# TMF Administrator’s Guide

007–3966–007

Version 1.4

# New Features in This Guide

This version contains the following:

- Clarifications about the TMF `tmf.config` configuration file. See "TMF Configuration File" on page 1.
- Removal of outdated information. Licensing information has been moved to *TMF Release and Installation Guide*
- Minor editorial changes.

# Record of Revision

| Version | Description |
|---|---|
| 1.0 | December 1998<br>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<br>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<br>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<br>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<br>Supports TMF release 1.3.5. |
| 006 | June 2003<br>Supports TMF release 1.4. |
| 007 | February 2007<br>Supports TMF release 1.4.7. |

# Contents

# Figures

# Tables

# Examples

# Procedures

# About This Guide

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

## 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 documentation, several online man pages describe aspects of TMF. Each man page includes a general description of one or more commands, system calls, or other topics, and provides usage details (command syntax, parameters, and so on). Man pages exist for the user commands, devices (special files), file formats, miscellaneous topics, and administration commands. Man page section identifiers appear in parentheses after man page names, as follows:

| | |
|---|---|
| User commands | (1) |
| Devices (special files) | (4) |
| File formats | (5) |
| Miscellaneous topics | (7) |

| | |
|---|---|
| Administration commands | (8) |

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

```
% man tmstat
```

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

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

## Obtaining Publications

You can obtain SGI documentation as follows:

- See the SGI Technical Publications Library at http://docs.sgi.com. Various formats are available. This library contains the most recent and most comprehensive set of online books, release notes, man pages, and other information.
- If it is installed on your SGI system, you can use InfoSearch, an online tool that provides a more limited set of online books, release notes, and man pages. With an IRIX system, enter `infosearch` at a command line or select **Help > InfoSearch** from the Toolchest.
- On IRIX systems, you can view release notes by entering either `grelnotes` or `relnotes` at a command line.
- On Linux systems, you can view release notes on your system by accessing the `README.txt` file for the product. This is usually located in the `/usr/share/doc/`*productname* directory, although file locations may vary.
- You can view man pages by typing `man` *title* at a command line.

## Conventions

The following conventions are used throughout this document:

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

## Reader Comments

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

You can contact SGI in any of the following ways:

- Send e-mail to the following address:

  techpubs@sgi.com

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

- Send mail to the following address:

  SGI
  Technical Publications
  1140 East Arques Avenue
  Sunnyvale, CA 94085–4602

SGI values your comments and will respond to them promptly.

# Configuration

This chapter describes the following:

- "TMF Configuration File" on page 1
- "Statement Order and Syntax" on page 10
- "Job Limits and the ULDB" on page 12
- "TMF and Comprehensive System Accounting (CSA)" on page 14

## TMF Configuration File

Before TMF is started, the TMF configuration file, `tmf.config`, must be updated to configure TMF. The `tmf.config` file resides `/etc/tmf/tmf.config`. TMF uses the values in these files to decide what to do in various situations. Example 1-1 on page 1 provides a sample file. You should set the parameters in your TMF configuration file to values that suit your system. You can update the file with any text editor. This chapter contains the basic information that you need for this task. For a description of the parameters, see the `tmf.config`(5) man page.

**Example 1-1** TMF Configuration File

The following sample TMF configuration file begins with a comment (the title of the file) preceded by the number sign character (#). Comments always begin with the # character. The rest of the file contains a number of statements. Each of these statements is described in succeeding sections. "Statement Order and Syntax" on page 10 describes syntax rules for these statements.

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

LOADER
      name = operator ,
      type = OPERATOR ,
      status = UP ,
      mode = ATTENDED ,
      message_path_to_loader = MSGDAEMON ,
      server = localhost ,
      scratch_volume_label_type = (AL,NL,SL) ,
      queue_time = 0 ,
      verify_non_label_vsn = YES ,
      message_route_masks = (MSGD) ,
      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 = (MSGD) ,
      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 = (MSGD) ,
      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 = (MSGD) ,
      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 = (MSGD) ,
      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_log                              = YES ,
loader_device_assignment_order          = ROUND_ROBIN ,
max_number_of_tape_users                = 100 ,
number_of_autoloader_retries            = 10 ,
operator_message_destination            = (MSGD) ,
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/TMF ,
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
```

## LOADER Statement

The TMF configuration file in Example 1-1 on page 1, contains five LOADER statements; these represent the five loaders that are available on this system. Each LOADER statement is composed of the parameters needed to describe a specified loader. For example, the first LOADER statement has 12 parameters:

```
LOADER
      name = operator ,
      type = OPERATOR ,
      status = UP ,
      mode = ATTENDED ,
      message_path_to_loader = MSGDAEMON ,
      server = localhost ,
      server_reply_wait_time = 300,
      scratch_volume_label_type = (AL,NL,SL) ,
      queue_time = 0 ,
      verify_non_label_vsn = YES ,
      message_route_masks = (MSGD) ,
      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 `localhost` and the amount of time that the `LOADER` process waits for a server response before declaring a timeout condition is 300 seconds.

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 volume serial numbers (VSNs) must be verified. `message_route_masks` is `MSGD`, 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.

## DEVICE_GROUP Statement

The file in Example 1-1 on page 1 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.

## AUTOCONFIG Statement

The AUTOCONFIG statement in Example 1-1 on page 1, 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
```

**Note:** The file fields in the first two DEVICE statements are for an IRIX system. For an SGI ProPack system, file indicates device files in the /dev/ts directory. For the actual location of the device files on your system, see the TS(7) man page.

## 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 1-1 on page 1, 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 "DEVICE_GROUP Statement" on page 7).

The pathname 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).

**Note:** The file field in this sample DEVICE statement is for an IRIX system. For an SGI ProPack system, file indicates device files in the /dev/ts directory. For the actual location of the device files on your system, see the TS(7) man page.

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

## 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 1-1 on page 1, 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 because `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.

## Statement Order and Syntax

A statement consists of a name followed by a list of parameters. This section describes the following:

- "Statement Order" on page 10
- "Statement Syntax" on page 11

### Statement Order

There are at least four statements in a TMF configuration file, 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) define devices that TMF will control and that are automatically configured during the system boot.
4. `OPTIONS` statement (one per system).

## Statement Syntax

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.
```

## Job Limits and the ULDB

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

**Procedure 1-1** 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 |
   | `datamanager` | Allows or prevents the use of absolute positioning requests |

   You set these permissions to 0 (disable) or 1 (enable):

   ```
   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 1-2** 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
```

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
        datamanager = 0
        STK9840 = 2
        DLT = 2
}
```

**Example 1-3** Specific Device Group Definitions

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

```
user adm {

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

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.

## TMF and Comprehensive System Accounting (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, see the `CSA`(1M) man page.

*Chapter 2*

# Administration

This chapter describes the following TMF administration topics:

## Tape Access

Users can access tapes attached to an SGI system by one of two interfaces:

- TMF, which is described in this manual.
- The character-special tape interface (TS). For more information, see the `ts`(7M) man page.

## Tape Libraries

This section describes how TMF interacts with the tape library software subsystem and also covers some high-level configuration information:

## Communication

TMF always communicates with the tape loader via an intermediate software system that is provided by the library vendor. TMF supports the following:

- StorageTek libraries using the ACSLS software interface running on a SUN host. 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). Check with your StorageTek representative to validate the values of `CSI_UDP_RPCSERVICE` and `CSI_TCP_RPCSERVICE`.
- IBM libraries using the controlled path service (CPS) software interface running on an IBM RISC System/6000 platform. TMF communicates with the IBM CPS software via a child process called `ibmnet`, which TMF starts after the library is configured up.
- EMASS libraries using the VolServ software interface running on a SUN host. TMF communicates with the vendor-supplied software interface, `VolServ`, via a child process called `esinet` which TMF starts after the library is configured up.

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.

Figure 2-1 shows the software and hardware configuration between the local host and the StorageTek, IBM, and EMASS libraries.

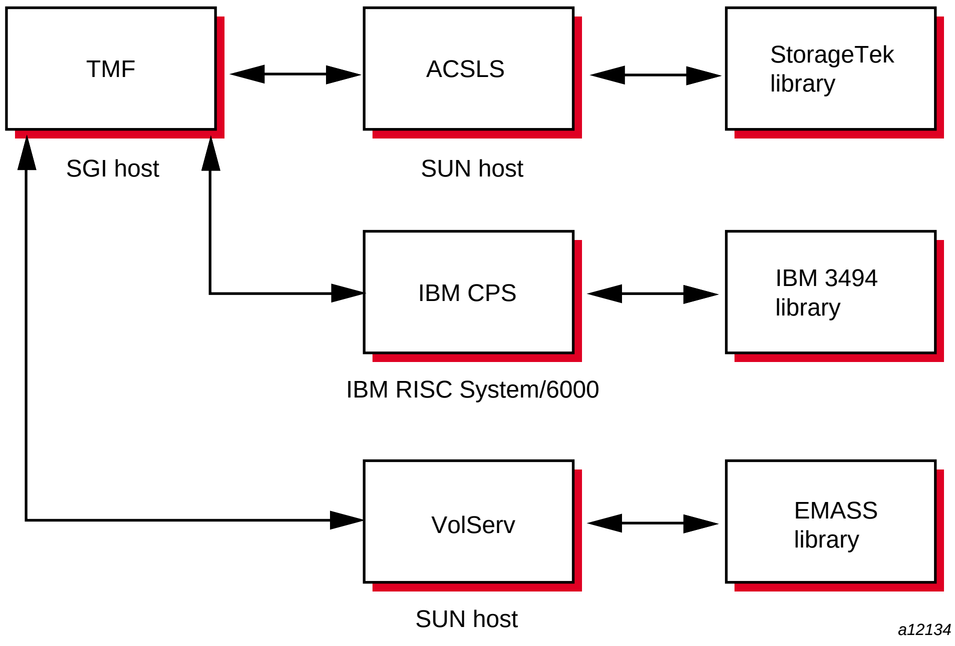

**Figure 2-1** Library Communication

## 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.

### 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*.

## OpenVault as a TMF Loader

You can use the OpenVault storage library management facility as a TMF loader. You can configure it on your local host or on a remote host. Figure 2-2 shows OpenVault on the same host as TMF and Figure 2-2 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 in "Checklists for Using OpenVault with TMF" on page 20 provide information on using OpenVault with TMF. For detailed information on using OpenVault, see the *OpenVault Operator's and Administrator's Guide*.

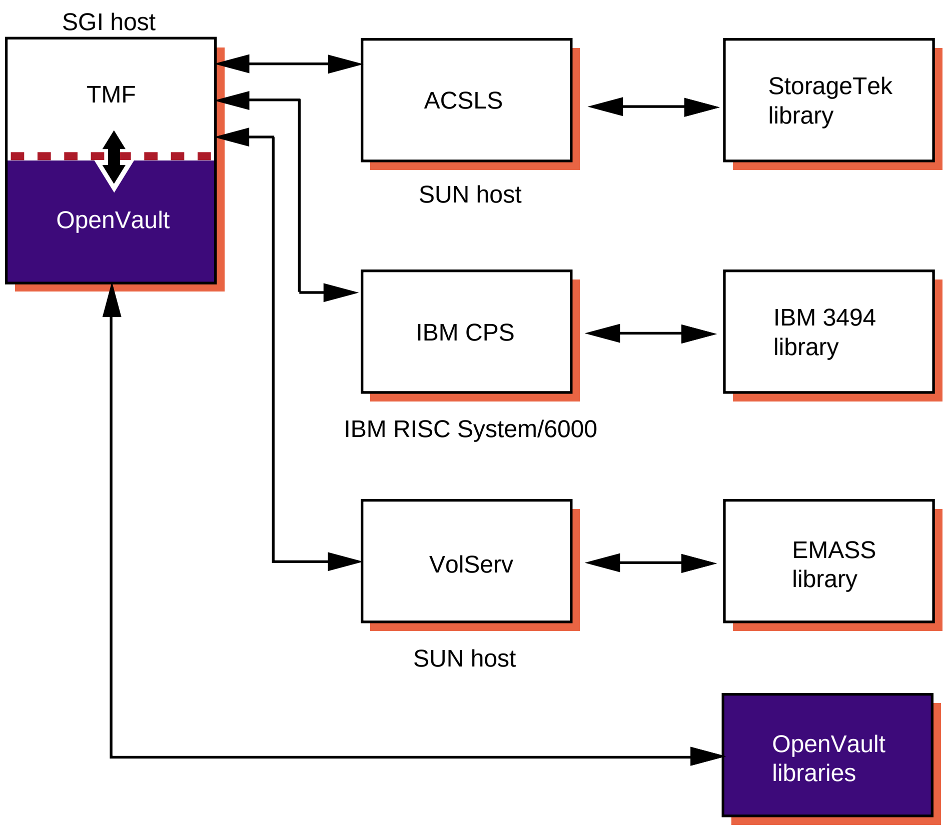

**Figure 2-2** OpenVault on the Local Host

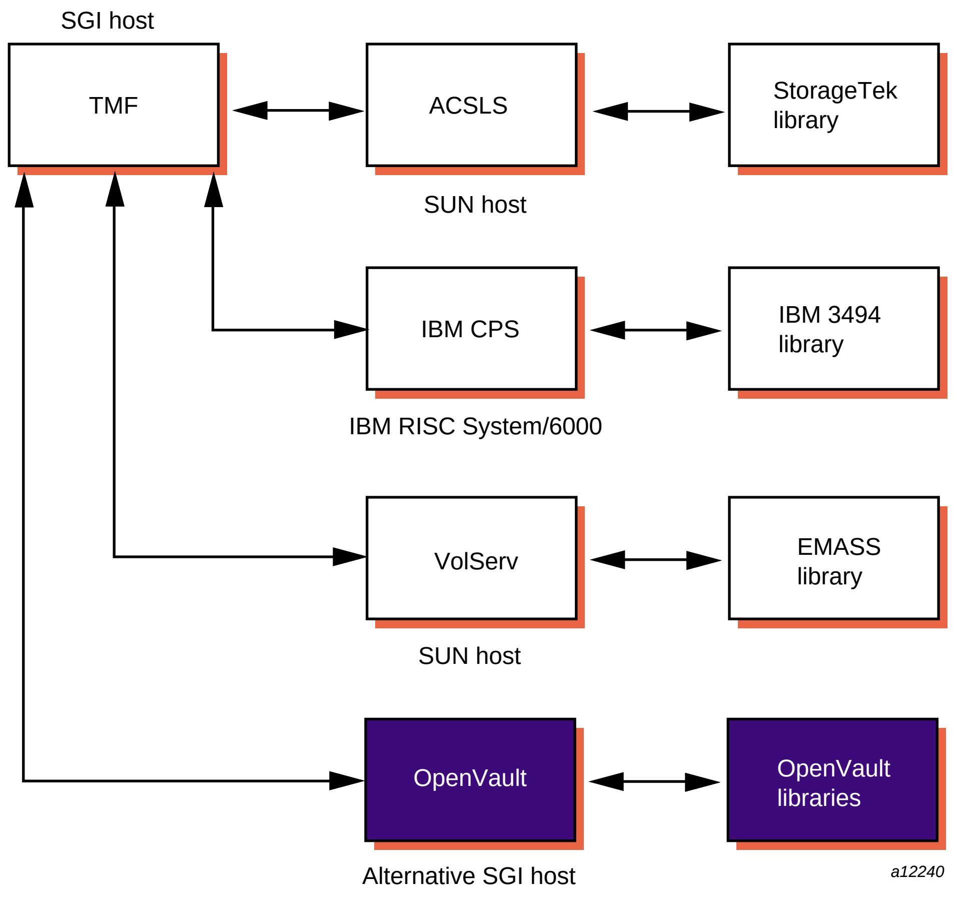

**Figure 2-3** Figure 2-2OpenVault on a Remote Host

## Checklists for Using OpenVault with TMF

The following terms are used in this section:

- *ovserver_host_name*: The name of the host on which the OpenVault server is running. When configuring TMF and OpenVault, use the results of executing the `hostname` command on the OpenVault server host to obtain this value.
- *tmf_host_name*: The name of the host on which TMF will run. Use the results of executing the `hostname` command on the TMF host to obtain this value.

- *tmf_node_name*: The node name of the host on which TMF will run. This may or may not be the same as the TMF hostname. Use the results of executing the `uname -n` command on the TMF host to obtain this value.
- *tmf_application_name*: The OpenVault application name to be used by TMF. If you do not specify a value in the TMF configuration file, TMF uses the default name `tmf`.
- *tmf_instance_name*: The OpenVault instance name to be used by this instance of TMF. If you do not specify a value in the TMF configuration file, TMF uses the following default name:

  *tmf_node_name*`.`*tmf_application_name*

- *tmf_key*: The TMF security key to be used for TMF. This alphanumeric string is used as a password by TMF to secure the connection with the OpenVault server. If you want to enable security, you must create and configure a *tmf_keyfile* containing the *tmf_key* that you want to use. If you do not specify a *tmf_keyfile* in the TMF configuration file, TMF uses `none` as the *tmf_key*, which means that security checking between TMF and the OpenVault server is disabled.
- *tmf_keyfile*: The OpenVault keyfile to be used by TMF when communicating with the OpenVault server. The *tmf_keyfile* contains the private security key *tmf_key* that TMF uses to establish an authorized connection with the OpenVault server. This file is required only if you decide that you want to enable security checking between TMF and the OpenVault server.

  When the key file is created, use the `chown` command to set the user ownership to `root`, use the `chgrp` command to set its group ownership to `sys`, and use the `chmod` command to set its files permissions to `0500`. The key file should contain a single line consisting of 5 blank or tab-separated fields in the following format:

  *ovserver_host_name*    *tmf_application_name*    *tmf_instance_name* `CAPI` *tmf_key*

  The *ovserver_host_name*, *tmf_application_name*, *tmf_instance_name* values in the *tmf_keyfile* must match the values specified in the TMF configuration file or the TMF default values if you do not specify them in the TMF configuration file. Mismatches between the TMF configuration file and the *tmf_keyfile* prevent TMF from contacting the OpenVault server.

Procedure 2-1 and Procedure 2-2 on page 26, list the steps you need to take before you use TMF with OpenVault.

**Procedure 2-1** OpenVault Checklist

Configure the drives and libraries in OpenVault so that TMF can use them. The following steps ensure that the TMF and OpenVault counterparts match.

1. Using the OpenVault configuration command, define a list of OpenVault drives and libraries. If you are using a version of OpenVault prior to 1.5, the configuration command is `/usr/OpenVault/setup`. For OpenVault 1.5 and later, the configuration command is `/usr/sbin/ov_admin`.

---

**Note:** TMF devices are defined subsequently. TMF device names **must** match the drive names used by OpenVault. TMF supports device names only up to 8 characters long. With that in mind, be sure to define OpenVault drive names consisting of 1 to 8 characters.

---

   TMF devices are grouped as follows:

   - In TMF, every drive belongs to a device group
   - In OpenVault, every drive belongs to a drive group
   - For every TMF device group to be managed by OpenVault, there must be a matching OpenVault drive group

2. Create an OpenVault application name so that a group of tapes, defined as a cartridge group, can later be used by the `tmf` application. For example, where `tmf` is the application name:

   ```
   # ov_app -c tmf
   Created Application: tmf
   ```

   In OpenVault, only one application can be assigned to a cartridge. OpenVault mounts a cartridge only if the request comes from the assigned application. Cartridges have the following characteristics:

   - 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

3. Create any new drive groups and/or cartridge groups, according to the following steps. The default group names are `drives` and `carts`, respectively.

   a. Use the following OpenVault commands to list the drive groups and cartridge groups:

   ```
   # ov_drivegroup
   group               unload time
   drives              60

   # ov_cartgroup
   group               group prio
   carts               1000
   ```

   b. Create a new drive group appropriate for your drive type(s). Keep in mind that this is the same name you will give to the device group you define in TMF. The following example uses `DLT8000` as a drive group name.

   ```
   # ov_drivegroup -c DLT8000
   created drive group: DLT8000
   ```

   To simplify things in the preceding example, the default cartridge group, `carts` is used. However, if you choose to create a new cartridge group, you can do so by using the following command:

   ```
   # ov_cartgroup -c new_cartgroup_name
   ```

   c. Verify that the new drive group (and any new cartridge groups) have been created, as follows:

   ```
   # ov_drivegroup
   group               unload time
   DLT8000             60
   drives              60

   # ov_cartgroup
   group               group prio
   carts               1000
   ```

4. Allow the OpenVault application name, `tmf`, to use the cartridge groups and drive groups that will be used by TMF (defined in step 3).

In OpenVault, more than one application can be assigned to a drive group; OpenVault uses a drive within a drive group only if the request comes from an assigned application. OpenVault drives have the following characteristics:

- 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)

Use the following steps for all drive and cartridge groups:

a. List applications (for all drive groups and cartridge groups).

   Use the following commands to list the application information for all drive groups and cartridge groups. The default OpenVault application name is `ov_umsh`. (Notice that the drive groups (and cartridge group) that were created in step 3 do not appear here. This is because they have not yet been given an application to use.)

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

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

b. Add TMF applications. Use the following commands with the `-a` option to add the default TMF application name, `tmf`, to any drive groups and cartridge groups that will be used by TMF.

```
# ov_drivegroup -a -G DLT8000 -A tmf
Drive-group-application creation:
       Application: tmf
       Group: DLT8000

# ov_cartgroup -a -G carts -A tmf
Cartridge-group-application creation:
       Application: tmf
       Group: carts
```

c. Recheck: List applications (for all drive groups and cartridge groups), as follows:

```
# ov_drivegroup -s -A ’.*’
application          group                group app prio  unload time
ov_umsh              drives               1000            60
tmf                  DLT8000              1000            60

# ov_cartgroup -s -A ’.*’
application          group                group app prio
ov_umsh              carts                1000
tmf                  carts                1000
```

5. Create a group of physical cartridges, assigned to the default cartridge group, `carts`, and to be used by the `tmf` application. The fields in the following example are as follows:

   Cartridge PCL name
   OpenVault cartridge type
   Volume name (should match the PCL name)
   Application name

   ```
   # ov_import -g carts
   DLT014 DLTIV DLT014 tmf
   DLT015 DLTIV DLT015 tmf
   <ctrl-d>
   #
   ```

   You can use the following command to list all of the PCLs/volumes used by the `tmf` application:

   ```
   # ov_lscarts -A tmf
   DLT014         DLT015
   ```

   Or, for a more extensive look:

```
# ov_lscarts -l -A tmf
PCL        cart type  owner           state      part     volume
DLT014     DLTIV      tmf             ok         PART 1   DLT014
DLT015     DLTIV      tmf             ok         PART 1   DLT015
```

6. Make sure that the following line is in the `core_keys` file. If you are using an OpenVault version prior to 1.5, the `core_keys` file is `/usr/OpenVault/var/core_keys`. For OpenVault version 1.5 and later, the `core_keys` file is `/var/opt/openvault/server/config/core_keys`.

   *tmf_host_name tmf_application_name tmf_instance_name* `CAPI` *tmf_key*

   For example, if `armadillo` is the host that TMF is running on (*tmf_host_name*), the TMF application name (*tmf_application_name*) is `tmf`, the TMF instance name (*tmf_instance_name*) is `armadillo.tmf`, the language is `CAPI`, and the security key (*tmf_key*) is not used (`none`), you would enter the following line in the `core_keys` file:

   ```
   armadillo  tmf   armadillo.tmf    CAPI  none
   ```

   In the OpenVault documentation, the terminology may differ: the *tmf_keyfile* file is the key authorization file, and the TMF application name (*tmf_application_name*) is the client.

7. To collect debugging information in the `OVLOG` file, enter the following command:

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

   If you are using an OpenVault version prior to 1.5, the `OVLOG` file is `/usr/OpenVault/var/OVLOG`.*YYYYMMDD*, where *YYYYMMDD* represents the year (*YYYY*), month (*MM*), and day (*DD*) that the file was created. For OpenVault version 1.5 and later, the `OVLOG` file is `/var/opt/openvault/server/logs/OVLOG`.*YYYYMMDD*.

   When you have finished debugging the `OVLOG` file, you can use the following command to reset the messages back to the `information` level:

   ```
   ov_msg -s -t core -m information
   ```

**Procedure 2-2** TMF Checklist

Modify the `tmf.config` file to support the OpenVault loader, device groups, and devices. For information on the `tmf.config` file, including a complete example file, see the `tmf.config`(5) man page.

1. Define an OpenVault `LOADER` statement, as follows:

   a. Define the `type` to be `OPENVAULT`:

      ```
      type = OPENVAULT
      ```

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

      `server =` *ovserver_host_name*

   c. 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*

      If this line is omitted from the `LOADER` statement, the default application name will be `tmf`.

   d. Either use the TMF default instance name or specify a different instance name for TMF with the following parameter:

      `ov_tmf_instance_name =` *tmf_instance_name*

      If you omit this line from the `LOADER` statement, a default instance name of *tmf_node_name.tmf_application_name* will be used, where the value of *tmf_application_name* is taken from step c, previously.

   e. If the OpenVault communication security feature is to be enabled, specify the pathname of the key file which will contain the TMF security key (*tmf_key*):

      `ov_tmf_keyfile =` *tmf_keyfile*

      If this line is omitted from the `LOADER` statement, the OpenVault communication security feature will not be used (*tmf_key* equals `none`). For more information on the key file and security keys, see the *OpenVault Operator’s and Administrator’s Guide*.

2. Make sure the `DEVICE_GROUP` names match the OpenVault drive group names.

3. Make sure the `name` field in each of the `DEVICE` statements match the corresponding OpenVault drive names.

## 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.

## 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:

- "Starting and Stopping the Message Daemon" on page 28
- "Messages" on page 29
- "Message Daemon Commands" on page 29

### 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/sbin/msgdaemon
```

To stop the message daemon, enter the following command:

```
/usr/sbin/msgdstop
```

## 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 `newmsglog` shell script saves the last several versions of the log. The `newmsglog` script resides `/usr/sbin`. 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.

## Message Daemon Commands

The message daemon request pipe is located in the `/var/spool/msg` directory. Table 2-1 shows the message daemon commands and the permissions required to access them.

**Table 2-1** Message Daemon Commands

| Command | Permission | Description |
|---|---|---|
| `msgdaemon`(8) | Administrator | Starts the message daemon |
| `msgdstop`(8) | Administrator | Stops the message daemon |
| `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 is used. The default configuration file resides in `/etc/tmf/oper`.

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

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

**Table 2-2** 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.

## 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 tmf on
```

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

```
chkconfig tmf off
```

## 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/sbin/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/sbin/tmstop
```

The `tmstop`(8) command has no options.

Table 2-3 shows these commands and the permissions required to access them.

**Table 2-3** TMF Commands

| Command | Permission | Description |
|---|---|---|
| `tmdaemon`(8) | Administrator | Starts TMF |
| `tmstop`(8) | Administrator | Stops TMF |

## 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

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:

### 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 and 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.

### TMF Daemon Problems

If the TMF daemon 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
```

For more information about the TMF daemon, see the `tmdaemon`(8) man page.

## Pertinent TMF Files

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

The following TMF user commands reside in the `/usr/bin` directory:

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

The following TMF administrator commands reside in the `/usr/sbin` directory:

`tmclr`(8)
`tmcollect`(8)
`tmconfig`(8)
`tmdaemon`(8)
`tmfrls`(8)
`tmgstat`(8)
`tmlabel`(8)
`tmmls`(8)
`tmmql`(8)
`tmset`(8)
`tmstop`(8)
`tmunld`(8)

The following TMF processes reside in the `/usr/lib/tmf` directory:

```
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 3-1 on page 35, describes this subset, and Example 3-1 on page 36, shows how you use the `tmstat`(1) command to identify a `tmf`*xxx* file.

**Table 3-1** 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.<br>The `tmset`(8) command must be issued with the `-T` option set to `off` in order to disable tracing and may impact problem diagnosis as minimal tracing may not provide enough information to resolve problem situations.<br>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. |
| `ldname.log` | This file is a linear log file of all the requests and responses to the named loader. |

| File | Description |
|---|---|
| *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 3-1** `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  NQSid
                                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.

## Using Tracing

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

- "`tmcollect` Utility" on page 37
- "Enabling and Disabling Tracing" on page 37
- "Sample Trace Analysis" on page 38

### `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.

### Enabling and Disabling 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.

## 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 3-2 shows the information you can obtain from a trace line.

**Example 3-2** 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. |

## Resolving Common Problems

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

## **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.

## **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.

## **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.

# Index