IRIX® TMF Release and Installation Guide

007–3967–005 Version 1.3.5

Cover design by Sarah Bolles Design, and Dany Galgani, SGI Technical Publications

New Features in This Guide

New features include changes in the format of the SGI node locked licenses issued for SGI Origin 300, SGI Origin 3000, and SGI Onyx 3000 series systems.

Record of Revision

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

Contents

Tables

Procedures

About This Guide

This guide documents the release and installation of the Tape Management Facility (TMF).

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 prose documentation, online man pages describe aspects of TMF. Man pages exist for the user commands, devices (special files), file formats, miscellaneous topics, and administration commands.

Individual man pages are available online and can be accessed by using the `man`(1) command as shown in the following example:

```
% man tmstat
```

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

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

Each man page includes a general description of one or more commands, routines, system calls, or other topics, and provides usage details (command syntax, parameters, and so on).

Obtaining Publications

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

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

Conventions

The following conventions are used throughout this document:

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

Reader Comments

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

You can contact us in any of the following ways:

- Send e-mail to the following address:

  `techpubs@sgi.com`

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

  `http://techpubs.sgi.com`

- Contact your customer service representative and ask that an incident be filed in the SGI incident tracking system.
- Send mail to the following address:

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

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

We value your comments and will respond to them promptly.

Introduction

This guide documents the Tape Management Facility (TMF) release 1.3. This release is supported by the hardware and software described in Section 3.3, page 9.

The guide describes the following release and installation topics:

- Software overview including TMF features and the differences between the UNICOS tape subsystem and TMF (Chapter 2, page 3)
- Release package contents, requirements, licensing, ordering directions, and support (Chapter 3, page 9)
- Installation overview including preparations and directory structure (Chapter 4, page 13)
- Procedures for installing TMF (Chapter 5, page 15)
- Building TMF with optional user exits (Chapter 6, page 17)
- FLEXlm license requirements (Chapter 7, page 29)
- Upgrading TMF (Chapter 8, page 31)

Software Overview

This chapter provides an overview of TMF features and lists differences between the UNICOS tape subsystem and TMF. For a description of TMF capabilities and the role of the IRIX system administrator, refer to the *TMF Administrator's Guide*.

2.1 Overview

It is not uncommon for large SGI customers to manage hundreds of thousands, even millions of reels or cartridges of valuable data in tape libraries. These sites typically service thousands of mounts and read/write terabytes of data per day, 7 days a week, 24 hours a day.

Managing and delivering high bandwidth I/O reliably to sites of this size requires sophisticated data center system software, optimized I/O device drivers, and high performance tape and disk options. SGI provides the data management solutions for users who require fast, reliable access to massive amounts of data.

2.1.1 Architecture

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

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

TMF is started by the system operator or the system administrator, or it is started automatically as part of the system startup. TMF has superuser privileges. Therefore, it can communicate directly with the TMF device driver and the IRIX 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

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

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

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

2.1.2 Tape Label Support

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

2.1.3 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.

2.1.4 Volume Mounting and Unmounting

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

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

2.1.5 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.

2.1.6 Front-End Servicing

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

2.1.7 User End-of-Volume Processing

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

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

2.1.8 Multifile Volume Allocation

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

2.1.9 Concatenated Tape Files

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

2.1.10 Tape Message Log File

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

2.1.11 Device Support

For this release, TMF provides support for the following tape libraries and tape devices:

- StorageTek libraries using the ACSLS software interface
- IBM 3494 libraries using the CPS software interface
- EMASS libraries using the VolServ software interface
- IBM 3590 tape drives
- StorageTek devices
  - 9840
  - 4481, 4480
  - 4491, Silverton 4490
  - RedWood SD-3
  - TimberLine 9490
- DLT 4000 and DLT 7000 tape drives

TMF also supports OpenVault and the OpenVault libraries.

2.2 Differences between the UNICOS Tape Subsystem and TMF

The basic structure of the TMF release 1.3 (for IRIX environments) is the same as the tape subsystem in the UNICOS 10.0 release. Changes were made to update the product in areas affected by operating system dependencies, and there have been changes in basic terminology.

These changes are briefly described as follows:

- TMF 1.3 product installation is different from the procedures used for the UNICOS system. The TMF release 1.3 has been adapted to use installation procedures similar to those used for installing the Data Migration Facility (DMF).
- In the TMF configuration file, TMF supports the `AUTOCONFIG` statement instead of the `IOP`, `IONODE`, `CHANNEL`, `BANK`, `SLAVE`, and `CONTROL_UNIT` statements. The `AUTOCONFIG` statement is composed of `DEVICE` statements.

- Because the IRIX system does not support the UNICOS user database feature (UDB), bypasslabel processing is only available with `root` permission.
- Accounting and security are not supported under TMF.
- The first two letters of TMF commands are `tm` in contrast to UNICOS commands that begin with `tp`. For example, the UNICOS command, `tpmnt`(1), is the TMF `tmmnt`(1) command.

  TMF files begin with `tmf` in contrast to the `tape` prefix of UNICOS files. For an example, see the `tmftrace`(5) man page; also see its UNICOS counterpart, `tapetrace`(5).
- Table 2-1 shows TMF and UNICOS name differences:

Table 2-1 Name Differences

TMF Name	UNICOS Name
`tmrls`(1)	`rls`(1)
`tmrsv`(1)	`rsv`(1)
`tmfdaem`(4)	`tpddem`(4)
`tmf.config`(5)	`text_tapeconfig`(5)
`tmfctl`(5)	`tapereq`(5)
`tmstop`(8)	`tpdstop`(8)
`tmunld`(8)	`tpu`(8)

  For more information, use the `man`(1) command to display any of the TMF man pages.
- The `msgi`(1), `infd`(8), `tpquery`(1), `tpapm`(8), `tpbmx`(8), `tpcore`(8), `tpdev`(8), `tpformat`(8), `tpinit`(8), `tpscr`(8), and `xtpldr`(8) commands are not supported under TMF.
- On TMF, the `tpdfixup` utility is replaced by the `tmcollect`(8) utility. For more information, see the `tmcollect`(8) man page.
- TMF determines configuration when it starts; as a result, control units used in the UNICOS tape subsystem are not specified.

Release Package

This chapter provides information on the TMF release package and the software and hardware platforms that TMF supports.

3.1 Release Package Contents

The TMF release 1.3 package includes the following:

- A CD-ROM that contains the installable binary packages for the TMF release 1.3
- *TMF Administrator's Guide*
- *TMF Release and Installation Guide* (this publication)
- *TMF User's Guide*
- A TMF entitlement number for licensing (see Chapter 7, page 29)

3.2 Release and Packaging for FFIO

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

3.3 Hardware and Software Requirements

The TMF release 1.3 is supported on 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 on the 32-bit SGI Challenge S system running the IRIX 6.5.2m or later operating system.

3.4 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.

3.5 Ordering the TMF Release

You can order this TMF release in the following ways:

- Customers can download the software and a temporary license from the following URL:

  `http://www.sgi.com/Products/Evaluation/#tmf`

- Customers outside of the United States and Canada can contact their local service or sales organization for ordering information.

Software will be shipped by ground service or 5-day international service unless otherwise requested.

3.6 Ordering Publications

To order SGI documentation, go to the SGI Technical Publications Library at `http://techpubs.sgi.com`. Find the title that you want and choose "order" to get the ordering information page for that document.

3.7 Documentation Support

The release package contains the documentation in InSight format. All of the TMF publications are included on the TMF media.

The documentation for this TMF release includes the following:

- *TMF Administrator's Guide*, which contains information about configuring, administering, and troubleshooting TMF.

  With this release, automatic volume recognition (AVR) and support for OpenVault has been added.

- *TMF Release and Installation Guide* (this publication).

- *TMF User's Guide*, which describes how to use TMF. It provides information on using tape formats, performing basic tape procedures, and writing C tape applications.
- Man pages. Man pages are preformatted files containing information about commands and other aspects of operating systems or compatible products. Table 3-1 lists the man pages in this release.

Table 3-1 TMF Man Pages

Category	Man Page
User commands	`msgr`(1), `tmcatalog`(1), `tmlist`(1), `tmmnt`(1), `tmrls`(1), `tmrst`(1), `tmrsv`(1), `tmstat`(1)
Administrator commands	`msgd`(8), `msgdaemon`(8), `msgdstop`(8), `newmsglog`(8), `oper`(8), `rep`(8), `tmclr`(8), `tmcollect`(8), `tmconf`(8), `tmconfig`(8), `tmdaemon`(8), `tmfrls`(8), `tmgstat`(8) `tmlabel`(8), `tmmls`(8), `tmmql`(8), `tmset`(8), `tmstop`(8), `tmunld`(8)
Devices (special files)	`tmfdaem`(4)
File Formats	`tmf.config`(5), `tmfctl`(5), `tmftrace`(5)
Miscellaneous topics	`tmf`(7)

Installation Overview

A successful installation of TMF at a site is comprised of four steps:

1. Installing TMF from the World Wide Web (WWW) or from CD-ROM using Software Manager (see Chapter 5, page 15)
2. Optionally modifying user exits and rebuilding TMF (see Chapter 6, page 17)
3. Obtaining and editing the FLEXlm license file (see Chapter 7, page 29)
4. Configuring TMF

Information on installation, building TMF, and FLEXlm license requirements are in this manual. TMF configuration as well as TMF and message daemon startup procedures and detailed FLEXlm editing instructions are described in the *TMF Administrator's Guide*.

This chapter describes installation preparations and the TMF directory structure. The procedures described in this guide are used for the installation of major releases, revisions, and product upgrades delivered on the release media.

4.1 Preparing for Installation

Before you begin a TMF installation, ensure that you meet the following requirements:

1. Verify that you are `root`.
2. Be sure that you have backed up the current installation material if you are installing a TMF replacement or upgrade.
3. Save your current TMF configuration file if you are upgrading TMF.

   You may be able simply to replace the sample file, which is delivered with the release materials, with your current TMF configuration file. For more information on anything that may affect changes in your configuration file, see the `README` file.

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

---

4. Be sure `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.
5. Make sure TMF is not executing before beginning the installation process.

4.2 TMF Directory Structure

The installation material creates a base directory of `/usr/tmf/`*version*. Table 4-1 describes the directories and files in this directory:

Table 4-1 TMF Directories and Files

File	Description
`bin`	TMF executables. A symbolic link from `/usr/tmf/bin` is created to this `bin` directory.
`include`	Include files which users need to compile programs that access TMF services. A symbolic link from `/usr/include/tmf` is created to this `include` directory.
`modules`	TMF `modules` file which may be installed in the local `/opt/modulesfile` directory. `/usr/tmf/bin` will be appended to the user's `bin` path.
`tmf.config`	Default TMF configuration file. A symbolic link from `/etc/config/tmf.config` to this file is created for the running version of TMF. For information on configuring TMF, see the *IRIX TMF Administrator's Guide*.
`uex.tar`	Tar file. It contains the user exit files, which, if required, you can use to rebuild TMF with user exits.

Installing TMF

This chapter describes two methods that you may use to install this TMF release. If you want to download the TMF software from the World Wide Web, complete the steps in Procedure 5-1. If you have ordered the software on a CD-ROM, complete the steps in Procedure 5-2, page 16.

Note: To install this TMF release, you are required to have `root` permission.

Note: Each time TMF is installed, it must be started. For information on starting and stopping TMF, see the *TMF Administrator's Guide*.

Procedure 5-1 Downloading the Software from the World Wide Web

1. Go to the following URL:

   `http://www.sgi.com/Products/Evaluation/#tmf`

2. If you have a license, skip to Procedure 5-1, step 4, page 15. If you do not have a license agreement, click on **Eval.Lic.** to obtain an evaluation license. Read the form and click on **Accept Terms**.

3. After you fill out the Evaluation Software Request Form and enter **Submit**, a temporary FLEXlm license file will be mailed to you. You will need the license when initializing TMF.

4. Click on the link for the **README** file. It contains essential information about functionality, software support, patches, and so on.

5. Select the desired distribution package for the system and click on it.

6. Enter the `root` password if you are prompted for it. (If you began the browser session with `root` permission, you are not prompted.) You will see a message that says:

   ```
   Please wait: Initializing Software Manager
   ```

   Software Manager is launched in a new window.

7. Click on **Customize Installation**. You will receive more information about the size of the TMF software including the directories and files. Click on the folder icon to view the contents of the software package.
8. Click on 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.
   ```

9. Select **File > Exit**.
10. Proceed to Chapter 6, page 17, and review before moving to the configuration chapter in the *TMF Administrator's Guide*.

Procedure 5-2 Installing the Software from the CD-ROM

1. Place the CD-ROM in the drive.
2. Using the left mouse button, select **System->Software Manager** on the pulldown menu.
3. On the **Available Software** list, select **/CDROM/dist**.
4. Click on **Customize Installation**. You will receive more information about the size of the TMF software including the directories and files. Click on the folder icon to view the contents of the software package.
5. Click on 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 **File > Exit**.
7. Eject the CD-ROM.

Building TMF

The TMF release package provides the capability to install 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 Section 6.1, page 17.

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. For descriptions of the individual user exits, see Section 6.2, page 19.

6.1 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 Section 6.3, page 28).

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/config/tmf, 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/tmf/version
$ 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/tmf/version
$ make uex
```

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

```
$ make install
```

The TMF installation is now complete.

6.2 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 which 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 which 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, which, in 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 three 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.

6.3 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.

FLEXlm License Requirements

The software licensing used by TMF is based on the FLEXlm product from GLOBEtrotter Software, Inc. For all supported platforms, a FLEXlm license is required to use TMF.

TMF includes a temporary license so that you may install and use TMF on a temporary basis; you must get a permanent TMF license and restart TMF once you have that license.

For more information on FLEXlm, you may order the *Flexible License Manager End User Manual* from GLOBEtrotter Software, Inc.

The TMF license is issued to a specific host identifier. You will be asked to provide the license manager host identifier when you obtain your permanent license. To obtain the host identifier of the system on which you will run TMF, execute the following FLEXlm command:

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

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

```
fsgi366% /usr/sbin/lmhostid
lmhostid - Copyright (C) 1989-2001 Globetrotter Software, Inc.
The FLEXlm host ID of this machine is "690c9f5c"
fsgi366%
```

If you are using an Origin 200, Origin 2000, Origin 300, Origin 3000, or Onyx 3000 series system, the `lmhostid` command displays more than one host identifier, as in the following example. You can use any of them to obtain your permanent license.

```
clink$ /usr/sbin/lmhostid
lmhostid - Copyright (C) 1989-2001 Globetrotter Software, Inc.
The FLEXlm host ID of this machine is ""b0011086 b0010981""
Only use ONE from the list of hostids.
clink$
```

When you are asked for the license manager host identifier, provide this FLEXlm host identifier. You must have a separate license for each host on which TMF is installed.

When you download or order TMF, you will receive a temporary license and an entitlement number.

Install the TMF software with the temporary license as described in Chapter 3, page 9, through Chapter 6, page 17, and configure the system as outlined in the configuration chapter of the *TMF Administrator's Guide*.

Along with your entitlement number, you will receive a URL to a key generation page. To obtain your permanent license, follow the instructions on the key generation page. After you have provided the required information, a key will be sent to you through electronic mail.

If for some reason you cannot use the World Wide Web key generation page, you can contact the order desk at +1 651 683 5907.

Upgrading TMF

Upgrading TMF software is essentially the same procedure as the one described in Procedure 5-1, page 15, except that you select **Upgrade** from the World Wide Web page.

The upgrade procedure differs from an initial installation in the following ways:

- You can download the TMF software while an old version is active.
- You should not install the new version on a running system. Doing so updates the high-level links to point at the new TMF version.

Index

R

S

T