IRIX® TMF Administrator's Guide

New Features in This Guide

New features include the deletion of the `NQSid` field from the output from the `tmstat` command and changes in the format of the SGI node locked licenses issued for the SGI Origin 300, SGI Origin 3000, and SGI Onyx 3000 series systems.

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 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.

Contents

Figures

Tables

Examples

Procedures

About This Guide

This guide documents the administration of the Tape Management Facility (TMF) running on the IRIX operating system. It introduces TMF, summarizes administration commands, covers configuration, documents administration information, and describes troubleshooting techniques.

Related Publications

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

- *TMF Release and Installation Guide*

- *TMF User's Guide*

If you are using TMF with OpenVault, see the following manual for OpenVault operating and administration information:

- *OpenVault Operator's and Administrator's Guide*

TMF Man Pages

In addition to printed and online prose documentation, several online man pages describe aspects of TMF. For a list of TMF man pages and usage information, see Appendix A, page 45.

Obtaining Publications

To obtain SGI documentation, go to the SGI Technical Publications Library at:

`http://techpubs.sgi.com.`

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 document, please tell us. Be sure to include the title and document number of the manual with your comments. (Online, the document number is located in the front matter of the manual. In printed manuals, the document number is located at the bottom of each page.)

You can contact us 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 World Wide Web page:

  http://techpubs.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 Pkwy., M/S 535
  Mountain View, California 94043–1351

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

We value your comments and will respond to them promptly.

Introduction

This guide describes key Tape Management Facility (TMF) administration tasks and provides information on performing them. It covers configuration, administration, and troubleshooting.

It documents the following administrative topics:

- TMF, tape interfaces, and the TMF administration commands

- Sample TMF configuration file, FLEXlm license file editing, and statement rules and syntax

- Tape libraries, the message daemon and operator interface, TMF startup, dumping to tape, and OpenVault usage

- Tape troubleshooting

- Names of TMF man pages and information on accessing them

- `tmf.config`(5) man page

1.1 Interfaces

Users can access tapes attached to an IRIX system by one of two interfaces: TMF, which is described in this manual, and the character-special tape interface (`tpsc`). For more information, see the `tpsc`(7M) man page. This manual describes TMF.

1.2 Administration Commands

This section briefly describes the TMF administration commands. For more information, see the individual man pages. Appendix A, page 45, contains information on accessing these pages.

`msgd`(8)	The `msgd`(8) command allows the operator to display action messages, such as tape mount messages.
`msgdaemon`(8)	The `msgdaemon`(8) command starts the message daemon, which handles the communication between users and operators.

msgdstop(8)	The msgdstop(8) command causes the message daemon to stop executing.
newmsglog(8)	The newmsglog(8) command saves the latest versions of the message log file.
oper(8)	The oper(8) command displays action messages and runs other commands as refresh displays.
rep(8)	The rep(8) command allows the operator to respond to action messages, such as tape mount messages.
tmclr(8)	The tmclr(8) command clears a tape stream. All tables and data associated with that device are cleared, if possible.
tmcollect(8)	The tmcollect(8) command collects information for TMF problem analysis.
tmconf(8)	The tmconf(8) command verifies the TMF configuration file.
tmconfig(8)	The tmconfig(8) command configures tape devices up and down, changes the status of the associated media loaders, assigns a media loader to a device, and reassigns a device group to a device.
tmdaemon(8)	The tmdaemon(8) command starts TMF. It provides the routing and control of the various components used in tape resource management, device management, volume mounts and dismounts through operator communication or library requests, label processing, volume switching, and error recovery.
tmfrls(8)	The tmfrls(8) command lets the operator release the tape reservations made by a user.
tmgstat(8)	The tmgstat(8) command displays the reservation status for each device group in the system for each tape user.
tmlabel(8)	The tmlabel(8) command labels tapes and may perform other functions depending on the interface being used.
tmmls(8)	The tmmls(8) command displays the status of the tape loaders in the system.

`tmmql`(8)	The `tmmql`(8) command displays the current mount request list for all users who have completed initial mount processing and have a mount request pending.
`tmset`(8)	The `tmset`(8) command sets features for TMF. It changes the status of automatic volume recognition (AVR), front-end servicing (FES), the status of tracing for TMF, or the destination of tape operator messages issued by TMF.
	All of these features can be set in the TMF configuration file.
`tmstop`(8)	The `tmstop`(8) command stops TMF, which terminates in an orderly fashion.
`tmunld`(8)	The `tmunld`(8) command is used by the system operator to unload tapes. This command has no effect on a tape that is currently in use.

Configuration

This chapter describes how to create your TMF configuration file and how to edit your FLEXlm license file. It also provides information on the configuration file statements, job limits, and comprehensive system accounting (CSA).

The TMF configuration file, `tmf.config` in the `/etc/config` directory, must be updated to configure TMF before TMF is started. You should set the parameters in your TMF configuration file to values that suit your system. TMF uses these values to decide what to do in various situations.

You can update the file with any text editor. The basic information that you need for this task is contained in this chapter. For a description of the parameters, see the `tmf.config`(5) man page. During initial system startup, refer to Appendix B, page 47.

2.1 TMF Configuration File

To become familiar with the TMF configuration elements, review the file in Example 2-1. The following subsections highlight how you configure the statements within the TMF configuration file.

Example 2-1 TMF Configuration File

This TMF configuration file contains five statements: LOADER, DEVICE_GROUP, AUTOCONFIG, which is composed of DEVICE statements, and OPTIONS

```
#
#       TAPE MANAGEMENT FACILITY CONFIGURATION FILE
#
#
#

LOADER
      name = operator ,
      type = OPERATOR ,
      status = UP ,
      mode = ATTENDED ,
      message_path_to_loader = MSGDAEMON ,
      server = IRIX ,
```

```
        scratch_volume_label_type = (AL,NL,SL) ,
        queue_time = 0 ,
        verify_non_label_vsn = YES ,
        message_route_masks = (IRIX) ,
        loader_ring_status = ALERT

LOADER
        name = wolfy ,
        type = STKACS ,
        status = DOWN ,
        mode = ATTENDED ,
        message_path_to_loader = NETWORK ,
        server = wolfcreek ,
        scratch_volume_label_type = NONE ,
        queue_time = 15 ,
        verify_non_label_vsn = NO ,
        message_route_masks = (IRIX) ,
        loader_ring_status = IGNORE

LOADER
        name = panther ,
        type = STKACS ,
        status = DOWN ,
        mode = ATTENDED ,
        message_path_to_loader = NETWORK ,
        server = stk9710 ,
        scratch_volume_label_type = NONE ,
        queue_time = 15 ,
        verify_non_label_vsn = NO ,
        message_route_masks = (IRIX) ,
        loader_ring_status = IGNORE

LOADER
        name = esys ,
        type = EMASS ,
        status = DOWN ,
        mode = ATTENDED ,
        message_path_to_loader = NETWORK ,
        server = esisun ,
        scratch_volume_label_type = NONE ,
        queue_time = 15 ,
```

```
            verify_non_label_vsn = NO ,
            message_route_masks = (IRIX) ,
            loader_ring_status = IGNORE

      LOADER
            name = tmfov ,
            type = OPENVAULT ,
            server = armadillo ,
            status = down ,
            mode = ATTENDED ,
            message_path_to_loader = NETWORK ,
            ov_tmf_application_name = tmf,
            scratch_volume_label_type = NONE ,
            queue_time = 15 ,
            verify_non_label_vsn = NO ,
            message_route_masks = (IRIX) ,
            loader_ring_status = IGNORE


      DEVICE_GROUP
            name = CART
            avr  = YES

      DEVICE_GROUP
            name = DLT

      DEVICE_GROUP
            name = EMASS

      DEVICE_GROUP
            name = STK9490

      AUTOCONFIG
      {
            DEVICE
                    name   = t1 ,
                    device_group_name = CART ,
                    file   = /hw/tape/tps3d1 ,
                    status = DOWN ,
                    loader = wolfy ,
                    vendor_address = (0,0,1,1)
```

```
DEVICE
        name   = t4 ,
        device_group_name = CART ,
        file   = /hw/tape/tps3d4 ,
        status = DOWN ,
        loader = wolfy ,
        vendor_address = (0,0,1,0)
DEVICE
        name   = dlt2 ,
        device_group_name = DLT ,
        file   = /hw/tape/tps5d2 ,
        status = DOWN ,
        loader = panther ,
        vendor_address = (1,0,2,0)
DEVICE
        name   = dlt3 ,
        device_group_name = DLT ,
        file   = /hw/tape/tps5d3 ,
        status = DOWN ,
        loader = panther ,
        vendor_address = (1,0,2,1)
DEVICE
        name   = ed0 ,
        device_group_name = EMASS ,
        file   = /hw/tape/tps10d0 ,
        status = DOWN ,
        loader = esys ,
        vendor_address = (1)
DEVICE
        name   = s9490s4 ,
        device_group_name = STK9490 ,
        file   = /hw/tape/tps22d4 ,
        status = down ,
        vendor_address = (0,0,1,0),
        loader = tmfov
DEVICE
        name   = s9490s1 ,
        device_group_name = STK9490 ,
        file   = /hw/tape/tps22d1 ,
        status = down ,
        vendor_address = (0,0,1,1),
```

```
                  loader = tmfov
}
OPTIONS
ask_label_switch                    = YES ,
ask_vsn                             = YES ,
blocksize                           = 32768 ,
blp_ring_status                     = UNRESTRICTED ,
check_expiration_date               = YES ,
check_file_id                       = YES ,
check_protection                    = YES ,
check_vsn                           = YES ,
device_group_name                   = CART ,
fes_daemon_frontend_id              = "mvs" ,
fes_daemon_socket_port_number       = 1167 ,
file_status                         = OLD ,
label_type                          = AL ,
loader_device_assignment_order      = ROUND_ROBIN ,
max_number_of_tape_users            = 100 ,
number_of_autoloader_retries        = 10 ,
operator_message_destination        = (IRIX) ,
operator_message_frontend_id        = "" ,
overcommit_max                      = 20
retention_period_days               = 0 ,
ring_status                         = (IN,OUT) ,
scratch_volume_retries              = 0 ,
scratch_volume_vsn                  = ?????? ,
servicing_frontend_id               = "" ,
servicing_frontend_mandatory        = NO ,
system_code                         = SGI/IRIX ,
tmf_major                           = 261 ,
trace_file_group_id                 = 3 ,
trace_file_mode                     = 0640 ,
trace_file_owner                    = 0 ,
trace_directory                     = /var/spool/tmf/trace ,
trace_file_size                     = 409600 ,
trace_state                         = ON ,
trace_save_directory                = /var/spool/tmf/trace_save ,
user_exit_mask                      = UEX_STOP ,
verify_scratch_vsn                  = NO
```

2.1.1 LOADER Statement

The TMF configuration file in Example 2-1, page 5, contains five LOADER statements; these represent the five loaders that are available on this IRIX system. Each LOADER statement is composed of the parameters needed to describe a specified loader. For example, the first LOADER statement has 11 parameters:

```
LOADER
        name = operator ,
        type = OPERATOR ,
        status = UP ,
        mode = ATTENDED ,
        message_path_to_loader = MSGDAEMON ,
        server = IRIX ,
        scratch_volume_label_type = (AL,NL,SL) ,
        queue_time = 0 ,
        verify_non_label_vsn = YES ,
        message_route_masks = (IRIX) ,
        loader_ring_status = ALERT
```

The name of the loader is operator, the type is OPERATOR, the status is UP, and the mode is ATTENDED.

The message path to the servicing loader is MSGDAEMON; the server name is IRIX.

The loader will process ANSI (AL), nonlabeled (NL), and IBM (SL) scratch requests. The system will queue a request and wait for the best loader to become available for up to 24 hours.

Nonlabeled VSNs must be verified. message_route_masks is IRIX, which means that mount request messages are routed to the message daemon. The loader is alerted to the ring status whenever a tape is mounted.

It may be necessary to specify an alternate network name for LOADER statements that represent NETWORK libraries. If a NETWORK library is not connected to the host primary network, the path must be specified with the return_host parameter so that the library can return responses to TMF. This parameter is only used if it is set; there is no default.

2.1.2 `DEVICE_GROUP` Statement

The file in Example 2-1, page 5, contains four DEVICE_GROUP statements, one for each of the system's device groups:

```
DEVICE_GROUP
      name = CART
      avr  = YES

DEVICE_GROUP
      name = DLT

DEVICE_GROUP
      name = EMASS

DEVICE_GROUP
      name = STK9490
```

The first DEVICE_GROUP statement supports the automatic volume feature.

2.1.3 `AUTOCONFIG` Statement

The AUTOCONFIG statement in Example 2-1, page 5, is made up of seven DEVICE statements, one for each device in the system:

```
AUTOCONFIG
{
      DEVICE
                  name  = t1 ,
                  device_group_name = CART ,
                  file  = /hw/tape/tps3d1 ,
                  status = DOWN ,
                  loader = wolfy ,
                  vendor_address = (0,0,1,1)
      DEVICE
                  name  = t4 ,
                  device_group_name = CART ,
                  file  = /hw/tape/tps3d4 ,
                  status = DOWN ,
                  loader = wolfy ,
                  vendor_address = (0,0,1,0)
      DEVICE
```

```
                            name   = dlt2 ,
                            device_group_name = DLT ,
                            file   = /hw/tape/tps5d2 ,
                            status = DOWN ,
                            loader = panther ,
                            vendor_address = (1,0,2,0)
            DEVICE
                            name   = dlt3 ,
                            device_group_name = DLT ,
                            file   = /hw/tape/tps5d3 ,
                            status = DOWN ,
                            loader = panther ,
                            vendor_address = (1,0,2,1)
            DEVICE
                            name   = ed0 ,
                            device_group_name = EMASS ,
                            file   = /hw/tape/tps10d0 ,
                            status = DOWN ,
                            loader = esys ,
                            vendor_address = (1)
            DEVICE
                            name   = s9490s4 ,
                            device_group_name = STK9490 ,
                            file   = /hw/tape/tps22d4 ,
                            status = down ,
                            vendor_address = (0,0,1,0),
                            loader = tmfov
            DEVICE
                            name   = s9490s1 ,
                            device_group_name = STK9490 ,
                            file   = /hw/tape/tps22d1 ,
                            status = down ,
                            vendor_address = (0,0,1,1),
                            loader = tmfov
```

2.1.4 `DEVICE` Statement

The DEVICE statement identifies the tape devices that are available on the system on which TMF is running. In the first DEVICE statement in the AUTOCONFIG statement, shown in the Example 2-1, page 5, the device is t1.

```
{
      DEVICE
                name   = t1 ,
                device_group_name = CART ,
                file   = /hw/tape/tps3d1 ,
                status = DOWN ,
                loader = wolfy ,
                vendor_address = (0,0,1,1)
```

This device is a member of the CART device group, which is specified by the first
DEVICE_GROUP statement (see Section 2.1.2, page 11).

The path name to the device specific file is /hw/tape/tps3d1. The initial status of
the device is DOWN. The vendor address of the drive in the library is (0,0,1,1).

The loader name is wolfy, and it is defined in the second LOADER statement in
Example 2-1, page 5.

2.1.5 OPTIONS Statement

The OPTIONS statement shows the values that TMF uses for the options. For a
description of each option, see the tmf.config(5) man page.

In the file in Example 2-1, page 5, the defaults are used for all options except the
following:

```
check_protection                          = YES ,
fes_daemon_frontend_id                    = "mvs" ,
scratch_volume_retries                    = 0 ,
user_exit_mask                            = UEX_STOP ,
verify_scratch_vsn                        = NO
```

YES for check_protection means the protection flag on the header is checked.
fes_daemon_frontend_id specifies mvs for the front-end identifier of the TCP
daemon. Because scratch_volume_retries is set to 0, users are not allowed to
retry scratch volume mount requests. TMF stops and enables one or more user exits
for the site since UEX_STOP is the value for user_exit_mask. Because the value for
verify_scratch_vsn is NO, users do not send the operator a message requesting
verification whenever they want to use a scratch tape.

2.2 FLEXlm License File Editing

During the process of installing the TMF product, a FLEXlm license is e-mailed or sent to you. The license takes the form of a FLEXlm feature line, for example:

```
FEATURE tmf_sgi sgifd 1.000 01-jan-0 0 DC7B2697BE94835E6E8B HOSTID="c0000029 \
cfc0001d cf80001d cf40001d cf00001d cec0001d ce80001d ce40001d ce00001d \
cdc0001d cd80001d cd40001d cd00001d ccc0001d cc80001d cc40001d cc00001d \
cbc0001d cb80001d cb40001d cb00001d cac0001d ca80001d ca40001d ca00001d \
c9c0001d c980001d c940001d c900001d c8c0001d c880001d c840001d c800001d \
c7c0001d c780001d c740001d c700001d c6c0001d c680001d c640001d c600001d \
c5c0001d c580001d c540001d c500001d c4c0001d c480001d c440001d c400001d \
c3c0001d c380001d c340001d c300001d c2c0001d c280001d c240001d c200001d \
c1c0001d c180001d c140001d c100001d c0c0001d c080001d c040001d" \
vendor_info="TMF 1.3.5 for IRIX" VENDOR_STRING="TMF IRIX" \
ISSUER="Silicon Graphics, Inc." ck=118
```

These lines need to be added to the FLEXlm `license.dat` file on the TMF system. You can edit the `license.dat` file using `vi(1)` or another editor. The file is located in the following directory:

```
/var/flexlm/license.dat
```

2.3 Statements

The TMF configuration file consists of comments (optional) and statements:

- A comment begins with the number sign character (#) and continues to the end of line.

- A statement consists of a name followed by a list of parameters.

2.3.1 Statement Order

There are, at least, four statements in a TMF configuration file; and one of which also consists of statements. Within the file, the statements must be in the following order:

1. `LOADER` statements (one per loader)

2. `DEVICE_GROUP` statements (one per device group)

3. `AUTOCONFIG` statement (one per system)

   The `AUTOCONFIG` statement consists of `DEVICE` statements.

   - `DEVICE` statements (one per device)

     `DEVICE` statements define devices that TMF will control and that are automatically configured during the system boot.

4. `OPTIONS` statement (one per system)

2.3.2 Syntax Rules

The following syntax rules apply to the TMF statements:

- The statement name and its parameters are separated by one or more white spaces (blank, tab, or newline characters).

- Adjacent parameters are separated by a comma.

- The end of the parameter list is indicated by the absence of a comma.

- Adjacent statements are separated by one or more white spaces.

The following syntax rules apply to keyword parameters:

- The keyword is separated from its value by the equal sign (=).

- The value of a keyword may consist of keywords, numbers, character strings, and lists of keywords, numbers, and character strings.

- If the value of a keyword is a list, then the list is enclosed within left and right parentheses. Adjacent elements of a list are separated by a comma. If the list consists of one element, you do not have to enclose it in parentheses. The elements of a list may be lists.

- Numbers may be specified in decimal, octal, and hexadecimal formats. These formats are the same as those used in the C programming language:

Decimal	The first digit is not 0 (for example, 1372).
Octal	The first digit is 0 (for example, 0563).

| Hexadecimal | The first 2 characters are either 0x or 0X (for example, 0xf2). |

- Character strings are series of characters. If any one of the special characters (white space, ", #, =, {, }, (,), ', \) is needed in the string, then the string must be enclosed in a pair of double quotation marks, ("). Within a pair of double quotation marks, the sequence of characters \ *x*, where *x* is any character, will be replaced by *x*. This is the only way a " and a \ may be specified in a quoted string.

- Comments may appear between any symbols described above.

You can code the names of statements and keywords in a mixture of uppercase and lowercase letters. The values specified by the user are case sensitive. The following mean the same thing:

```
Name = A
name = A.
```

The following are different:

```
name = A
name = a.
```

2.4 Job limits and the ULDB

The user limits database (ULDB) allows a site to configure subsystems and related limits on a global or per user basis. A site may add a TMF domain to the ULDB to control access to TMF managed tapes and devices as in Procedure 2-1. The TMF domain defines tape access permissions and resource allocation limits. For more information on configuring the ULDB, see *IRIX Admin: Resource Administration*.

Procedure 2-1 Defining a TMF Domain

1. Define the following permissions as part of the TMF domain:

| bypasslabel | Allows or prevents the use of -l blp on the tmmnt(1) command. |
| rwnonlabel | Allows or prevents the use of -l nl on the tmmnt(1) command. |

<table>
<tr><td>datamanager</td><td>Allows or prevents the use of absolute positioning requests.</td></tr>
</table>

You set these permissions to 0 or 1:

```
bypasslabel = 0 | 1
rwnonlabel = 0 | 1
datamanager = 0 | 1
```

These typically are set to 0 for the global TMF domain definition.

2. Define the appropriate limit on the number of devices for each group that may be allocated by users.

The groups defined in the TMF configuration file must be defined with an appropriate limit on the number of devices for each group that may be allocated by users.

Example 2-2 Global Device Group Definitions

The TMF configuration file, `tmf.config`, contains the following two DEVICE_GROUP statements:

```
DEVICE_GROUP
        name = STK9840
DEVICE_GROUP
        name = DLT
```

And if a user can allocate only two devices from either group, then the TMF domain definitions are:

```
STK9840 = 2
DLT = 2
```

A global TMF domain for the above specifications is:

```
domain tmf {
        bypasslabel = 0
        rwnonlabel = 0
        datamanger = 0
        STK9840 = 2
        DLT = 2
}
```

Example 2-3 Specific Device Group Definitions

If an administrator has a domain that allows users to label tapes or handle
nonlabelled tapes, the following specifications apply:

```
user adm {

        tmf {
                bypasslabel = 1
                rwnonlabel = 1
                datamanager = 0
                STK9840 = unlimited
                DLT = unlimited
        }
```

Currently, TMF does not have the means to determine to which job domain a user
belongs; that is, TMF cannot distinguish whether a user who is requesting TMF
services is using batch or interactive facilities. Consequently, you must define the
TMF domain and related user limits to accommodate any domain to which the
user may belong.

2.5 TMF and CSA

To generate comprehensive system accounting (CSA) records, you start TMF with the
-c option on the tmdaemon(8) command line. This option may be added to the
tmf.options file to enable accounting at TMF startup. If the necessary accounting
routines are not present, a warning message is generated in the
/var/spool/tmf/daemon.stderr file and TMF accounting is disabled. For more
information on the comprehensive system accounting, see the CSA(1M) man page.

Administration

This chapter describes the following TMF administration topics:

- Tape libraries

- OpenVault as a loader

- Automatic volume recognition (AVR)

- Message daemon and operator interface

- Starting and stopping TMF

- Dumping to tape

3.1 Tape Libraries

This section describes how TMF interacts with the tape library software subsystem and also covers some high-level configuration information for StorageTek, IBM, and EMASS libraries (automatic loaders).

3.1.1 Communication

TMF always communicates to the tape loader via an intermediate software system that is provided by the library vendor.

For the StorageTek library, a software package called ACSLS runs on a SUN host. For the IBM library, a software package called `Controlled Path Service` (CPS) runs on an IBM RISC System/6000 platform. For the EMASS library, the package is called `VolServ` and runs on a SUN host.

These software systems, ACSLS, CPS, and `VolServ`, receive requests from TMF and pass them on to the actual tape libraries for processing. They also send responses back to TMF for any given action.

The diagram in Figure 3-1, page 20, shows the software and hardware configuration between the IRIX host and the StorageTek, IBM, and EMASS libraries.
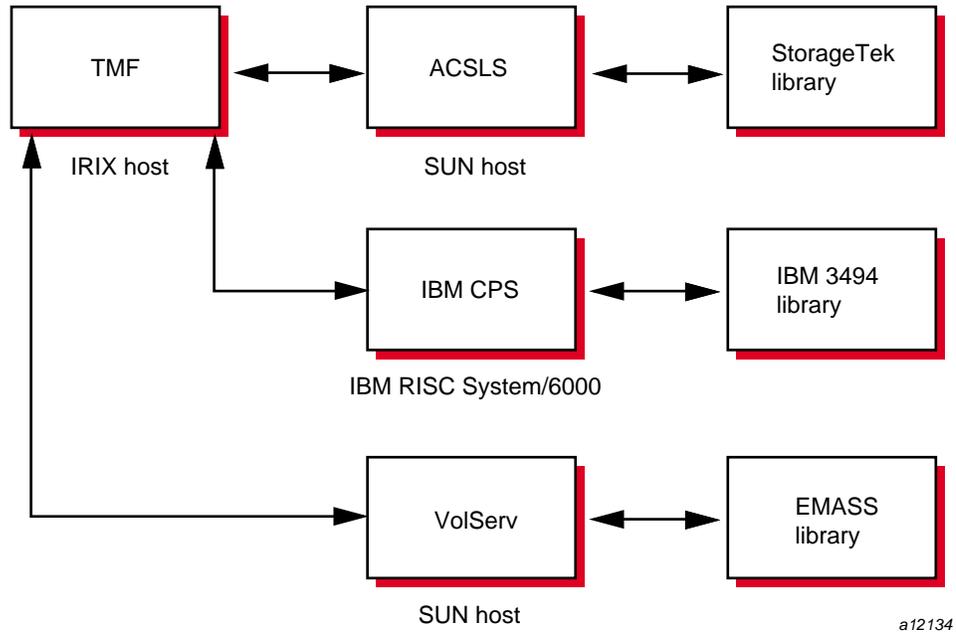
Figure 3-1 Library Communication

3.1.2 StorageTek Library

TMF supports a variety of StorageTek tape devices and libraries. For a definitive list, see the *TMF Release and Installation Guide*.

TMF communicates with the ACSLS software via a child process called stknet, which TMF starts after the library is configured up (*up* means that it is running and waiting for tape requests).

Note: Check with your StorageTek representative to validate the values of CSI_UDP_RPCSERVICE and CSI_TCP_RPCSERVICE.

3.1.3 IBM Library

TMF supports one IBM library, IBM 3494, and specific IBM tape devices. For a definitive list, see the *TMF Release and Installation Guide*.

TMF communicates with the IBM CPS software via a child process called `ibmnet`, which TMF starts after the library is configured up.

3.1.4 EMASS Library

TMF supports EMASS libraries. For a definitive list, see the *TMF Release and Installation Guide*.

TMF communicates with the vendor-supplied software interface, `VolServ`, via a child process called `esinet` which TMF starts after the library is configured up.

3.1.5 General Installation Information

The UNIX storage server host name must be defined in the TMF configuration file, the local `/etc/hosts` file. For more information, see the `hosts`(4) man page. The UNIX storage system host name also must be specified in the `server` parameter of the `LOADER` definition in the `/etc/config/tmf.config` file.

If you are using the UNIX version of the StorageTek library, you must also ensure that `CSI_UDP_RPCSERVICE` and `CSI_TCP_RPCSERVICE` are set to `TRUE` in the `/usr/ACSSS/rc.acsss` file of the UNIX storage server host. Your local StorageTek representative should be able to assist you in this matter.

It is recommended that you use the installation documentation for the libraries at your site to correctly install these products.

3.1.6 Organizing Your Devices in Attended and Unattended Modes

A *mixed environment* consists of devices serviced by a manual operator (attended mode) and devices serviced by a library (unattended mode). If TMF services mount requests in a mixed environment, you must organize the devices to use both devices and loaders in the most efficient manner possible.

A volume has a domain associated with it and, as such, has a preferred or best loader to service a mount request. If the domain of a tape cartridge is a tape vault, the best loader is an operator. If the tape cartridge resides in the library's domain (silo), the best loader is the library.

Each tape device belongs to a *device group*, which is a collection of devices with equivalent physical characteristics. Although cartridge devices can have equivalent

physical characteristics, you should consider the manner in which the devices will be serviced to determine whether or not they should be grouped.

One of the principal reasons for using a library is that the loader can be run in unattended mode (that is, without an operator). Using the library in this manner means that no imports or exports are considered, and a user-requested tape mount that cannot be satisfied by the library is canceled.

The easiest way to prevent canceled mounts is to assign the library drives to a device group different from the one serviced by manual operators. A user can then determine whether the required device group is available before requesting a tape mount. The only drawback to this method is that the user must be aware of the domain in which the tape resides and, if necessary, make changes to scripts if the domain of the tape changes.

For operations that have 24-hour operator coverage, all tape cartridges can be assigned to one device group, with the operator deciding whether the mount request should be queued or canceled, or whether the volume should be imported or exported. In this case, the user need not be concerned about the domain of the tape.

3.1.7 Accessing Tape Cartridges

Another administration issue is the accessibility of tape cartridges in a library. In the past, control of a volume serial number (VSN) was provided by an operator or by security programs on a front-end computer. With a library, control of VSNs does not exist; therefore, with the distributed TMF software, any user may request the mounting of any VSN in the domain of the library.

A site may provide access control to VSNs through two user exits. For information on user exits, see the *TMF Release and Installation Guide*.

3.2 OpenVault as a TMF Loader

You can use OpenVault, a storage library management facility, as a TMF loader and can configure it on your local IRIX host or a remote one. Figure 3-2, page 23, shows OpenVault on the same IRIX host as TMF, and Figure 3-3, page 24, shows OpenVault on a different host than TMF.

OpenVault supports a wide range of removable media libraries as well as a variety of drives associated with these libraries. The checklists and sample Perl command script in this section provide information on using OpenVault with TMF. For detailed information on using OpenVault, see the *OpenVault Operator's and Administrator's Guide*.
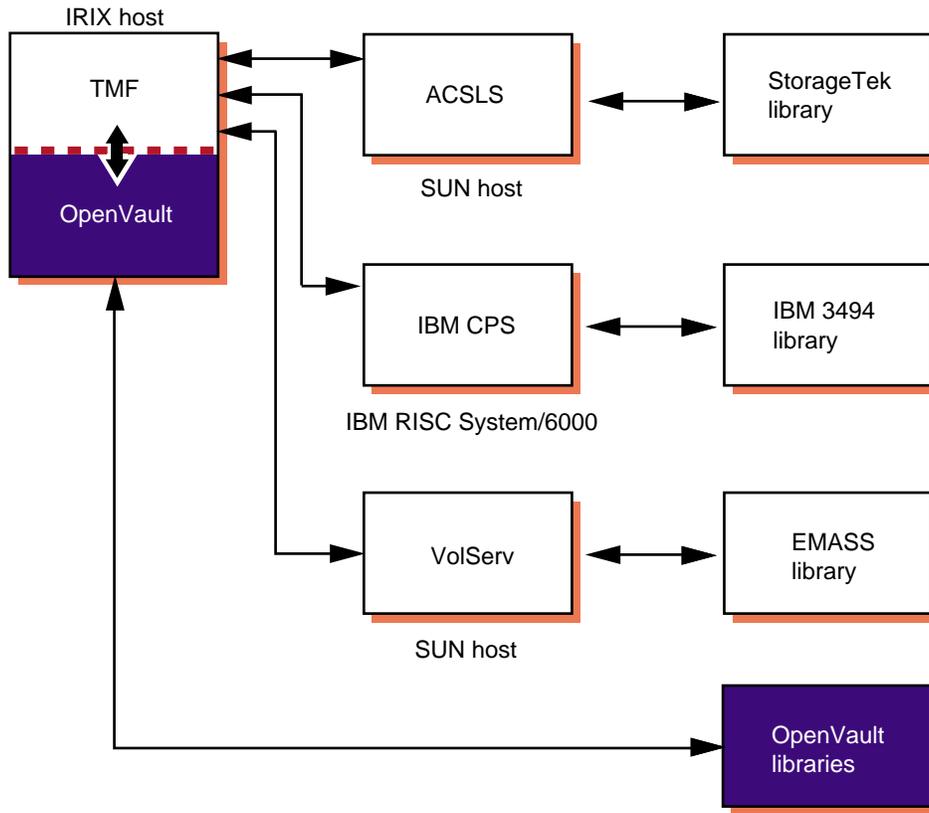
Figure 3-2 OpenVault on the Local Host
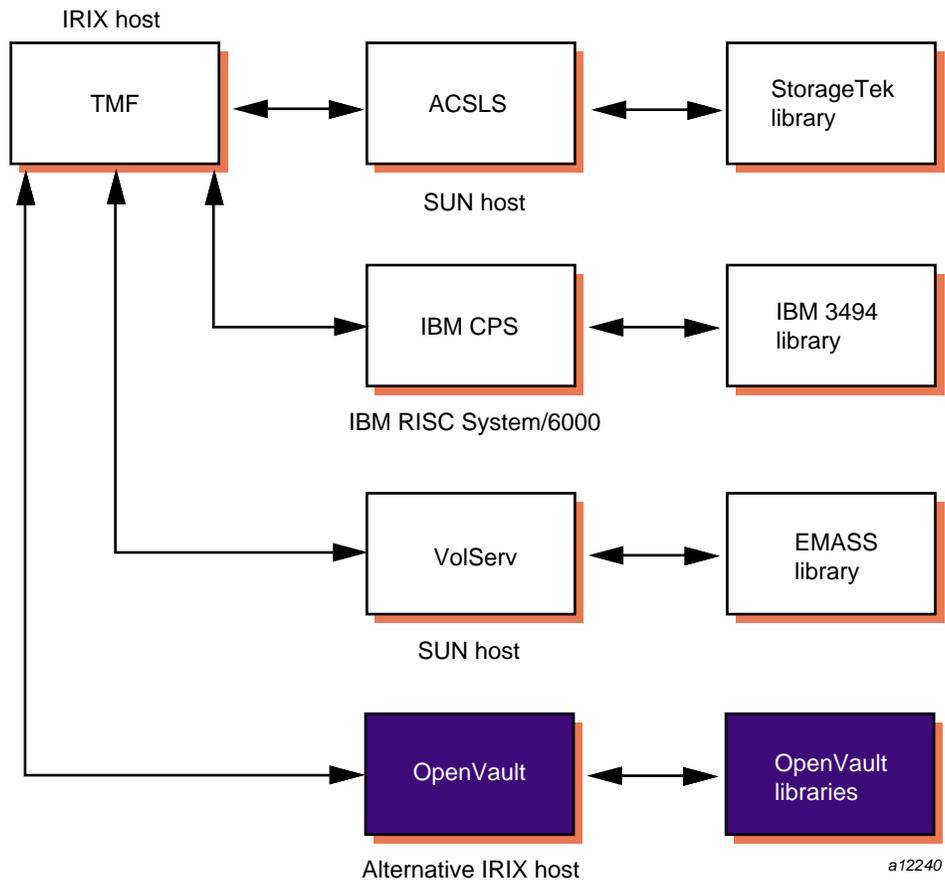
Figure 3-3 OpenVault on a Remote Host

3.2.1 Checklists

Procedure 3-1, and Procedure 3-2, page 28, list the steps you need to take before you use TMF with OpenVault.

Procedure 3-1 OpenVault Checklist

Configure the drives and libraries in OpenVault so that TMF can use them. Ensure that the following steps are taken so that TMF and OpenVault counterparts match.

1. Use the TMF name for each matching OpenVault drive.

   - In TMF, every drive belongs to a device group.

   - In OpenVault, every drive belongs to a drive group.

2. Use the TMF device group name for each matching OpenVault drive group.

   For every device group in TMF, there must be a matching drive group in OpenVault containing the same drives.

3. Make sure that the following line is in the `/usr/OpenVault/var/core_keys` file:

   server_name   *app_name*   *   *language*   *key*

   For example, if `armadillo` is the OpenVault server (*server_name*, the TMF application name (*app_name*), is `tmf`, the language is `CAPI`, and the security key (*key*) is not used (`none`), you enter the following line in the key file:

   `armadillo       tmf            *                CAPI    none`

   Currently, TMF uses `CAPI` as its language; so you must specify `CAPI` in the file.

   In OpenVault documentation, the terminology may differ: the key file is the *key authorization file*, the server (*server_name*) is a *host*, the TMF application name (*app_name*) is the *client*.

4. Verify that the application name of TMF in OpenVault can use the cartridge groups (groups of tapes) and drive groups (device groups) that TMF uses.

   - In OpenVault, more than one application may be assigned to a drive; OpenVault will use a drive only if the request comes from an assigned application.

     – In OpenVault, a drive is a TMF device.

     – Each drive is assigned to a drive group.

     – Each drive group is assigned to one or more applications (clients).

   - In OpenVault, only one application may be assigned to a cartridge; OpenVault mounts a cartridge only if the request comes from the assigned application.

     – In OpenVault, a cartridge is a physical cartridge (also called a physical tape in TMF).

    – Each cartridge is assigned to a cartridge group.

    – Each cartridge group is assigned to an application (a client).

    – A cartridge is identified by its physical cartridge label (PCL), which is used to identify a cartridge in an OpenVault loader library.

a. Use the following OpenVault commands to get the application information for the drive groups and cartridge groups:

```
ov_drivegroup -l '.*' -A '.*'
ov_cartgroup -l '.*' -A '.*'
```

b. Use the following commands with the -a option to add tmf, the default TMF application name, to the default drive group (drives) and to the default cartridge group (carts):

```
ov_drivegroup -a drives -A tmf
ov_cartgroup -a carts -A tmf
```

The following four examples illustrate step-by-step how you can use these commands to ensure that the required components in TMF and OpenVault match.

Example 3-1 Applications for All Drive Groups

The following command requests all applications of all drive groups. The output shows that there is only one drive group, SD3, with one application, ov_umsh.

```
# ov_drivegroup -l '.*' -A '.*'
application          group              group app prio  unload time
ov_umsh             SD3                1000            60
#
```

Example 3-2 Addition of TMF Application

The following command adds the tmf application to the SD3 drive group.

```
# ov_drivegroup -a SD3 -A tmf
Drive-group-application creation:
        Application: tmf
        Group: SD3
#
```

Example 3-3 Recheck: Applications of All Drive Groups

The following command requests all applications for all drive groups. This command is the same as the one in Example 3-1; the resulting output shows that `tmf` has been added as an application of the SD3 drive group.

```
# ov_drivegroup -l '.*' -A '.*'
application          group                 group app prio  unload time
ov_umsh             SD3                   1000            60
tmf                 SD3                   1000            60
#
```

Example 3-4 Applications of All Cartridge Groups

The following command requests all applications for all cartridge groups. The output shows that `tmf` is already an application of the `carts` cartridge group.

```
# ov_cartgroup -l '.*' -A '.*'
application          group                 group app prio
ov_umsh             carts                 1000
tmf                 carts                 1000
#
```

5. If you want to change a default, see the `/usr/OpenVault/etc/ov_environ` file. It contains the environment variables and the default values that you can change.

   For example, if the default drive group, cartridge group, and library names are not the names you want to use, you can change these defaults values by setting the following environment variables before you run the OpenVault `setup` command:

   ```
   export OVDEFAULTDGROUP=DLT
   export OVDEFAULTCGROUP=dlt
   export OVDEFAULTLNAME=panther
   ```

6. To collect debugging information in the `/usr/OpenVault/var/OVLOG` file, enter the following command:

   ```
   ov_msg -s -t core -m debug
   ```

Procedure 3-2 TMF Checklist

Modify the `tmf.config` file to support OpenVault in the `LOADER` statement. For information on the `tmf.config` file, see the `tmf.config`(5) man page.

1. Define a OpenVault loader:

   `type = OPENVAULT`

2. Specify where the OpenVault server is running by entering the name of the host:

   `server = ` *host_name*

3. Either use the TMF default (`tmf`) for the OpenVault application name or specify a different name for TMF with the following parameter:

   `ov_tmf_application_name = ` *tmf_application_name*

4. Specify the pathname of the key file for TMF if the OpenVault security key is used:

   `ov_tmf_keyfile = ` *keyfile_path_name*

   For more information on the key file and security key, see the *OpenVault Operator's and Administrator's Guide*.

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

For more information on `tmmnt`(1) usage, see the `tmmnt`(1) man page and the *TMF User's Guide*.

3.2.2 Perl Script for Commands

You can also use a Perl script to issue OpenVault commands shown in Example 3-5, page 29.

⚠ **Caution:** If you decide to use scripts, you should be sure they are doing what you intend.

Example 3-5 OpenVault Command Script

This script shows the drives on which you may mount a cartridge.

```perl
#! /usr/bin/perl -w        require 5.002;
use Socket;
use FileHandle;

$ov_server = `hostname`;
chop $ov_server;
$ov_port = "44444";

# Setup connection to OpenVault server process.
# See "Programming Perl" 2nd ED., page 498, for discussion
# on network programming with Perl.
$iaddr = inet_aton($ov_server) or die "no host: $ov_server";
$paddr = sockaddr_in($ov_port, $iaddr);
$proto = getprotobyname("tcp");
socket(SOCK, PF_INET, SOCK_STREAM, $proto) or die "socket: $!";
connect(SOCK, $paddr) or die "connect: $!";

SOCK -> autoflush();

# Once a connection is made to the OpenVault server
# process, it is treated just like any other file.
# Send initial data to server.
print SOCK "$ov_server\r\ntmf\r\nOnlyInstance\r\nCAPI\r\n0\r\n";

# Get response and ignore it.
# Response should only be 'ok'.
$line = <SOCK>;

# Send 'hello' greeting.
print SOCK "hello client['tmf']instance['onlyInstance']";
print SOCK "language['CAPI']versions['1.0'];\r\n";

$line = <SOCK>;
if (substr($line,0,7) eq "welcome") {
   # 'welcome' is the correct result.
}
elsif ( substr($line,0,9) eq "unwelcome") {
   # The server has rejected us.
```

```
      die "Server Not Allowing Request\n";
   }
   else {
      # Got an undefined answer from server.
      # We should not get here.
      die "Undefined Error\n";
   }

   # Send 'show' command.
   $ov_cmd = "show match[strEQ('123456' CARTRIDGE.'CartridgePCL')]
             report[DRIVE.'DriveName']
             reportmode[value]";
   $task_id = " task['66666'];\r\n";
   print SOCK $ov_cmd . $task_id;
   print $ov_cmd . $task_id;

   # Get command 'accepted' from server.
   $line = ;
   print $line;

   # Get command 'success' from server
   # along with results.
   $line = <SOCK>;
   print $line;
   # A non-trivial script would parse out the results
   # and display it in a more human readable form.

   print "Saying Goodbye to OV\n";
   print SOCK "goodbye task['1'];\r\n";

   # Get command 'accepted' from server.
   $line = ;

   # Get command 'success' from server.
   $line = <SOCK>;

   close(SOCK) or die "close: $!";
   exit;
```

3.3 Automatic Volume Recognition

Automatic volume recognition (AVR) is a TMF feature that allows TMF to recognize volumes mounted on drives prior to them actually being requested by applications, and it allows an operator to direct the mounting of tapes to specific devices.

Tape mount messages request that the operator mount a tape on a device in a device group. Upon receiving a message, you locate the tape and choose the device to be used.

The overcommit option is an extension to AVR. It allows you to set the number of outstanding mount requests to a number larger than the actual number of tape devices. It gives you additional flexibility in choosing which request to satisfy and on which device.

Note: Only those requests that cannot cause a device to deadlock are allowed into the overcommitted request process.

You may enable or disable the AVR and overcommit options on a global or on a specific device-group basis. Neither option is available to device groups that also contain devices serviced by a tape library (automatic loader).

When a device that has been configured to use AVR is configured up with the tmconfig(8) command, a child process, called tmavr, is created to monitor the device and wait for a volume to be mounted. When tmavr detects a mounted volume, the label and ring status information is sent to the TMF daemon. If tmavr cannot determine the volume label, an operator message is issued for the correct volume information to send to the TMF daemon. The child process waits for the TMF daemon to direct it to exit or look for a new volume to mount.

3.4 Message Daemon and Operator Interface

The message daemon and its associated operator interface provide mount messages for administrators and operators who are loading and unloading tapes. This section provides a brief overview of the daemon and interface.

3.4.1 Starting and Stopping the Message Daemon

You must have superuser privileges to start or stop the message daemon.

Start the message daemon prior to starting TMF by entering the following command:

/usr/tmf/bin/msgdaemon

To stop the message daemon, enter the following command:

/usr/tmf/bin/msgdstop

3.4.2 Messages

Only one message daemon can be running at any time. If you attempt to start the message daemon while it is already running, you will receive an error message.

All messages are logged by the message daemon as they are received. The logs are kept in the `msglog.log` log file in the `/var/spool/msg` directory. The `/etc/newmsglog` shell script, which resides in the `/usr/tmf/bin` directory, saves the last several versions of the log. The versions are called `msglog.log.0`, `msglog.log.1`, and so on, with `msglog.log.0` being the most recent. This script also instructs the message daemon to reopen the log file; it should be run from the `crontab(1)` command.

3.4.3 Commands

The message daemon request pipe is located in the `/var/spool/msg` directory.

Table 3-1 shows the message daemon commands and the permissions required to access them.

Table 3-1 Message Daemon Commands

Command	Permission	Description
msgdaemon(8)	Administrator	Starts the message daemon.
msgdstop(8)	Administrator	Stops the message daemon.

Command	Permission	Description
oper(8)	Administrator	Invokes the operator display; displays messages.
msgr(1)	All users	Sends action message to operator.

The operator display provided by the oper(8) command can be run from any terminal defined in the /usr/lib/terminfo file. It requires at least 80 columns and 24 lines. The three lines at the bottom of the operator display screen are used for input and for running commands that do not display information on the screen. The rest of the screen is used as a refresh display to display messages and to run other display commands.

The $HOME/.operrc configuration file lists the commands to be run as refresh displays and those that require full control of the screen. $HOME is the user's home directory. If this file does not exist, the default configuration file, /usr/tmf/*version*/oper.rc, is used.

Commands not listed in the configuration file are assumed to be nondisplay commands, which are also called action commands.

Table 3-2 describes two of the action commands available from the operator display:

Table 3-2 Operator Action Commands

Command	Description
msgd(8)	Displays action messages.
rep(8)	Replies to action messages.

Action messages that require replies from the operator are primarily tape mount messages, but they may be other types of messages to which users need responses. These messages are logged by the message daemon. An action message is deleted when the operator replies to it or the sender cancels it.

3.5 Starting and Stopping TMF

You can start and stop TMF automatically or explicitly.

3.5.1 Starting and Stopping TMF Automatically

Installing TMF does not enable starting TMF automatically at system startup. To enable automatic startup of TMF and the message daemon, execute the following chkconfig(1m) command as root:

```
chkconfig -f tmf on
```

To stop TMF from starting automatically at system startup, execute the following as root:

```
chkconfig -f tmf off
```

3.5.2 Starting and Stopping TMF Explicitly

If you chose not to use the chkconfig(1m) command, you can start and stop TMF with the tmdaemon(8) and tmstop(8) commands. You can also use these commands to stop and start TMF once it has been started automatically when the system is booted.

To start TMF explicitly, enter the following tmdaemon(8) command:

/usr/tmf/bin/tmdaemon

Options exist for the tmdaemon(8) command. For descriptions of these, see the tmdaemon(8) man page.

TMF is stopped by the following tmstop(8) command:

/usr/tmf/bin/tmstop

The tmstop(8) command has no options.

Table 3-3 shows these commands and the permissions required to access them.

Table 3-3 TMF Commands

Command	Permission	Description
tmdaemon(8)	Administrator	Starts TMF.
tmstop(8)	Administrator	Stops TMF.

3.6 Using `xfsdump` and `xfsrestore`

When you use the `xfsdump`(1m) command to dump files to tape, the command 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.

When you are using TMF, `xfsdump`(1m) knows nothing about end of volume. If you expect the dump to occupy more than one tape volume, you must specify the volumes on the `tmmnt`(1) command with the `-v` option. If you specify multiple volumes, you do not really know how many `xfsdump`(1m) will use.

If you do not specify enough volumes to hold the dump, you will receive an error message. If this happens, you can restart the dump by issuing another `tmmnt`(1) command with the `-b` option set to 262144 and with additional volumes specified on the `-v` option. Then you enter a `xfsdump`(1m) command with the `-R` option to resume the interrupted dump session.

To restore tape files from dumps produced by `xfsdump`(1m), use the `xfsrestore`(1m) command.

Troubleshooting

This chapter describes the following troubleshooting topics:

- Addressing drive, job, and daemon issues

- Using tracing

- Resolving common problems

4.1 Addressing Drive, Job, and Daemon Issues

Occasionally, you may experience problems with the hardware or the software while running magnetic tapes. If so, there are certain steps you should take to try to clear the user, job, tape drive, or the TMF daemon itself. This section describes those steps and identifies TMF daemon files that you may encounter.

4.1.1 Tape Drive or Job Problems

If a tape drive appears to be hung, but the TMF daemon is still responding to commands such as `tmstat`(1) and `tmgstat`(8), you can use the `tmfrls`(8) command to clear the user's tape reservation. If this method does not work, try the `tmclr`(8) command.

If the problem appears to be hardware related, free the user by the preceding method (check the result with the `tmstat`(1) command). Then configure the drive down with the `tmconfig`(8) command, and discuss the problem with the appropriate hardware personnel.

4.1.2 TMF Daemon Problems

If the TMF daemon (see `tmdaemon`(8)) is hung (that is, no tapes are moving nor are there any responses from any tape commands), you must take the TMF daemon down. First try the `tmstop`(8) command. If this command does not work, determine the process identifier of the TMF daemon (by using the `ps`(1) command), and enter the following `kill`(1) command:

```
kill -2 pid
```

The *pid* argument of the `kill`(1) command is the process identifier of `tmdaemon`(8). If the previous command does not work, enter the following:

```
kill -9 pid
```

4.1.3 Pertinent TMF Files

A number of files throughout the system relate to tapes. This section deals with those files specific to the TMF daemon.

All of the following TMF commands and processes reside in the `/usr/tmf/bin` directory:

- Commands: `tmcatalog`(1), `tmclr`(8), `tmcollect`(8), `tmconfig`(8), `tmdaemon`(8), `tmfrls`(8), `tmgstat`(8), `tmlabel`(8), `tmlist`(1), `tmmls`(8), `tmmnt`(1), `tmmql`(8), `tmrls`(1), `tmrst`(1), `tmrsv`(1), `tmset`(8), `tmstat`(1), `tmstop`(8), `tmunld`(8)

- Processes: `esinet`, `fesdex`, `fesnet`, `ibmnet`, `stknet`, `tmavr`, `tmdaemon`, `tmssp`

During the course of its activity, the TMF daemon and its components write a number of trace files, which are located in the `/usr/spool/tmf/trace` directory. Table 4-1, page 39, describes this subset, and Example 4-1, page 39, shows how you use the `tmstat`(1) command to identify a tmf*xxx* file.

Table 4-1 TMF Trace Files

File	Description
avr_*device_name*	Each tmavr process records events in a trace file based on the device name it is monitoring. If AVR is active for the s4781s0 device, the relevant trace entries for the tmavr process are in avr_s4781s0.
daemon	This file contains all activity traced by the TMF daemon. It is the main TMF daemon trace file. The tmset(8) command must be issued with the -T option set to off in order to disable traceing and may impact problem diagnosis as minimal tracing may not provide enough information to resolve problem situations. A site must weigh the benefits of disabling tracing against the potential drawbacks. Disabling tracing does enable the TMF daemon to run more efficiently.
daemon.stdout, daemon.stderr	These files contain any information that goes to standard output or error. They are in the /usr/spool/tmf directory. The daemon.stderr file is especially helpful in tracking down problems as it contains error messages as well as informational messages pertaining to various administrative commands.
ldrname	Each media loader also has its own trace file. The name of this file corresponds to the loader name as defined in the tmf.config file.
tmf*xxx*	Once a tape is assigned a drive, subsequent traces specific to that process are logged in a tmf*xxx* file. The final three characters of the trace file can be determined from the stm field of the tmstat(1) command.

Example 4-1 tmstat Output

In this tmstat output, the traces for drive s4781s0 are in the tmf002 file. Leading zeros are added to the stream number to make it a 3-character number to create the tmf*xxx* file name.

```
armadillo%>tmstat
          user     sess    group    a stat device   stm rl ivsn    evsn    blks
                           STK9490  - idle s9490s4

                           STK9490  - idle s9490s1

          bar      3854    STK4781  - assn s4781s0    2 is 002335 002335     1
                           STK4781  - idle s4781s1
```

```
STK4781 – idle s4781s2

STK4781 – idle s4781s3

STK4781 – idle s4781s4
```

In addition, communication pipes are maintained within the `/usr/spool/tmf` directory. If the TMF daemon abnormally terminates, its core file is also saved in the directory.

The message daemon logs can provide insight into tape problems. These log files are generally saved and maintained in the `/usr/spool/msg` directory. All operator interaction is saved in the `msglog.log` file. In addition, a debug log for the message daemon is in the `dbglog.log` file.

4.2 Using Tracing

Using tracing can help identify and resolve tape problems. The `tmcollect`(8) utility enables you to collect the trace information needed.

4.2.1 `tmcollect` Utility

The `tmcollect`(8) utility collects TMF information. A user with `root` permission may run this script when a tape-related problem occurs. The information is placed in a separate directory so that it can be easily packaged and shipped for offline analysis. For the collected information to be of optimal use, TMF tracing should be enabled. For more information about this administration command, see the `tmcollect`(8) man page.

Before anything is copied to the information directory, the `tmcollect`(8) utility attempts to determine whether the TMF daemon is in its normal state, and if not, runs a few checks for known hang situations.

The `tmcollect`(8) utility should be executed to gather information once trouble with the TMF daemon is suspected prior to attempting to terminate the TMF daemon.

4.2.2 Tracing

TMF tracing is turned on by default. All child processes created by the TMF daemon have tracing enabled. While tracing is a very important tool for debugging TMF problems, it uses additional CPU time. Tracing can be turned on and off by issuing the tmset(8) command. To turn tracing off, enter the following command:

```
tmset -T off
```

To turn tracing on, enter the following command:

```
tmset -T on
```

If the stability of TMF at a site has been established, tape tracing may be unnecessary overhead. The CPU cycles saved by turning tracing off depends on the mix of jobs submitted, because some tape operations generate more trace information than others.

When tracing is turned off, the TMF daemon and its child processes still trace entry to and exit from child processes and abnormal termination of tape processes. Abnormal terminations include those induced by the operator and terminations caused by errors within TMF. A tape mount request canceled by an operator or interrupted user job is considered an abnormal termination induced by the operator.

The option of turning TMF tracing off allows sites at which TMF is stable to reduce substantially the system and user time used by the TMF daemon. This gain in system and user time must be weighed with the knowledge that some error information and all trace information will be lost in case of a TMF daemon problem.

The only way to analyze a problem is to turn tracing on, resubmit the job, and collect traces when the problem reappears.

4.2.3 Sample Trace Analysis

To obtain a complete picture of a problem, save trace information as soon as possible after you identify an error situation. You can use the tmcollect(8) utility to aid in the data gathering process.

This utility saves all the pertinent trace files in /var/spool/tmf. If the TMF daemon is not hung, the TMF command output is also saved. When you execute the utility, you are asked to comment on how the system was behaving at the time tmcollect(8) was run.

All of the trace files are circular. For instance, if a particular tape drive is hung, by the time it is noticed the TMF daemon trace has probably been overwritten. However, the

device trace should provide some useful information. By default, the device traces are 409600 bytes in length while the daemon file is 10 times that value (the default is 4096000 bytes). You can configure this parameter by specifying the trace_file_size option in the OPTIONS statement in the TMF configuration file. For more information, see the tmf.config(5) man page.

Each time a TMF daemon routine is entered, tracing for that routine begins. Additional tracing may also exist which provides more information for software engineering in case problems occur. By using this information, the paths that the software took to perform various tape functions can be followed.

Information is also written into the respective TMF daemon device traces (tmf*xxx*). In addition, there are trace files for esinet, stknet, and ibmnet. By using all of the appropriate traces, you can obtain the entire picture of what was happening when a failure occurred.

Example 4-2 shows the information you can obtain from a trace line.

Example 4-2 Trace Lines

This example identifies and describes each trace line segment.

```
10:59:58 151257598.1241 1450 tmmsp media_select function entered
^^^^^^^^ ^^^^^^^^^^^^^^ ^^^^ ^^^^^ ^^^^^^^^^^^^ ^^^^^^^^^^^^^^^^.......
AAAAAAAA BBBBBBBBBBBBBB CCCC DDDDD EEEEEEEEEEEE FFFFFFFFFFFFFFFF.......
```

The fields in this line are labeled as follows:

Field	Description
A	References the wall clock time. Having this time available is helpful in relating events in one trace to other traces, console messages or daemon.stderr messages.
B	References the real time clock. You use this time when timing issues are more important. It helps to determine whether the events truly took place in the proper order.
C	References the process number of the main routine. In the daemon file, this value will invariably be tmdaemon(8); in the tmf*xxx* files, the value will be the particular child tmdaemon(8) forks off to process the request (for example, tmmsp).
D	Identifies the main routine.

| E | References the particular routine called by the main routine. |
| F | Provides detailed trace information about the entry. |

4.3 Resolving Common Problems

This section identifies some common tape problems that you may encounter and some possible solutions.

4.3.1 TM003 - Resource *group_name* is not available

This error indicates that you issued a `tmrsv`(1) command for a device group that does not exist, or that you attempted to reserve more devices than are currently configured up.

4.3.2 TM060 - Waiting for device *device_name*

This message is returned when a `tmmnt`(1) command has been issued, but has not yet been satisfied because a requested device type is not available. The command will be satisfied once a device is made available either by the operator configuring one up or by a currently running job releasing its resources.

4.3.3 TM064 - File *file_name* could not be found on volume *vsn*

This error is returned when the file specified with the `-f` parameter on the `tmmnt`(1) command (or `-p` if `-f` is not present) does not exist on a labeled tape. When a labeled tape is created, the lower 17 characters specified by the `-f` (or `-p`) parameter are written into the HDR1 label. Subsequent attempts to read that tape file must include the correct file identifier. The file identifier is not checked if the `check_file_id` option is set to `NO` in the `tmf.config` file.

Man Page List

This appendix provides a list of the TMF man pages. Man pages exist for the user commands, devices (special files), file formats, miscellaneous topics, and administration 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 usage details (command syntax, parameters, and so on).

The following five categories of man pages exist:

- User commands

  msgr(1)
  tmcatalog(1)
  tmlist(1)
  tmmnt(1)
  tmrls(1)
  tmrst(1)
  tmrsv(1)
  tmstat(1)

- Devices (special files)

  tmfdaem(4)

- File formats

  tmf.config(5)
  tmfctl(5)
  tmftrace(5)

- Miscellaneous topics

  `tmf`(7)

- Administration commands

  `msgd`(8)
  `msgdaemon`(8)
  `msgdstop`(8)
  `newmsglog`(8)
  `oper`(8)
  `rep`(8)
  `tmclr`(8)
  `tmcollect`(8)
  `tmconf`(8)
  `tmconfig`(8)
  `tmdaemon`(8)
  `tmfrls`(8)
  `tmgstat`(8)
  `tmlabel`(8)
  `tmmls`(8)
  `tmmql`(8)
  `tmset`(8)
  `tmstop`(8)
  `tmunld`(8)

`tmf.config` Man Page

The `tmf.config`(5) man page is provided below for your reference during the initial system startup. The EXAMPLES section is also shown in Example 2-1, page 5.

For information on displaying and printing the `tmf.config`(5) page once your system is running, see Appendix A, page 45.

```
tmf.config(5)                                               tmf.config(5)


NAME
     tmf.config - TMF configuration file


IMPLEMENTATION
     SGI IRIX systems licensed for the Tape Management Facility (TMF)


DESCRIPTION
     The system uses a Tape Management Facility (TMF) configuration file
     named tmf.config in the/etc/config directory.  This file defines all
     of the tape devices that the system uses.

     The TMF configuration file consists of comments (optional) and
     statements.  A comment begins with the # symbol and continues to the
     end of line.  A statement consists of a name followed by a list of
     keyword parameters.  There are four statements; one of these
     statements also consists of substatements.  Statements must be in the
     order shown:

     1. LOADER statements (one per loader)

     2. DEVICE_GROUP statements (one per device group)

     3. AUTOCONFIG statement (one per system)

        The AUTOCONFIG statement consists of DEVICE statements.

             DEVICE statements (one per device)

             DEVICE statements define devices that TMF will control and
             that are automatically configured during the system boot.
```

4. OPTIONS statement (one per system)

Statement Syntax Rules
  The following syntax rules apply to tmf.config statements:

* The statement name and its parameters are separated by one or more
  white spaces (blank, tab, or newline characters).

* Adjacent parameters are separated by a comma.

* The end of the parameter list is indicated by the absence of a
  comma.

* Adjacent statements are separated by one or more white spaces.

The following is a list of keyword parameter syntax rules:

* The keyword is separated from its value by the = symbol.

* The value of a keyword may consist of keywords, numbers, character
  strings, and lists of keywords, numbers, and character strings.

* If the value of a keyword is a list, the list is enclosed within
  left and right parentheses. Adjacent elements of a list are
  separated by a comma. If the list consists of one element, you do
  not have to enclose it in parentheses. The elements of a list may
  be lists.

* Numbers may be specified in decimal, octal, and hexadecimal formats.
  These formats are the same as those used in the C programming
  language:

    Decimal   First digit is not 0 (1372)
    Octal   First digit is 0 (0563)
    Hexadecimal First 2 characters are either 0x or 0X (0xf2)

* Character strings are series of characters. If any one of the
  special characters (white space, ", #, =, {, }, (,), ', \ ) is
  needed in the string, you must enclose the string in a pair of
  double quotation marks ("). Within a pair of double quotation

marks, the sequence of characters will be replaced by x; x is any character. This is the only way you can specify a " and a \ in a quoted string.

* Comments may appear between any symbols described previously.

You can code the names of statements and keywords in a mixture of uppercase and lowercase letters. The values specified by the user is case sensitive. The following specify the same thing:

    Name = A
    name = A

The following are different:

    name = A
    name = a

The following are descriptions of the tape configuration statements. You must specify a value for each parameter unless a default is specified or the parameter is described as optional.

LOADER Statement
  The LOADER statement identifies the loaders in the tmf.config file and has the following format:

      LOADER parameter_list

  A description of the parameters follows:

Parameter   Description

loader_ring_status = status   Specifies whether the loader
                                    is alerted to ring status.

                                    ALERT   Alerts loader
                                                to the ring
                                                status when a
                                                tape is
                                                mounted, and
                                                checks that the

```
                                        ring status
                                        matches the
                                        ring status
                                        requested by
                                        the tape user.

                             IGNORE      Ignores the
                                         ring status
                                         when a tape is
                                         mounted.  A
                                         logical ring
                                         out status is
                                         used for a tape
                                         that has been
                                         requested with
                                         a ring out
                                         status, but its
                                         actual ring
                                         status is ring
                                         in.  The
                                         default is
                                         ALERT.

message_path_to_loader = path      Specifies the message path to
                                   the servicing loader.

                             MSGDAEMON   Uses message
                                         daemon to send
                                         message to
                                         loader.

                             NETWORK     Uses TCP/IP
                                         protocol to
                                         send message to
                                         loader.

message_route_masks = location     Routes mount request messages.
                                   You can route the mount
                                   request message to multiple
                                   locations. The list may
                                   consist of the following:
```

	FRONTEND	Issues the mount message to the front end that may be reached through a TPC/IP connection.
	SERVER	Issues the mount message to the server station.
	IRIX	Issues the mount message to the message daemon. For more information, see msgdaemon(8).
mode = value	Specifies attended mode:	
	ATTENDED	Prompts for operator intervention.
	UNATTENDED	Assumes negative response for operator intervention.
name = name	Specifies the loader name, which is the object of several tmconfig(8) requests.	
network_retry_tries = number	Specifies the number of times the TMF loader child program	

attempts to send a request
over the network to the server
after an initial attempt
fails. The default for each
child program is 5.

network_send_timeout = number   Specifies the time in seconds
                                       during which the TMF loader
                                       child program tries to send a
                                       request over the network to
                                       the server. The default for
                                       each child program is 3
                                       seconds.

ov_tmf_application_name = tmf_application_name
                                       Specifies the OpenVault
                                       application name for TMF. The
                                       default is tmf.

ov_tmf_keyfile = keyfile_path_name   Specifies the pathname of the
                                       OpenVault key file for TMF.
                                       You specify this parameter
                                       only if your site is using the
                                       OpenVault security key.

                                       The key file specifies the
                                       security key for TMF when it
                                       initiates a session with
                                       OpenVault.

                                       For example, if OpenVault is
                                       running on citron, the
                                       application name of TMF is tmf
                                       and the security key is
                                       Zyh3wi, the key file contains
                                       the following line:

                                           citron tmf * CAPI Zyh3w

                                       For more information on the
                                       key file and security key, see

                                        the OpenVault Operator's and
                                        Administrator's Guide.

queue_time = seconds   Each volume has a designated
                                        "best" loader type for the
                                        tape mount. If the best
                                        loader is not available, this
                                        time is used to queue the tape
                                        mount request and to wait for
                                        the best loader to become
                                        available. If the best loader
                                        does not become available
                                        during this time, the mount
                                        request will be issued to the
                                        next best loader.

                                        A value of 0 indicates to wait
                                        up to 24 hours; a nonzero
                                        value specifies the number of
                                        seconds to wait.

return_host = host_name   Specifies the name of the IRIX
                                        host that serves as the return
                                        address for the server. This
                                        parameter is only used if it
                                        is set; there is no default.

scratch_volume_label_type = scratch_type
                                        Specifies the types of scratch
                                        requests that the loader may
                                        process. If you specify
                                        OPERATOR for the type
                                        parameter on the LOADER
                                        statement, the following types
                                        of scratch requests are
                                        available. If you specify any
                                        other loader type for the type
                                        parameter, only NONE is valid.

                                        AL   ANSI labeled scratch
                                                  tape requests.

	NL Nonlabeled scratch tape requests.
	NONE Scratch labels cannot be used.
	SL IBM standard labeled scratch requests.

server = server_name
: Specifies the server name.

  If you specify OPENVAULT as type, the server name is the name of the host where OpenVault is running.

server_reply_wait_time = number
: Specifies the time in seconds during which a request that is being processed by the server is kept in a queue by the TMF loader child program. If a reply has not been received within this time, the child program queries the state of the outstanding request.

  The default value for each child program is 180 seconds. For a StorageTek library, this value is multiplied by the number of Library Storage Modules in the Automated Cartridge System.

status = status
: Specifies the status (UP or DOWN) of the loader when TMF starts.

type = type
: Specifies the loader type. Currently, supported types are

as follows:

EMASS   An EMASS Grau
                 library running
                 VolServ is
                 used.

IBMTLD   IBM 3494 Tape
                 Library
                 Dataserver is
                 used.

OPERATOR   Operator loads
                 the drive.

STKACS   A StorageTek
                 library that is
                 supported by a
                 SGI system
                 running the
                 IRIX operating
                 system is used.

OPENVAULT   OpenVault, a
                 storage library
                 management
                 facility, is
                 used.

You must specify at least one
OPERATOR type loader in the
TMF configuration file. If
the file does not contain such
an entity, TMF in its
initialization process creates
one assuming the following
values:

```
LOADER
name = Operator ,
type = OPERATOR ,
```

```
                              status = UP ,
                              message_PATH_TO_LOADER = MSGDAEMON ,
                              server = "" ,
                              scratch_volume_label_type = NONE ,
                              queue_time = 1 ,
                              verify_non_label_vsn = YES ,
                              message_route_masks = IRIX ,
                              mode = ATTENDED ,
                              return_host = ""
```

verify_non_label_vsn = value   Specifies whether the nonlabel
                                      VSN should be verified. value
                                      may be either YES or NO.

DEVICE_GROUP Statement

The DEVICE_GROUP statement has the following format:

      DEVICE_GROUP parameter_list

A description of the parameters follows:

Parameter   Description

avr = value   Specifies whether this device
                                      group uses the automatic
                                      volume recognition (AVR)
                                      feature of TMF. To do so, the
                                      device group must be
                                      associated with the OPERATOR
                                      type in the LOADER statement.
                                      This optional parameter may be
                                      either YES or NO; omission
                                      implies NO.

name = name   Specifies the device group
                                      name.

overcommit = value   Specifies whether this device
                                      group uses the overcommit
                                      feature. This optional
                                      parameter may be either YES or

                                        NO; omission implies NO.

AUTOCONFIG Statement
  The AUTOCONFIG statement is made up of DEVICE statements, one for each
  device in the system.

DEVICE Statement
  The DEVICE statement specifies the characteristics of a device and has
  the following format:

       DEVICE parameter_list

  A description of the parameters follows:

  Parameter   Description

  device_group_name = device_group_name   Specifies the name of the
                                       device group defined by a
                                       DEVICE_GROUP statement.

  file = file   Specifies the path name of the
                                       device specific file.

  loader = loader   Specifies the loader name
                                       defined in a LOADER statement.

  name = name   Specifies the device name.

  status = status   Specifies the initial status
                                       (UP or DOWN) of the device.

  vendor_address = vendor_address   Specifies the vendor address
                                       of the drive in a library.

                                       The format for a StorageTek
                                       drive is as follow:

                                            acs#,lsm#,panel#,drive#

                                       The format for an EMASS Grau
                                       VolServ drive is as follows:

```
                                              drive#
```

OPTIONS Statement

  The options in force when TMF is built are specified in the
  /usr/include/tmf/tmfdefaults.h file. You can specify most of these
  options in the OPTIONS section of the tmf.config file.

  To override the value with which TMF was built, specify the following
  options and their corresponding values. The options that you can
  specify in the tmf.config file with the OPTIONS statement are similar
  to the options in tmfdefaults.h, but not identical. Values are often
  given in a different form in the two files (for example, the value for
  the ask_blp keyword is expressed as 0 or 1 in tmfdefaults.h, but it is
  expressed as YES or NO in tmf.config).

  The format of the OPTIONS statement follows:

     OPTIONS parameter_list

  The following parameter list includes valid values or brief
  definitions of the options.

Parameter	Description
ask_label_switch = value	Seeks permission (YES or NO) from the operator to switch label type. Default: YES
ask_vsn = value	Seeks permission (YES or NO) from the operator to specify a VSN when a nonlabel tape is mounted. Default: YES
blp_ring_status = value	Specifies the user status for the use of the -r option of the tmmnt(1) command when the user requests bypass label

<table>
<tr><td></td><td>processing. UNRESTRICTED specifies the user can use both -r in and -r out. OUT specifies the user can use only -r out. Default: UNRESTRICTED.</td></tr>
<tr><td>blocksize = size</td><td>Specifies the maximum block size to use when the user does not specify a maximum block size by using the tmmnt(1) command -b option. Default: 32768</td></tr>
<tr><td>check_expiration_date = value</td><td>Specifies whether the operator should check and confirm (YES or NO) the expiration date on the header label of a labeled tape. Default: YES</td></tr>
<tr><td>check_file_id = value</td><td>Specifies whether the file identifier on a labeled tape should be checked (YES or NO) when the file is opened. Default: YES</td></tr>
<tr><td>check_protection = value</td><td>Specifies whether the protection flag on the header should be checked (YES or NO). Default: NO</td></tr>
<tr><td>check_vsn = value</td><td>Specifies whether the VSN on labeled tapes should be checked (YES or NO). Default: YES</td></tr>
<tr><td>device_group_name = name</td><td>Specifies the default</td></tr>
</table>

	device group name if it is not specified on the -g option of the tmmnt(1) command. Default: CART
fes_daemon_frontend_id = identifier	Specifies the front-end identifier of the TCP daemon. Default: " "
fes_daemon_socket_port_number = number	Specifies the socket port number of the TCP daemon. Default: 1167
file_status = status	Specifies the file status (NEW or OLD) if it is not specified on the tmmnt(1) command. Default: OLD
label_type = type	Specifies the label type (AL, SL, or NL) if it is not specified on the tmmnt(1) command. Default: AL
loader_device_assignment_order = method	Specifies the method (DEVICE_LIST or ROUND_ROBIN) with which the loader assigns devices. Default: ROUND_ROBIN
max_number_of_tape_users = number	Specifies the maximum number of tape users. Default: 100
number_of_autoloader_retries = number	Specifies the number of times to try to send a request to the library before informing the operator of an error. Default: 10

```
operator_message_destination = value          Specifies where operator
                                              messages are sent; IRIX,
                                              SERVER, and FRONTEND.
                                              Default:  (IRIX)

operator_message_frontend_id = identifier     Specifies the front-end
                                              identifier for operator
                                              messages.  Default:  " "

overcommit_max = value                        Specifies the maximum
                                              number of overcommitted
                                              mount requests that TMF
                                              can issue.  When the
                                              number of tape mount
                                              requests exceeds this
                                              number, the system stops
                                              processing requests until
                                              one or more of the
                                              already overcommitted
                                              mount requests are
                                              satisfied.  To change
                                              this setting, see the
                                              tmset(8) command.
                                              Default: 20

retention_period_days = days                  Specifies the retention
                                              period (in days).
                                              Default:  0

ring_status = status                          Specifies the ring status
                                              when the ring option (-r)
                                              is not specified on the
                                              tmmnt(1) command (IN,
                                              OUT, or (IN,OUT)).
                                              Default:  (IN,OUT)

scratch_volume_retries = number               Specifies the number of
                                              retries to get a scratch
                                              volume mounted.  Default:
                                              3
```

scratch_volume_vsn = vsn   Specifies the scratch
                                              tape VSN. Default:
                                              ??????

servicing_frontend_id = identifier   Specifies the servicing
                                              front-end identifier to
                                              use when the -m option is
                                              missing on the tmmnt(1)
                                              command. Default: " "

servicing_frontend_mandatory = value   Specifies whether the
                                              front-end identifier
                                              specified by the
                                              servicing_frontend_id
                                              parameter is used (YES or
                                              NO) regardless of the -m
                                              option on the tmmnt(1)
                                              command. Default: NO

system_code = value   Specifies the system code
                                              to put on tape labels.
                                              Default: SGI/IRIX

tmf_major = number   Specifies the major
                                              device number of the TMF
                                              driver. Default: 261

trace_file_group_id = identifier   Specifies the group
                                              identifier of the TMF
                                              trace files. Default: 3

trace_file_mode = mode   Specifies the file mode
                                              of the TMF trace files.
                                              Default: 0640

trace_file_owner = identifier   Specifies the owner
                                              identifier of the TMF
                                              trace files. Default: 0

trace_directory = value   Specifies the TMF trace

	file prefix. Default: /var/spool/tmf/trace
trace_file_size = size	Specifies the size (in bytes) of the TMF trace files. Default: 409600
trace_state = value	Specifies whether tape tracing is enabled (ON or OFF). Default: ON
trace_save_directory = value	Specifies the prefix to the TMF save files. Default: /var/spool/tmf/trace_save
user_exit_mask = value	Enables the use of the listed user exits. If no user exits are required, this entry is not needed. For a list of user exits, see the IRIX TMF Release and Installation Guide. Default: UEX_NONE
verify_scratch_vsn = value	Indicates (YES or NO) that you may need to send the operator a message that requests verification that a scratch tape is being used to satisfy a tape mount request. You must consult the operator if front-end servicing is not in use. Default: YES

EXAMPLES

The following example shows the sample tmf.config file that is shipped with TMF.

```
#
#        TAPE MANAGEMENT FACILITY CONFIGURATION FILE
#
#
#

LOADER
       name = operator ,
       type = OPERATOR ,
       status = UP ,
       mode = ATTENDED ,
       message_path_to_loader = MSGDAEMON ,
       server = IRIX ,
       scratch_volume_label_type = (AL,NL,SL) ,
       queue_time = 0 ,
       verify_non_label_vsn = YES ,
       message_route_masks = (IRIX) ,
       loader_ring_status = ALERT

LOADER
       name = wolfy ,
       type = STKACS ,
       status = DOWN ,
       mode = ATTENDED ,
       message_path_to_loader = NETWORK ,
       server = wolfcreek ,
       scratch_volume_label_type = NONE ,
       queue_time = 15 ,
       verify_non_label_vsn = NO ,
       message_route_masks = (IRIX) ,
       loader_ring_status = IGNORE

LOADER
       name = panther ,
       type = STKACS ,
       status = DOWN ,
       mode = ATTENDED ,
       message_path_to_loader = NETWORK ,
       server = stk9710 ,
       scratch_volume_label_type = NONE ,
```

```
        queue_time = 15 ,
        verify_non_label_vsn = NO ,
        message_route_masks = (IRIX) ,
        loader_ring_status = IGNORE

LOADER
        name = esys ,
        type = EMASS ,
        status = DOWN ,
        mode = ATTENDED ,
        message_path_to_loader = NETWORK ,
        server = esisun ,
        scratch_volume_label_type = NONE ,
        queue_time = 15 ,
        verify_non_label_vsn = NO ,
        message_route_masks = (IRIX) ,
        loader_ring_status = IGNORE

LOADER
        name = tmfov ,
        type = OPENVAULT ,
        server = armadillo ,
        status = down ,
        mode = ATTENDED ,
        message_path_to_loader = NETWORK ,
        ov_tmf_application_name = tmf,
        scratch_volume_label_type = NONE ,
        queue_time = 15 ,
        verify_non_label_vsn = NO ,
        message_route_masks = (IRIX) ,
        loader_ring_status = IGNORE


DEVICE_GROUP
         name = CART
         avr  = YES

DEVICE_GROUP
         name = DLT

DEVICE_GROUP
```

```
          name = EMASS

     DEVICE_GROUP
          name = STK9490

     AUTOCONFIG
     {
          DEVICE
                  name    = t1 ,
                  device_group_name = CART ,
                  file    = /hw/tape/tps3d1 ,
                  status = DOWN ,
                  loader = wolfy ,
                  vendor_address = (0,0,1,1)
          DEVICE
                  name    = t4 ,
                  device_group_name = CART ,
                  file    = /hw/tape/tps3d4 ,
                  status = DOWN ,
                  loader = wolfy ,
                  vendor_address = (0,0,1,0)
          DEVICE
                  name    = dlt2 ,
                  device_group_name = DLT ,
                  file    = /hw/tape/tps5d2 ,
                  status = DOWN ,
                  loader = panther ,
                  vendor_address = (1,0,2,0)
          DEVICE
                  name    = dlt3 ,
                  device_group_name = DLT ,
                  file    = /hw/tape/tps5d3 ,
                  status = DOWN ,
                  loader = panther ,
                  vendor_address = (1,0,2,1)
           DEVICE
                  name    = ed0 ,
                  device_group_name = EMASS ,
                  file    = /hw/tape/tps10d0 ,
                  status = DOWN ,
                  loader = esys ,
```

```
                     vendor_address = (1)
           DEVICE
                     name    = s9490s4 ,
                     device_group_name = STK9490 ,
                     file    = /hw/tape/tps22d4 ,
                     status = down ,
                     vendor_address = (0,0,1,0),
                     loader = tmfov
           DEVICE
                     name    = s9490s1 ,
                     device_group_name = STK9490 ,
                     file    = /hw/tape/tps22d1 ,
                     status = down ,
                     vendor_address = (0,0,1,1),
                     loader = tmfov

    }
    OPTIONS
    ask_label_switch                        = YES ,
    ask_vsn                                 = YES ,
    blocksize                               = 32768 ,
    blp_ring_status                         = UNRESTRICTED ,
    check_expiration_date                   = YES ,
    check_file_id                           = YES ,
    check_protection                        = YES ,
    check_vsn                               = YES ,
    device_group_name                       = CART ,
    fes_daemon_frontend_id                  = "mvs" ,
    fes_daemon_socket_port_number           = 1167 ,
    file_status                             = OLD ,
    label_type                              = AL ,
    loader_device_assignment_order          = ROUND_ROBIN ,
    max_number_of_tape_users                = 100 ,
    number_of_autoloader_retries            = 10 ,
    operator_message_destination            = (IRIX) ,
    operator_message_frontend_id            = "" ,
    overcommit_max                          = 20
    retention_period_days                   = 0 ,
    ring_status                             = (IN,OUT) ,
    scratch_volume_retries                  = 0 ,
    scratch_volume_vsn                      = ?????? ,
```

```
    servicing_frontend_id                 = "" ,
    servicing_frontend_mandatory          = NO ,
    system_code                           = SGI/IRIX ,
    tmf_major                             = 261 ,
    trace_file_group_id                   = 3 ,
    trace_file_mode                       = 0640 ,
    trace_file_owner                      = 0 ,
    trace_directory                       = /var/spool/tmf/trace ,
    trace_file_size                       = 409600 ,
    trace_state                           = ON ,
    trace_save_directory                  = /var/spool/tmf/trace-save ,
    user_exit_mask                        = UEX_STOP ,
    verify_scratch_vsn                    = NO
```

FILES

    /etc/config/tmf.config   TMF configuration file

    /usr/include/tmf/tmfdefaults.h   Default TMF values

    /usr/include/tmf/tmfreq.h   TMF interface definition file

SEE ALSO

    msgdaemon(8), tmconf(8), tmconfig(8), tmmls(8)

    IRIX TMF Administrator's Guide

    IRIX TMF Release and Installation Guide

Index