to complete the` *req* `request issued to device` *device* `for stream` *pathname* `because the volume was not loaded correctly`

A load error was detected when attempting to mount a tape volume. – USER

`TM453 -- Unable to complete the` *req* `request issued to device device for stream` *pathname* `because the volume is write protected`

A `write`(2) request failed because `write`(2) requests are disabled for the mounted volume. – USER

`TM454 -- Unable to complete the` *pathname* `request issued to device` *device* `for stream` *pathname* `because the physical end of the media was detected`

A `write`(2) request failed because the physical end of the tape media had been reached. – USER

`TM455 -- Unable to complete the` *req* `request issued to device` *device* `for stream` *pathname* `because the physical end of the media was detected`

The request failed because the beginning of the tape media was detected. – USER

`TM456 -- Unable to complete the` *req* `request issued to device` *device* `for stream` *pathname* `because a filemark was detected`

The request failed because a filemark was detected. – USER

`TM457 -- Unable to complete the` *req* `request issued to device` *device* `for stream` *pathname* `because the end of data was detected`

The request failed because the end of the recorded data was detected. – USER

`TM458 -- Unable to complete the` *req* `request issued to device` *device* `for stream` *pathname* `because the end of tape warning was detected`

The request failed because the early warning marking, indicating that the tape is positioned close to the physical end-of-tape, has been detected. – USER

`TM459 -- Unable to complete the` *req* `request issued to device` *device* `for stream` *pathname* `because the device state has changed`

Either the volume has been changed or the device was reset while processing the *req* request. – USER

`TM460 -- Unable to complete the` *req* `request issued to device` *device* `for stream` *pathname* `because a setmark was detected`

The request failed because a setmark was detected. – USER

`TM461 -- Unable to complete the` *req* `request issued to device` *device* `for stream` *pathname* `because the device mode is incompatible with the media mode`

The mode in which the media was recorded is incompatible with the current device mode. – USER

`TM462 -- Unable to complete the` *req* `request issued to device` *device* `for stream` *pathname* `because of an unrecovered data error`

An error was detected with the recorded data. – USER

`TM463 -- Unable to complete the` *req* `request issued to device` *device* `for stream` *pathname* `because the request was invalid`

An invalid request was sent to the tape device. – USER

`TM464 -- Unable to complete the` *req* `request issued to device` *device* `for stream` *pathname* `because the block size exceeds the request size`

A `read`(2) request failed because the buffer size is not large enough for the recorded data block. – USER

`TM465 -- Unable to complete the` *req* `request issued to device` *device* `for stream` *pathname* `because the volume was not loaded correctly`

The requested tape volume was not properly loaded which caused the user's request to fail. Contact your system support staff. – USER/ADM

`TM466 -- Unable to complete the` *req* `request issued to device` *device* `for stream` *pathname* `because of a hardware error`

The tape driver detected a fatal hardware error, which prevented the successful execution of the user's request. Contact your system support staff. – USER/ADM.

`TM467 -- Unable to complete the` *req* `request issued to device` *device* `for stream` *pathname* `because the device is not ready`

The requested tape has been loaded but is currently not ready. Contact your system support staff. – USER/ADM

`TM468 -- Unable to complete the` *req* `request issued to device` *device* `for stream` *pathname* `because the actual block size differs from the set size`

The block size of data on the requested tape is different from the size specified in the `tmmnt`(1) command. – USER

`TM469 -- Unable to position by files`

TMF was unable to skip the specified number of files. Check your TMF message file for additional information about the positioning error. – USER

`TM470 -- Unable to write a filemark to volume` *volume*

TMF was unable to write a filemark to the *volume* volume. Check your TMF message file for additional information about the write filemark error. – USER

`TM471 -- Option` *option* `cannot be specified in the tmmnt parameters`

The `tmmnt`(1) *option* option was specified on the additional parameters option, `-p` or `-P` of the `tmlist`(1) command. It is not a valid option. Remove this option from the additional parameters option value and reissue the `tmlist`(1) command. – USER

`TM472 --The specified loader,` *loader*`, has no associated devices`

The administrator issued a `tmconfig`(8) request for a loader that does not have any associated devices. – ADM

`TM473 -- Device` *device* `is not supported`

The *device* device, specified in the TMF configuration file, is a type that TMF does not support. TMF startup will be terminated. Remove the *device* device, which is not a valid entry, and restart TMF. – ADM

`TM474 -- No devices have been configured`
No valid devices have been specified in the TMF configuration file. TMF startup will be terminated. Correct the configuration file and restart TMF. – ADM

`TM475 -- Should error recovery be attempted for the session on device` *device*`. Reply y/n?`
An I/O error was encountered on the *device* device. You must specify whether error recovery should be attempted. Reply **y** for "yes" or **n** for "no." – ADM

`TM476 -- Unable to allocate a new device for error recovery. Reply y/n to retry the assigned device.`
An I/O error was encountered on the assigned device. TMF error recovery could not assign another device. Reply **y** to attempt error recovery using the assigned device. – ADM

`TM477 -- The load failure retry limit has been reached`
TMF was unable to recover from a volume load error before reaching the retry limit. – USER

`TM478 -- Unable to load volume` *volume* `on device` *device*
An autoloader volume mount of the *volume* volume on the *device* device completed successfully, but the device was not able to correctly load the volume. The mount request has been terminated. – USER

`TM479 -- Unable to complete the mount request (errno =` *errno*`)`
A mount request failed with the *errno* error. See your TMF message file for additional information on the error. – USER

`TM480 -- The operator has forcibly released your tape resources`
The administrator has issued a `tmfrls`(8) command, and all your resources have been released. – USER

`TM482 -- Accounting routines are not available, accounting is disabled`
The administrator attempted to start TMF with accounting enabled, but the accounting feature is not installed on this operating system. – USER

`TM500 -- Select error (errno =` *error*`)`

An error occurred while the `tmavr` process was waiting for a request from the TMF daemon. Run the `tmcollect`(8) command and give this information to your system support staff. – ADM

`TM501 -- Device operation` *operation type* `failed for device` *name* `file` *path* `(`*reason*`: errno =` *error*`)`
An error *operation type* occurred while the `tmavr` process was attempting to collect label information on the *name* device. Run the `tmcollect`(8) command and give this information to your system support staff. – ADM

`TM502 -- An invalid AVR request (`*code*`) was sent to tmavr.`
The `tmavr` process received an unexpected command *code* while automatic volume recognition (AVR) was being used. Run the `tmcollect`(8) command and give this information to your system support staff. – ADM

`TM503 -- Unexpected errno (`*error*`), expected ESRCH on kill for process` *pid*
The system could not deliver a signal to a *tmavr* process *pid*. Run the `tmcollect`(8) command and give this information to your system support staff. – ADM

`TM504 -- Label read failed for device` *name* `(`*reason*`); should the tape be used? Reply y/n`
A label could not be read on the *name* device for *reason*. The administrator is asking you if the tape should be used. Answer **y** for "yes"or **n** for "no." – USER/ADM

`TM505 -- Error in file` *file*`, line` *line*`, offset` *offset*`, with value` *value group device* `a library device may not belong to a device group with AVR enabled`
The library device may not belong to a device group for which automatic volume recognition (AVR) has been specified. Correct the TMF configuration file and restart TMF. – ADM

`TM911 -- TMF failure, please run /usr/sbin/tmcollect for:` *reason*
An internal TMF error has been detected *reason,* and this message is logged to the `/var/spool/tmf/daemon.stderr` file. Run the `tmcollect`(8) command and give this information to your system support staff. – ADM

`TM994 -- Unable to open TMF lockfile` *lockfile* `(errno =` *errno*`)`
During TMF initialization, the TMF daemon could not open the lockfile required for startup. – ADM

`TM995 -- The use of the Tape Management Facility (TMF) is regulated by\n license.  The License Manager is unable\n to grant permission to use TMF (error` *errno - reason*`)\n Contact SGI to obtain a license and/or a key\n to use this software.`

`TM996 -- Process` *process id* `has open tape device and can't be killed.`

TMF is attempting to remove an active user and cannot kill the process that opened the tape file. – ADM

`TM997 -- Process` *process id* `exited , killing related pids with` *command*`.`

TMF has detected the exit of the user process that owns the open tape file. It is attempting to kill any related processes. – ADM

`TM998 -- Tape subsystem terminating, request denied`

A request was terminated because TMF is in the shutdown process. Reissue the request after TMF has been brought up. – USER

`TM999 -- Tape subsystem busy, TMF daemon termination pending.`

TMF is terminating and has found an active user; this delays the termination. – ADM

Man Pages

This appendix lists the man pages for the TMF user commands.

Individual man pages are available online and can be accessed by using the `man`(1) command as shown in the following example:

```
% man tmstat
```

You can print copies of online man pages by using the pipe symbol with the `man`(1), `col`(1), and `lpr`(1) commands. In the following example, these commands are used to print a copy of the `tmstat`(1) man page:

```
% man tmstat | col -b | lpr
```

Each man page includes a general description of one or more commands, system calls, or other topics, and provides details of their usage (command syntax, parameters, and so on). User commands are as follows:

`msgr`(1)
`tmcatalog`(1)
`tmlist`(1)
`tmmnt`(1)
`tmrls`(1)
`tmrst`(1)
`tmrsv`(1)
`tmstat`(1)

Index

E

F

H

I

L

M