TMF Release and Installation Guide

007–3967–008

Version 1.4.7

New Features in This Guide

This revision includes the following:

- Support for SGI License Key (LK) licensing software for the SGI ProPack 5 platform and clarifications to the information about FLEXlm licensing for IRIX and SGI ProPack 4. See Chapter 5, "License Requirements" on page 25.
- Removal of outdated information.
- Minor editorial changes.

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.
006	June 2003 Supports TMF release 1.4.
007	September 2004 Supports TMF release 1.4.2.

008	February 2007 Supports TMF release 1.4.7.

Contents

Tables

About This Guide

This guide documents the release and installation of the Tape Management Facility (TMF) release 1.4.7. This release is supported by the hardware and software described in "Software Requirements" on page 8.

The following release and installation topics are provided:

Related Publications

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

- *TMF Administrator's Guide*
- *TMF 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.
GUI	This font denotes the names of graphical user interface (GUI) elements such as windows, screens, dialog boxes, menus, toolbars, icons, buttons, boxes, fields, and lists.

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.

Software Overview

TMF provides data management solutions for users who require fast, reliable access to massive amounts of data. This chapter provides an overview of the following TMF features:

For a description of TMF capabilities and the role of the system administrator, refer to the *TMF Administrator's Guide*.

Architecture

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

The basic elements of TMF are the TMF daemon and TMF tape device driver. TMF provides operating personnel with a means to view and manage the tape resources

configured within TMF. It also is the backbone for the operation of the Data Migration Facility (DMF).

TMF is started by the system operator or the system administrator, or it is started automatically as part of the system startup. TMF has superuser privileges. Therefore, it can communicate directly with the TMF device driver and the SCSI tape device driver to process your requests. Most vendors only support a character-special functionality, defined simply as the ability to open, to read from and write to, and to close a device that is recognized by the system software.

SGI offers a well-defined set of advanced functionality on all devices that TMF supports:

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

Tape Label Support

TMF supports the following:

- ANSI standard labels
- IBM standard labels
- Single filemark format tapes
- Nonlabeled tapes

Single filemark format tapes do not have labels and are terminated by a single filemark at the end-of-volume, whereas a normal nonlabeled tape is terminated by two filemarks at the end-of-volume. Also, bypasslabel processing is available to users with `root` permission. Bypasslabel processing lets these users read or write tape labels as regular files.

Resource Management

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

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

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

Volume Mounting and Unmounting

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

TMF supports several different families of libraries; see "Device Support" on page 5. The automatic volume recognition (AVR) feature allows the operator to premount tapes prior to use and to direct the mounting of tapes to specific devices.

Tape Positioning

Tape positioning lets you position a tape to the beginning of a tape block. Tape movement may be forward or backward; however, tape positioning directives cannot be used to circumvent normal tape label processing or label checking unless you have `root` permission and use an absolute track address positioning request. You can position the tape file relative to a filemark, tape block, or volume; or you can position the tape file to an absolute track address.

Front-End Servicing

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

User End-of-Volume Processing

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

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

Multifile Volume Allocation

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

Concatenated Tape Files

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

Tape Message Log File

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

Device Support

TMF provides support for the following tape libraries and tape devices:

- EMASS libraries using the VolServ software interface
- IBM libraries using the CPS software interface
- OpenVault-supported libraries when using OpenVault as a loader
- StorageTek libraries using the ACSLS software interface
- Any tape drive supported by the SGI TS tape driver

Release Package

This chapter discusses the following:

Release Package Contents

The TMF release package includes the following:

- A CD-ROM that contains the installable binary packages for the TMF release
- *TMF Administrator's Guide*
- *TMF Release and Installation Guide* (this publication)
- *TMF User's Guide*
- A TMF entitlement ID for licensing (see Chapter 5, "License Requirements" on page 25)

Release and Packaging for FFIO

To use the flexible file I/O (FFIO) library interface to TMF, you must install the MIPS 7.3 product, which includes compilers, libraries, and tools.

Software Requirements

The TMF release is supported on the following:

- IRIX 6.5.28 or later
- SGI ProPack 4 SP3 and SGI ProPack 5
- OpenVault 1.5.8 or later (if OpenVault will be used as a loader)

Licensing Information

TMF is released independently of operating system releases and is distributed by order only to licensed sites. Software keys are used to enforce licensing. Each TMF license applies to a specific system. TMF license fees vary depending on the type of hardware. See Chapter 5, "License Requirements" on page 25.

Installing TMF

This chapter discusses the following:

Note: After you have completed the TMF installation, you must configure and restart TMF prior to using it. For information, see the *TMF Administrator's Guide*.

Preparing for Installation

Before you install TMF software, do the following:

- Verify that you are `root`.
- Back up the current installation material if you are installing a TMF replacement or upgrade.
- *(IRIX only)* Ensure `mediad`, the movable media daemon, is not accessing the same devices as TMF. Using `mediad` with the same devices causes error messages to be generated in the `SYSLOG` 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. You should use the installation documentation for the libraries at your site to correctly install these products.
- Ensure that TMF is not executing before beginning the installation process.

Installing TMF on an IRIX System

1. Insert and mount the TMF CD.
2. Using the left mouse button, select the following:

   System
   > Software Manager

3. On the **Available Software** list, select **/CDROM/irix/dist**.
4. Click **Customize Installation**. You will receive more information about the size of the TMF software, including the directories and files. Click the folder icon to view the contents of the software package.
5. Click the **Start** button to install the package. Software Manager will issue the following message when the TMF installation is complete:

   ```
   Installations and removals were successful.
   ```

6. Select the following:

   File
   > Exit

Installing TMF on an SGI ProPack System

1. Insert and mount the TMF CD.
2. Change to the directory in which the `tmf-cmd` package resides. Read the `README` file in this directory for further information and instructions.

   - For SGI ProPack 4 systems:

     ```
     $ cd /CDROM/ProPack4/rpms
     ```

   - For SGI ProPack 5 systems:

     ```
     $ cd /CDROM/ProPack5/rpms
     ```

3. Install the `tmf-cmd` package, which contains the TMF user commands, administrator commands, and TMF processes:

   ```
   $ rpm -i tmf-cmd-1.4.7-0-ia64.rpm
   ```

You must install the TMF kernel module RPM for the currently running kernel. The TMF kernel module RPM name varies according to the SGI ProPack release (where *kernel_variant* is the kernel variant output by the `uname -r` command):

- SGI ProPack 4:

  `km-`*kernel_variant*`-sgi-tape`

- SGI ProPack 5:

  `sgi-tmf-kmp-`*kernel_variant*

For more information on RPMs, see the `rpm`(8) man page.

TMF Directory Structure

TMF components are installed into the locations shown in Table 3-1:

Table 3-1 TMF Directories and Files

File	Description
`/etc/init.d/tmf`	Shell script to start and stop TMF.
`/etc/tmf/tmf.config`	The `/etc/tmf` directory contains TMF configuration files, where `tmf.config` is the default. For information on configuring TMF, see the *TMF Administrator's Guide*.
`/usr/bin/`	TMF user commands.
`/usr/include/tmf/`	TMF include files, which users need to compile programs that access TMF services.
`/usr/lib/tmf/`	TMF libraries, modules and TMF processes.
`/usr/lib/tmf/uex.tar`	Tar file containing the user exit files, which, if required, you can use to rebuild TMF with user exits.
`/usr/sbin/`	TMF administrator commands.

Building TMF for User Exits

User exits allow customers to add special routines to communicate with TMF without having access to the TMF source code. User exits allow a system process to examine and modify a structure associated with a tape file.

You can install the TMF release package with or without user exits. If you do not require user exits, then no further building of TMF is needed. If you require user exits, follow the instructions in "User Exit Implementation" on page 13.

For descriptions of the individual user exits, see "User Exits Requiring Configuration" on page 15 and "User Exits Not Requiring Configuration" on page 24

User Exit Implementation

To implement user exits, it is necessary to modify and recompile the user exit files in the `uex` directory (`uexcmd.c`, `uexmsg.c`, `uextmf.c`, and `vsnexit.c`). To switch the individual or all user exits on or off, make an entry in the TMF configuration file.

Some user exits do not require configuration; those exits, which are defined in `uexcmd.c`, are used only by the TMF commands and do not require configuration (see "User Exits Not Requiring Configuration" on page 24).

The following is an example of the entry to add to the `OPTIONS` statement of the TMF configuration file:

```
user_exit_mask = (UEX_ASK_EXPDT,UEX_ASK_LBSW,UEX_ASK_RETRY),
```

The options for this entry are as follows:

`UEX_ALL`	Enables all user exits
`UEX_ASK_EXPDT`	Enables all `uex_askexpdt` user exits
`UEX_ASK_HDR1`	Enables all `uex_ask_hdr1` user exits
`UEX_ASK_LBSW`	Enables all `uex_asklbsw` user exits
`UEX_ASK_RETRY`	Enables all `uex_askretry` user exits
`UEX_ASK_VERSCR`	Enables all `uex_askverscr` user exits
`UEX_ASK_VSN`	Enables all `uex_askvsn` user exits

UEX_ASK_SCR_VSN	Enables all uex_scr_vsn user exits
UEX_CHK_ACCESS	Enable all uex_chk_access user exits
UEX_CLS_FILE	Enable all uex_cls_file user exits
UEX_MAC_HDR2	Enables all uex_mac_hdr2 user exits
UEX_MNT_MSG	Enables all uex_mnt_msg user exits
UEX_SM_DEX	Enables all uex_sm_dex user exits
UEX_SM_DUX	Enables all uex_sm_dux user exits
UEX_SM_VAX	Enables all uex_sm_vax user exits
UEX_SM_VUX	Enables all uex_sm_vux user exits
UEX_START	Enables the uex_start user exit
UEX_STOP	Enables the uex_stop user exit

If an invalid option is used, an error message appears in the daemon.stderr file similar to the following:

```
TM425 - Error in file /etc/tmf/tmf.config, line 122, offset 49, with value "UEX_STOPP" :
syntax error: expecting:
```

To modify and recompile the user exits, unpack the user exit code by entering the following instructions:

```
$ cd /usr/lib/tmf
$ tar -xvf uex.tar
```

These commands unpack the user exit tar file and create the uex directory. Examples on how to code user exits can be found in the distributed user exit files in the uex directory. Once the user exits have been modified, they can be recompiled by using the following commands:

```
$ cd /usr/lib/tmf
$ make TOPDIR=‘pwd‘ uex
```

To complete the integration of the modified user exits into TMF, enter the following command:

```
$ make TOPDIR=‘pwd‘ install
```

The TMF installation is now complete.

User Exits Requiring Configuration

User exits returning the values of `0` or `-1` can use the defined symbolic values of `YES` (`0`) or `NO` (`-1`) defined in the `tmuex.h` file. Descriptions of TMF user exits that require configuration follow:

`uex_askexpdt(uex_table)`

Receives the `uex_table` structure and returns an integer value of `YES` or `NO`. This exit returns an answer to the question "Can user *userid* write on unexpired VSN *vsn*?"

`uex_asklbsw(uex_table)`

Receives the `uex_table` structure and returns an integer value of `YES` or `NO`. This exit returns an answer to the question "Can user *userid* switch from the *original label* label to *new label* for VSN *vsn*?"

```
uex_askretry(uex_table, message_type, server_or_front-end,
message_id, reason_for_retrying)
```

Receives the `uex_table` structure and the additional parameters as shown above and returns an integer value of `YES` or `NO`. It is called when the TMF daemon is unable to send a request to a front end or server and returns an answer to the question "Should message to the front end be re-sent or aborted?" The returned value of `YES` means to retry the request; `NO` means to cancel the request.

`uex_askverscr(uex_table, vsn)`

Receives the `uex_table` structure and a VSN. It returns an integer value of `YES` or `NO`. This exit returns an answer to the question "Is the *vsn* volume on the *dvn* device a valid scratch volume for the session identifier ?"

`uex_askvsn(uex_table)`

Receives the `uex_table` structure and returns either a character pointer with the value `NULL` or an address of a string. This exit returns an answer to the question "What is the VSN on the *dvn* device ?" The returned value of `NULL` means no VSN was returned while a pointer to a string is used as the value of the scratch VSN. If no VSN is returned, then the TMF daemon will call the `askvsn`() routine, just as if the user exit had not been taken.

`uex_ask_hdr1(uex_table)`

Receives the `uex_table` structure and returns an integer value of `YES` or `NO`. This user exit is called from a child process in the TMF daemon at the point where the volume header (VOL1) and first file header (HDR1) labels have been read from a tape and the child process prepares to check some of the values in the HDR1 label against values that are kept by the TMF daemon and its child processes.

This user exit provides a site an opportunity to add code that enables the TMF daemon to do the following:

- Obtain a number that controls how many characters of the file identifier field in a HDR1 label are compared to a character string kept by the TMF daemon or to an alternate character string that is provided by this user exit.

  The TMF daemon uses the number in the `user_fidl` field of the `uex_table` structure. If the site changes this number, the modified number must have a value that is equal to or greater than 1 and less than or equal to 16. The number must be returned in the `user_fidl` field, while the return value from this user exit must be `YES`. If `NO` is returned, the `user_fidl` field is not examined.

- Obtain an alternate character string for the file identifier to be compared to the character string in the file identifier field in the HDR1 label from the tape.

  The character string that the TMF daemon uses is in the `user_fid` field of the `uex_table` structure. If the site changes this string, the modified character string must be stored in the `user_fid` field, while the return value from this user exit must be `YES`. If `NO` is returned, the `user_fid` field is not examined.

- Obtain an alternate one-character string that, in the case of ANSI labels, is compared to the accessibility character string in the accessibility field in the HDR1 label from the tape. If the character strings match, the action that is taken is the same as the action taken for the space character as defined in the ANSI standard.

  The character string that the TMF daemon uses is in the `user_vac` field of the `uex_table` structure. If the site changes

this string, the modified character string must be stored in the `user_vac` field, while the return value from this user exit must be `YES`. If `NO` is returned, the `user_vac` field is not examined.

If this user exit returns `YES`, the following actions occur:

- The `user_fidl` field is checked for a number that is equal to or greater than `1` and less than or equal to `16`. If the returned number is outside this range, the default value of `17` is used.
- The contents of the `user_fid` field is copied into a TMF daemon structure.
- The contents of the `user_vac` field is copied into a TMF daemon structure after it is checked against the following characters:

  ```
  A ... Z, 0 ... 9, " !\"%&'()*+,-./:;<=>?_"
  ```

If `NO` is returned, `uex_table` information is not used to update data structures in the TMF daemon.

`uex_chk_access(uex_table)`

Receives the `uex_table` structure and returns an integer value of `YES` or `NO`.

This user exit is called from a child process in the TMF daemon at the point where the TMF daemon has accepted a tape volume to read from or to write to. It provides a site an opportunity to add code. For example, the code could allow or reject access to a tape volume after it has checked a locally maintained permission file.

When this user exit has been entered to check permission to access an output tape, the TMF daemon upon return from this user exit checks the `user_error` field of the `uex_table` structure. If this field contains error code `ETNSC` (`90089 - not scratch`), the TMF daemon rejects the tape volume. If more tape volumes have been specified in the `tmmnt`(1) command, `tmmnt`(1) tries the next tape volume.

Besides setting the `user_error` field to `ETNSC`, this user exit also sets the `uex_table` bit field `uex_lst.flg.nsc` to 1. The TMF daemon updates the `fit` field `lst.flg.nsc` with this information from the `uex_table` field.

If the `user_error` field contains any other error number, the TMF daemon upon return aborts the child process with error code `EACCES` (`13 - permission denied`). If this user exit returns the value `NO`, the TMF daemon aborts the child process with error code `EACCES`. If this user exit returns the value `YES`, the TMF daemon accepts the tape volume and processing continues.

When this user exit has been entered to check permission to access an input tape, the TMF daemon upon return from the user exit checks the return code. If the return code is `NO`, the tape volume is rejected and the child process aborts with error code `EACCES`. If the return code is `YES`, the tape volume is accepted and processing continues.

Besides the possible update of the `lst.flg.nsc` bit field in `fit`, no other information from `uex_table` is used to update data structures in the TMF daemon.

`uex_cls_file(uex_table)`

Receives the `uex_table` structure and returns. The TMF daemon upon return from this user exit does not update any of its information with information from `uex_table`.

This user exit is called from a child process in the TMF daemon after the tape processing is completed and when the tape file is about to be closed. The exit provides a site an opportunity to add code. For example, the code could enable the TMF daemon to add information to the `tape.msg` file concerning the tape volumes that were used while processing the tape file.

`uex_mac_hdr2(uex_table)`

Receives the `uex_table` structure and returns an integer value of `YES` or `NO`.

This user exit is called from a child process in the TMF daemon after the TMF daemon reads the label information from the tape and has called the security code to check proper beginning-of-tape structure: VOL1, HDR1, and HDR2 labels. The user exit is called when a tape has a VOL1 and a HDR1 label, but not a HDR2 label. It provides a site an opportunity to add code, which could allow or reject access to the tape volume.

If this user exit returns the value NO, the TMF daemon continues its normal processing. It allows the tape to be overwritten, but not to be read. If the exit returns the value YES, the TMF daemon allows the user access to the tape. A return code of NO complies with security guidelines.

No information from the uex_table structure is used to update data structures in the TMF daemon.

uex_mnt_msg(uex_table)

Receives the uex_table structure and returns.

This user exit is called from either a child process or the TMF daemon itself after it has built a tape mount message and before it is sent to be processed. The exit provides a site an opportunity to add code, which could add information to the existing mount message or change it in any other way.

When the TMF daemon transfers control to this user exit, the user_buff field of the uex_table structure contains the address of the mount message character string and the user_bytes field of the uex_table structure contains the length in bytes of the memory block that are allocated to hold the mount message.

If this user exit extends the length of the delivered character string beyond the size of the allocated memory block, the user exit allocates the necessary memory to store the newly composed mount message. The address is returned to the TMF daemon in the user_buff location. The length in bytes of the newly allocated memory block is returned in the user_bytes field.

The update field in uex_table has to be set to a nonzero value.

If this user exit does not extend the length of the delivered character string beyond the size of the allocated memory block, the user exit does not have to allocate another memory block and the address in the user_buff field is left unaltered. The length in bytes of the allocated memory block in the user_bytes field also remains unaltered. The update field of the uex_table structure has to be set to zero.

When this user exit returns, the TMF daemon checks the value of the update field. If its value is zero, the TMF daemon continues its

normal processing. It sends the mount message from the location it has allocated off to be processed.

If the value in the `update` field is nonzero, the TMF daemon compares the address of the memory block it has allocated for its mount message and the address that has been returned in the `user_buff` field. If these addresses are the same, the TMF daemon continues its normal processing. If these addresses differ, the TMF daemon frees the memory block it had allocated for its mount message and takes the address from the `user_buff` field as its replacement.

No other `uex_table` information is used to update data structures in the TMF daemon.

`uex_scr_vsn(uex_table)`

Receives the `uex_table` structure and returns either a character pointer with the value `NULL` or an address of a string.

This exit allows a site to specify the VSN for a scratch request.

The returned value of `NULL` means no VSN was returned while a pointer to a string is used as the value of the scratch VSN. If no VSN is returned, the TMF daemon uses the default scratch VSN, just as if the user exit had not been taken.

`uex_sm_dex_1(uex_table)`

Receives the `uex_table` structure and returns.

This user exit is called from a child process in the TMF daemon after the TMF daemon has built a dataset enquiry (dex) request for a servicing front end and before it is sent to be processed. The exit provides a site an opportunity to add code, which could add information to the existing dex request or change it in any other way.

When the TMF daemon transfers control to this user exit, the `user_buff` field of the `uex_table` structure contains the address of the dex request and the `user_bytes` field of the `uex_table` structure contains the length in bytes of the memory block that are allocated to hold the dex request. The `festbls.h` header file contains a layout of the data structures making up the format for the delivered dex request.

If this user exit extends the length of the delivered dex request beyond the size of the allocated memory block, it allocates the necessary memory to store the newly composed dex request. The address of this allocated memory block is returned to the TMF daemon in the `user_buff` location. The length in bytes of the newly allocated memory block is returned in the `user_bytes` field. The length in words of the newly composed dex request is returned in the `user_wc` field of the `uex_table` structure. The `update` field of the `uex_table` structure must be returned set to a nonzero value, while the `yes_no` field is returned set to `YES`.

If this user exit does not extend the length of the delivered dex request beyond the size of the allocated memory block, it does not have to allocate another memory block and the address in `user_buff` field remains unaltered. The length in bytes of the allocated memory block in the `user_bytes` field also remains unaltered. The length in words of the newly composed dex request is returned in the `user_wc` field. The `update` field is returned set to a nonzero value, while the `yes_no` field must be returned set to `YES`. If this user exit does not change the delivered dex request in any way, the `update` field must be returned set to zero, while the `yes_no` field is returned set to `YES`. If the user exit determines that the dex request should not be sent to the servicing front end, the `yes_no` field is returned set to `NO`.

When this user exit returns, the TMF daemon checks the value returned in the `yes_no` field. If the value in this field is `NO`, the TMF daemon does not send the request to the servicing front end and continues its processing.

If the value returned in the `yes_no` field is `YES`, the TMF daemon checks the value of the `update` field. If its value is zero, the TMF daemon continues its normal processing. It sends the dex request from the location it has allocated to the servicing front end to be processed. If the value in the `update` field is nonzero, the TMF daemon replaces its value of the length in words of the dex request with the value which is returned in the `user_wc` field. It compares the address of the memory block it has allocated for its dex request and the address which has been returned in the `user_buff` field. If these addresses are the same, the TMF daemon continues its normal processing. If these addresses differ, the TMF daemon frees the memory block it allocated for its dex request and takes the address

from the user_buff field as its replacement, after which it continues its normal processing.

No other uex_table information is used to update data structures in the TMF daemon.

uex_sm_dux_2(uex_table)

Receives the uex_table structure and returns.

This user exit is called from a child process in the TMF daemon at the point where the TMF daemon receives a reply from the servicing front end to a dataset enquiry (dex) request and before it processes this reply. The exit provides a site an opportunity to add code, which could process the reply in accordance to site local requirements. When the TMF daemon transfers control to this user exit, the user_buff field of the uex_table structure contains the address of the dex reply and the user_bytes field contains the length in bytes of the memory block which has been allocated to hold the dex reply. The festbls.h header file contains a layout of the data structures making up the format of the delivered dex reply.

If the user exit determines that the servicing front end has returned a message in the reply, the address of the message is returned to the TMF daemon in the user_tmsgp field of the uex_table structure. It assures the message is properly processed. If the servicing front end has not returned a message in the reply field, user_tmsgp is zero.

The user_error field is provided in case the user exit encounters an error condition that has to abort the TMF daemon child process. When the user exit returns this field set to a nonzero value and the yes_no field of the uex_table structure set to value NO, the TMF daemon passes the value in user_error on to the abort function.

If this user exit completes without errors and updates information in the uex_table structure from the information delivered in the dex reply, it returns a nonzero value to the TMF daemon in the update field of the uex_table structure and in the yes_no field of the uex_table structure value YES. This causes various data structures in the TMF daemon to be updated with uex_table information. The uex_sm_dex_2() in the tmuex.c file contains an example which is based on the way the TMF daemon processes the dex reply. It shows the uex_table fields which have to be updated. If this user exit has determined that the TMF daemon has to do the processing of the dex

reply and completes without updating the `uex_table` fields, exit returns to the TMF daemon with the `yes_no` field set to `YES` and the `update` field set to zero. This prevents the TMF daemon from updating its data structures with information from the `uex_table` structure. If the user exit relies on the TMF daemon to process the dex reply, the reply must be in the format the TMF daemon can handle.

`uex_sm_vax(uex_table)`

Receives the `uex_table` structure and returns an integer value of `YES` or `NO`.

This exit returns an answer to the question "Can user *uid* access volume *vsn*?" It is called in place of the volume access request made to the front-end system.

This routine must validate access for a VSN and set the following:

- Expiration information for the file
- The allowed-permission-bits structure

Returning a nonzero value denies access to the dataset.

`uex_sm_vux(uex_table)`

Receives the `uex_table` structure and returns an integer value of `YES` or `NO`.

This exit provides the opportunity to update tables and log fields after a volume has been accessed. It is called in place of the volume access request made to the front-end system.

The return value `YES` means that the update was successful, while the return value `NO` means that the update failed.

User Exits Not Requiring Configuration

The following user exits are used by TMF commands and do not require user configuration:

`uex_blp`	This exit allows a site to specify who, besides `root`, can bypass label processing. Returning zero denies permission to bypass label processing.
`uex_vsn_ok_to_use`	This exit allows a site to cancel a mount if the user is not allowed to use the specified volumes. Returning zero denies use of the volumes.

License Requirements

The licensing used for TMF on IRIX and SGI ProPack 4 is based on the FLEXlm product from Macrovision Corporation. For TMF on SGI ProPack 5, the licensing is based the SGI License Key (LK) software.

This chapter discusses the following:

FLEXlm Licensing on IRIX and SGI ProPack 4

This section discusses TMF licensing on IRIX and SGI ProPack 4:

Gathering the Host Information for IRIX and SGI ProPack 4

When you order TMF, you will receive an entitlement ID. You must submit the system host ID, host name, and entitlement ID when requesting your permanent TMF license key.

To obtain the host information for an IRIX or SGI ProPack 4 server, execute the following command, execute the following FLEXlm command:

```
/usr/sbin/lmhostid
```

This command displays the FLEXlm host identifier, as shown in the following example:

```
% /usr/sbin/lmhostid
lmhostid - Copyright (c) 1989-2004 by Macrovision Corporation. All rights reserved.
The FLEXlm host ID of this machine is "690c9f5c"
```

When you are asked for the license manager host identifier, provide this FLEXlm host identifier.

For more information, see the *FLEXlm End User Manual*, which provides detailed information on using and administering software licenses.

Obtaining the License Keys for IRIX and SGI ProPack 4

To obtain your TMF license key, see information provided in your customer letter and the following web page:

http://www.sgi.com/support/licensing

Installing the License Keys on IRIX and SGI ProPack 4

For IRIX and SGI ProPack 4 servers, you will install the license keys in the following location:

`/var/flexlm/license.dat`

Do the following:

1. Create the `/var/flexlm` license key directory if necessary.

   For example:

   ```
   # mkdir -p /var/flexlm
   ```

2. Copy the key to the `license.dat` file.

LK Licensing on SGI ProPack 5

This section discusses TMF licensing on SGI ProPack servers:

Gathering the Host Information for SGI ProPack 5

When you order TMF, you will receive an entitlement ID. You must submit the system host ID, host name, and entitlement ID when requesting your permanent TMF license key.

To obtain the host information for an SGI ProPack 5 server, execute the following command (assuming that the LK `rpm` from SGI ProPack has been installed):

```
/usr/sbin/lk_hostid
```

For example, the following shows that the serial number is `N0000302` and the license ID is `e000012e`:

```
[root@linux64 root]# /usr/sbin/lk_hostid

N0000302 e000012e socket=16 core=16 processor=16
```

Obtaining the License Key from SGI for SGI ProPack 5

To obtain your TMF license key, see information provided in your customer letter and the following web page:

http://www.sgi.com/support/licensing

Installing the License Key on SGI ProPack 5

For SGI ProPack 5 servers, you will install the license keys in the following location:

```
/etc/lk/keys.dat
```

Do the following:

1. Create the `/etc/lk` license key directory if necessary. For example:

   ```
   [root@linux cdrom]# mkdir -p /etc/lk
   ```

2. Copy the key to the `keys.dat` file.

Index