DMF Recovery and Troubleshooting Guide for IRIX® Systems

007–3682–006

Cover Design by Sarah Bolles, Sarah Bolles Design, and Danny Galgani, SGI Technical Publications

New Features

Major new features of release 2.7 are distributed commands and the Library Server.

Record of Revision

Version	Description
2.6.1	January 1998 Original printing to support the Data Migration Facility (DMF) release 2.6.1 running under SGI IRIX systems.
2.6.2	July 1998 Reprint to support the Data Migration Facility (DMF) release 2.6.2 running under SGI IRIX systems.
2.6.2.2	December 1998 Reprint to support the Data Migration Facility (DMF) update release 2.6.2.2 running under SGI IRIX systems. The only change associated with this manual was a change in the title.
004	May 2000 Reprint to support the Data Migration Facility (DMF) update release 2.6.3 running under SGI IRIX systems. Only minor editing changes are included.
005	October 2000 Reprint to support the Data Migration Facility (DMF) update release 2.6.3.2 running under SGI IRIX systems. Only minor editing changes are included.
006	April 2002 Reprint to support the Data Migration Facility (DMF) update release 2.7 running under SGI IRIX systems.

Contents

Figures

Tables

About This Guide

This publication documents administrative troubleshooting for the Data Migration Facility (DMF), release 2.7, on SGI systems running the IRIX operating system 6.5 and later releases.

Related Publications

The following documents contain additional information that may be helpful:

- *DMF Administrator's Guide for IRIX Systems* describes DMF configuration and administration.
- *DMF Release and Installation Guide for IRIX Systems* contains release-specific information about features and describes how to install DMF.

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.

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.
`manpage`(*x*)	Man page section identifiers appear in parentheses after man page names.
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 manual tells you how to use the `dmaudit`(8) command to detect and report every known type of discrepancy between your file systems and the DMF daemon database, including the following:

- Migrated files for which there are no database entries
- Database entries for which there are no migrated files
- Duplicate file bit file identifiers (bfids) in either the file systems or the database (the bfid is the ID assigned to each file during the migration process; it links a migrated file to its data on alternate media)

The `dmaudit` command is intended primarily for interactive use. It uses a series of scrolling menus to display information and to solicit option selections. However, after you complete the initial configuration, `dmaudit` can also accept the `snapshot` and `report` operations from the command line. This allows `dmaudit` to be run as a background process or in batch mode.

Some of the advantages of using `dmaudit` are as follows:

- `dmaudit` executes while DMF is active. Users can continue to access their migrated files while you simultaneously search for and correct errors.
- `dmaudit` is accurate. `dmaudit` automatically adjusts for any user activity occurring in file systems it searches. It therefore is not confused into reporting false errors, nor does it miss errors.
- Discrepancies can be examined interactively. `dmaudit` performs most of its analysis in batch mode and saves the output in indexed files. You can then interactively examine any discrepancy quickly and easily.
- You can decide which errors you want to fix, and you have some control over how a discrepancy is to be fixed.
- You do not need to immediately fix discrepancies. Because `dmaudit` saves all information it needs to fix each error, the errors can be fixed hours or days after they are detected.

Note: The `dmaudit` command is capable of showing you **what** discrepancies exist, but is not able to tell you **why** they happened. Discovering why an error occurred requires some detective work and a considerable knowledge of the internal workings of DMF.

If you do want to determine why a discrepancy occurred, there is information available in some of the `dmaudit` menus to help you narrow down exactly when an error occurred. Examination of daemon logs and journal files may fill in the remaining blanks. Some tips are given in later sections of this manual to help you determine why certain discrepancies occurred.

The Need for `dmaudit`

During normal system operation, the daemon database and the file systems stay synchronized with each other. Each migrated user file in your file systems has a unique bfid. Each bfid has one or more active daemon database entries. If an entry in the daemon database is not in use, it is soft-deleted. (A database entry is soft-deleted when the MSP or volume group (VG) copy of the data is no longer current. Data remains on the alternate media until the database entry is deleted.)

However, things can get out of synchronization. Examples of some of the discrepancies that might occur are as follows:

- Migrated files that have bfids for which no database entries exist
- Active database entries for which no migrated files exist
- Multiple user files that have the same bfid

System crashes are a major source of discrepancies because I/O operations in progress at the time of the crash are not guaranteed to complete successfully. For example, a migrating file might receive a new bfid, but the rewrite of its inode to disk might not succeed. Or perhaps the inode update does complete, but the corresponding database entries are not successfully made.

Inconsistencies also arise if users are allowed to modify or remove migrated files during periods when the daemon is not running, because the kernel is then unable to tell the daemon to soft-delete the corresponding database entries. The unused, or orphan, database entries then accumulate in the database, wasting space on the alternate media.

Use of the `xfsdump`(1m) and`xfsrestore`(1m) commands can also create inconsistencies.

For example, files that have been removed or modified can be restored to their previous state. If the same migrated file is restored multiple times, there will be more than one inode containing the same bfid.

Sometimes the inconsistencies are harmless, or only result in wasted space on the alternate media. In other cases, the discrepancies can prevent a user from accessing one or more files, or can result in the loss of files. `dmaudit` allows the administrator to quickly detect and correct such inconsistencies when they occur, possibly before any data loss becomes permanent.

When to Use `dmaudit`

After `dmaudit` has been initially configured, most sites use `dmaudit` in batch mode on a periodic basis, perhaps once a week. The easiest way to do this is through the use of a `cron` script. Sites may also want to use `dmaudit` after a known failure such as a major system crash. Instructions on how to generate a report both interactively and through a `cron` script are provided in Chapter 6, "Detecting Discrepancies", page 27.

When errors are detected, you should interactively examine and correct them.

Running `dmaudit` on a periodic basis will help you determine why discrepancies have appeared. For example, if your system crashes and a subsequent `dmaudit` run shows discrepancies that were not previously present, you can be reasonably sure that they occurred because of events at the time of the crash.

Daemon Internals

To use `dmaudit` effectively, you must understand how DMF keeps track of copies of a file's data stored on alternate media, how that data is restored to disk when the user accesses the file, and what happens to those copies when the file is modified or removed.

"Bfids and Bfid Sets" explains how migrated files are identified, and "Migration Life-Cycle", page 11, provides an overview of the various bfid-set states that are part of the DMF process.

Bfids and Bfid Sets

The bit file identifier (bfid) is an object that links a migrated file to copies of its data on alternate media (such as tape). The daemon assigns a unique bfid to each file that it migrates.

A bfid consists of an opaque 16-byte value. (*Opaque* in this context means that the content and format have no fixed definition. The value can be interpreted differently by different processes.)

The bfid represents a unique ID that the daemon inserts into a migrated file's inode and into database entries that point to copies of the file's data. No two migrated files on the same machine should have the same bfid.

For `dmaudit` purposes, a migrated file is one whose inode contains a bfid. A file that does not have a bfid is a nonmigrated file (often referred to in DMF documentation as a regular file).

A *bfid set* is the collection of all database entries, special files, and migrated files associated with a particular bfid. The different components that make up a bfid set define its state. During the normal course of migrating and moving files, there are four states that a bfid set can be moved through. Any time that a bfid set contains a combination that differs from one of the four states, it is considered an error.

To understand these bfid-set states, you must understand how DMF uses each of the possible components of a bfid set. "Migration Life-Cycle", page 11, describes possible states and how files move from one state to the next. The remaining sections describes how the DMF daemon handles migration errors, remigration, and modification of files. "DMF Bfid State Summary", page 15, summarizes the information.

DMF Inode-Resident Data

Data migration uses fields within a file's inode to store the bfid, the state of the file, and other information. `dmaudit` retrieves and reports this information when discrepancies are discovered. The information is retrieved and reported by `dmaudit`. These fields can be examined by anyone with proper access permission to a file using the `stat` system call.

Figure 2-1 shows the DMF fields in the structure returned by `stat`, together with definitions of the possible values for some of those fields.

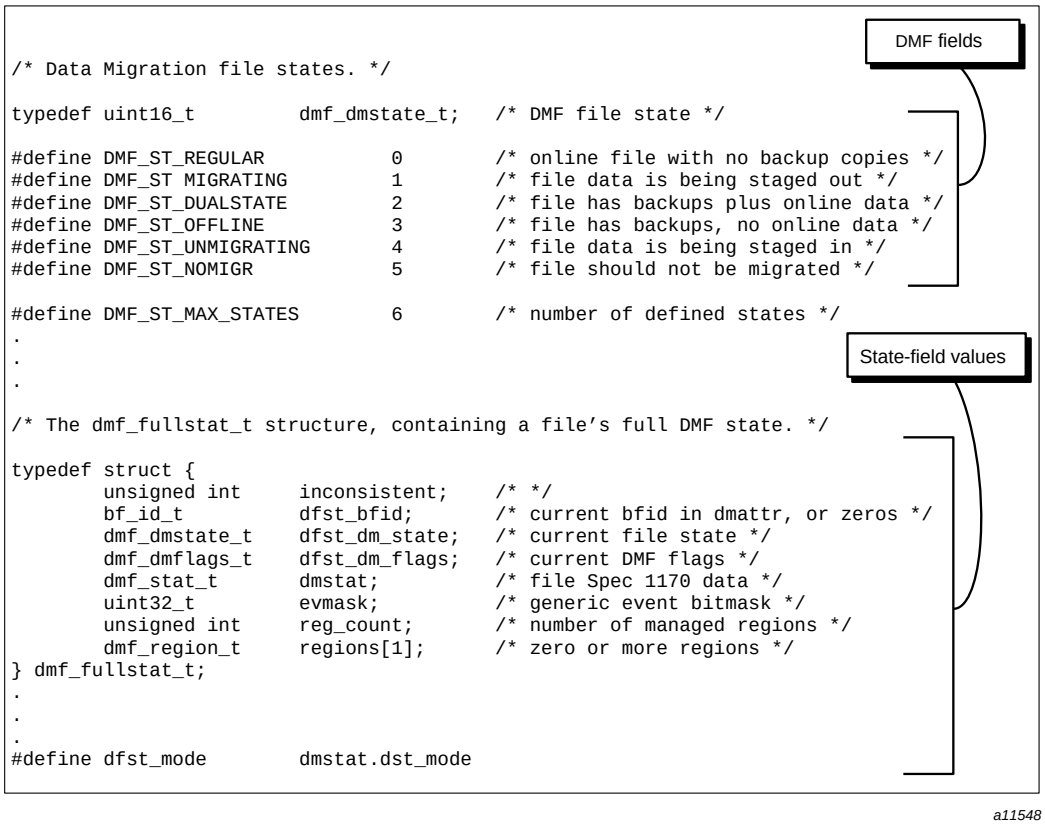

Figure 2-1 DMF fields

The Bit File Identifier (Bfid)

The bfid is assigned by the daemon. When a file's inode contains a bfid, it indicates that the file is under DMF control. This field is used as a key into the daemon database to check for validity of the file against the database.

DMF File State Field

The state field shows the current migration state of a file. Supported states are as follows:

Table 2-1 DMF file states

State	Description
`DMF_ST_REGULAR`	A *nonmigrated file*; that is, a file that contains data and whose inode has not yet been assigned a bfid.
`DMF_ST_MIGRATING`	A *migrating file*; that is, a file that is in the process of migrating. The inode may or may not contain a bfid.
`DMF_ST_DUALSTATE`	A *dual-state file*; that is, a file that contains data and whose inode also contains a bfid pointing to matching copies of that data on alternate media.
`DMF_ST_OFFLINE`	An *offline file*; that is, a file whose inode has been assigned a bfid that points to copies of the file's data on alternate media, and all data in the file itself has been removed.
`DMF_ST_UNMIGRATING`	An *unmigrating file*; that is, a file created when a file is recalled. It holds the data pointers until the MSP or VG successfully copies the file's data back onto the file system disk from alternate media. When the copy is complete, the daemon moves the data pointers from the unmigration file inode back into the user file's inode.
`DMF_ST_NOMIGR`	This state is never assigned to user files. The daemon uses it exclusively for unmigration files. The `DMF_ST_NOMIGR` state indicates to programs and to the daemon that this file should never be chosen as a candidate for migration.

Although the daemon is usually responsible for changing the state of a file, an `xfsrestore`(1m) command may also have to do so when restoring a file system from tape. The kernel changes a file's state when the file is modified or removed.

DMF Daemon Database Contents

Four files make up the DMF daemon database. They reside in the daemon's home directory (*HOME_DIR*/*daemon_name*; *HOME_DIR* is specified in the DMF configuration file; *daemon_name* is the name of the `daemon` object in the configuration file). The files are as follows:

- `dbrec.dat` is the data file containing the base record information about each file copy.
- `dbrec.keys` is the index file used to access individual records in the data file. The daemon uses the bfid as the index file key; it has the following format:

```
typedef struct {
u_char  bfid[16];
} bf_id_t;
```

- `pathseg.dat` is the data file containing the overflow `path` segment blocks for any base (`dbrec`) records that require extra `path` field characters.
- `pathseg.keys` is the index file used to access the `path` segment overflow records for any base (`dbrec`) records that apply.

The daemon inserts a new entry into its database each time a copy of a file is created. Each database entry contains the same bfid that was stored in the file being migrated.

There are three types of entries in the daemon database, each of which serves a different purpose:

- An *incomplete database entry*. This entry is created when a user file is currently under migration. It represents a database entry with an empty `path` field in the `dbrec` structure (shown following this list).
- A *complete database entry*. This entry is created when the media-specific process (MSP) or volume group (VG) has completed the file's migration and has entered the file's MSP/VG key into the `path` field.
- A *soft-deleted database entry*. This entry is created when the `delflag` field of the entry is nonzero, indicating the time that the deletion occurred. Although the entry physically exists in the database, no corresponding user file should exist for the entry. When soft-deleted entries are physically deleted from the database they are said to be *hard-deleted*.

The following `dbrec` structure is used by the daemon to store the contents of one database entry in the `dbrec.dat` file:

```
struct  dbrec   {
    bf_id_t bfid;           /* off line file bfid */
    dev_t   origdv;         /* original device no. */
    ino_t   origino;        /* original inode number */
    off64_t origsz;         /* original file size (in blocks) */
    time_t  otime;          /* original entry time */
    time_t  utime;          /* last update time */
    time_t  ctime;          /* last check time */
    time_t  delflag;        /* delete time */
    int userid;             /* user id, when archived */
    int pathlen;            /* length of complete path */
    char    ofilenm[15];    /* original file name (last element) */
    char    proc[8];        /* process name */
    char    path[34];        /* MSP key string */
};
```

Each of the fields in a database entry is explained in Table 2-2. The fields that `dmaudit` uses in its analysis are marked with an X; other fields are not used by `dmaudit` in its analysis because they are not always the most current information and their data does not help `dmaudit` solve inconsistencies. You, however, may be able to use them as hints to determine why an error occurred. For example, although it is permissible for a file's ownership to change at any time, in practice that seldom happens, so the user ID of the person who owned the file when it originally migrated is often still the most current information.

Table 2-2 Database fields

Field	Used by `dmaudit`	Description
`bfid`	X	Key by which the daemon accesses its database entries.
`origdv`		Specifies the device number of the file system in which the file resided at the time it was migrated. If you change your disk configuration, the device numbers of your file systems may change, causing `origdv` to become out-of-date.
`origino`		Specifies the inode number of the migrated file at the time it was migrated. This can also become out-of-date if you dump and restore the file system that contains the migrated file.

Field	Used by `dmaudit`	Description
`origsz`	X	Specifies the size of the migrated file in bytes. Because the size of a file cannot change while it is migrated, this field is always current.
`otime`		Specifies the date and time when this database entry was first created. This is sometimes called the *entry origination time*. Usually it corresponds to the time the file first migrated.
`utime`		Specifies the date and time when any field in this database entry was last changed.
`ctime`		Specifies the origination time when the database entry was created. Although this can be reset by the administrator using the `dmdadm`(8) command, in normal practice it never changes.
`delflag`	X	Specifies whether a database entry is valid or invalid (soft-deleted). This value is initialized to 0 when the database entry is created, and it remains set to 0 as long as the database entry is still valid. When the database entry becomes invalid (for example, if the migrated file was removed), it is soft-deleted by storing the current date and time into `delflag` in time-stamp format. The database entry cannot be removed immediately because the migrated file might later be restored. You hard-delete the soft-deleted database entries by using the `dmhdelete`(8) command only when there is no further chance of the file being restored.
`userid`		Specifies the user ID (UID) of the owner of the file at the time the file was migrated. This can become out-of-date if someone uses the `chown` command to change the ownership of the file after it has migrated.
`pathlen`		Specifies the length of the `path` field described below. This field is used to regulate the creation of `pathseg` records to contain the overflow characters of the path that will not fit into the `path` field of this record.
`ofilenm`		Specifies a null-terminated string composed of up to the first 14 characters of the base name of the file if the path name was part of the original migrate request. If the path name was not known at the time the file was migrated, this field contains the string `/NONAME`.

Field	Used by dmaudit	Description
proc	X	Specifies the name of an MSP or VG expressed as a null-terminated character string. This field tells the daemon which MSP or VG to contact to retrieve a copy of the migrated file.
path	X	Specifies either a path name or a key. For MSP and VG dmdaemon database entries, it is the key that the MSP/VG uses to retrieve a copy of the file. For the tape MSP or VG, the path field is a key into an MSP/LS CAT (catalog) database that contains information about the tape that should be mounted to retrieve the copy. It is a null-terminated string.

The path field of the dbrec base record is a fixed-length field, making the dbrec record a fixed-length record. The path field is supplied by the MSP/LS and can be of any length. The daemon determines the fixed length of the path field of the dbrec record. If the value that the daemon must store in the path field to accommodate the MSP/VG is longer than the fixed length, the daemon will allocate a sufficient number of pathseg records to hold the overflow path characters.

The pathseg records are keyed from the dbrec bfid and proc fields that uniquely define a dbrec record. When a daemon database record is accessed, any pathseg record path segment extensions are concatenated with the dbrec path field to accurately reconstruct the path value that the MSP/VG originally supplied to the daemon.

The path field length of the dbrec structure as it is supplied to DMF has a value of 34, which will accommodate the largest dmatmsp/dmatls bfid without requiring any overflow pathseg records. If you are running only dmatmsp or dmatls, you should not need to adjust the path field of the dbrec structure. However, the most efficient use of disk space for a tape MSP/LS is a path field length of 26.

The procedure for addjusting the path field of the dbrec structure is described in *DMF Administrator's Guide for IRIX Systems*.

Migration Life-Cycle

For you to be able to understand all the errors and actions reported by dmaudit, you must understand the valid bfid set states. Figure 2-2, page 12, shows the five states a bfid set can be in during its lifetime, and the arrows connecting various states show what actions cause a bfid set to progress from one state to another.

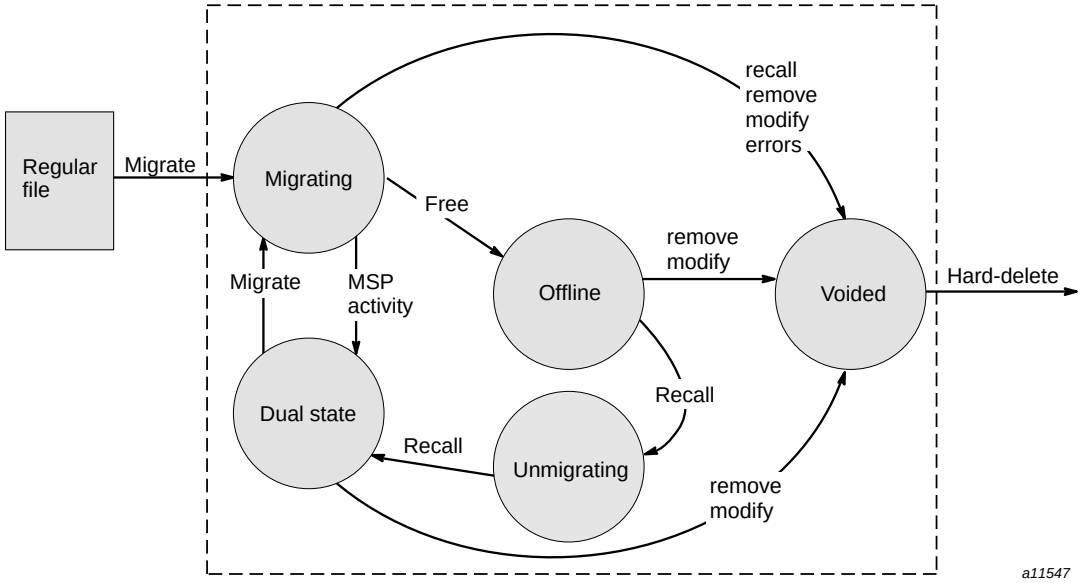

Figure 2-2 DMF bfid-set state diagram

Recognizing invalid states is the essence of what `dmaudit` does. It collects all information available for a bfid set and attempts to determine which bfid set state best matches the available data. When this has been determined, `dmaudit` then reports as inconsistencies any deviations from the ideal bfid-set state and determines what actions are necessary to return the bfid set to a correct state.

The following sections describe file states and state transitions.

Regular Files

A regular file contains no DMF state information. The file's inode contains no bfid and there are no MSP or VG `dmdaemon` database entries corresponding to the file.

Incompletely Migrated State Files

When a migration request is issued on a regular file (either through a `dmput`(1) command or through automated space management) the DMF daemon sets the file state to `DMF_ST_MIGRATING`, assigns a bfid to the file, and creates an incomplete

MSP/VG dmdaemon database entry containing the bfid and other information relating to the file.

Dual-State Files

When all copies of the file's datablocks have been written to the associated media of their respective MSPs/VGs, the file's state is set to DMF_ST_DUALSTATE. This indicates that the file is online and has one or more complete backup copies. In this state, the file's inode contains a bfid. In addition, each copy of the file has a corresponding complete entry in the dmdaemon database.

Similarly, when an offline file is recalled (either explicitly by using a dmget(1) command or by a normal file system access to the file data (such as a read system call)) the file blocks are retrieved from the backup media and placed back into the file system and the file's state is changed to DMF_ST_DUALSTATE.

Offline-State Files

If the file's data blocks were released from the file system (either due to an automated space management policy or because the owner executed a dmput -r command), the file's state is set to DMF_ST_OFFLINE. The file's inode contains a valid bfid, the file's underlying data blocks are released, and a complete MSP/VG dmdaemon database entry exists for the file.

Voided-State Files

A file backup copy can be voided when any of the following occurs:

- An offline file is modified
- A dual-state file is modified
- A migrating file is modified or recalled

In these situations, the bfid is cleared from the file and the dmdaemon database entry is soft-deleted.

Errors during Migration

Sometimes an MSP or VG may encounter problems that prevent it from completing its copy of a file. For example, the tape MSP could encounter tape errors that prevent it from proceeding. The failing MSP then returns an error code to show that it was unsuccessful.

When the daemon receives the error reply, it immediately soft-deletes the MSP's or VG's `dmdaemon` database entry (or entries, if more than one MSP/VG has an entry). The daemon does soft-deletes entries by placing the current date and time into a `delflag` field.

The daemon must then *void the bfid* (remove the bfid from the user file inode). This is done because the user file could not be migrated exactly as requested.

The daemon then sets the file's state to `DMF_ST_REGULAR`.

Remigrating a File

The process of migrating a dual-state file is sometimes known as *remigrating* the file. There is only one minor difference between remigrating a file and migrating a file for the first time: the amount of time the migration takes.

When a dual-state file is remigrated, the daemon checks to see whether a copy of the file already exists on each of the requested MSPs and VGs . If so, the daemon does not have to instruct the MSP or VG to make a new copy of the file, because the old copy is still valid.

Unless the DMF configuration has changed, a dual-state file always migrates to the same MSPs and VGs to which it originally migrated. This means that the bfid set only spends a few milliseconds in the incompletely migrated state before advancing to the fully migrated state.

If the dual-state file is being migrated to a new MSP or VG, a considerable amount of time is spent in the incompletely migrated state while waiting for the new MSP/VG to make a copy of the file.

Removing or Modifying a File

When a user either modifies or removes a file that contains a bfid, the kernel notifies the daemon by generating an event for the file that the daemon has registered to

receive. If the user modified the file, a write event is generated and the daemon will remove the bfid from the inode and change the file's state to `DMF_ST_REGULAR`, because the copies of the file are no longer current. If the user removed the file, a destroy event is generated and the daemon will not need to do anything to the file (because it is no longer present).

In either case, the daemon will soft-delete all remaining database entries for that bfid to indicate that those copies are no longer valid.

DMF Bfid State Summary

Table 2-3 shows a summary of the information presented in previous sections.

Table 2-3 State change map for DMF

DMF bfid-set state	User file state	MSP/VG entries	Entries soft-deleted?
Incompletely migrated	Migrating	At least one incomplete[1]	No
Fully migrated	Dual state	All complete	No
Freed	Offline	All complete	No
Incompletely unmigrated	Unmigrating	All complete	No
Voided	[2]	Either complete or incomplete	Yes

`dmaudit` uses the information in this table when it examines all the information available for each bfid set. Any bfid set that exactly matches one of the entries in the appropriate table is considered error-free.

If `dmaudit` finds a bfid set that does not fit exactly into one of the entries in the table, it is reported as having errors, and `dmaudit` lists the actions necessary to remove the inconsistencies.

1 In the incompletely migrated state, complete MSP/VG entries may also exist.
2 The file state is regular or the file has been removed.

How dmaudit Detects Errors

The dmaudit command collects and analyzes information from the file systems and the daemon database, and stores that information in a directory known as the *working directory*. This process is known as *taking a snapshot* and is normally done in batch mode. dmaudit takes a snapshot by performing the following actions:

1. Scans file systems for migrated files. Each file system that might contain migrated files is searched for files that contain a bfid. Information about the state of each of these files is saved in the working directory. At the same time, dmaudit instructs the daemon to maintain a log of all changes that occur to migrated files while the search is in progress.
2. Copies the daemon database to the working directory. When the file system scans are complete, dmaudit asks the daemon to place a copy of the daemon database into the working directory together with the log of changes that occurred during the file system scan.
3. Brings the file system information up-to-date. The data collected during the file system scan is updated using the migrated file change log supplied by the daemon. When completed, this makes the file system information appear as if it were collected at the same instant that the daemon database copy was made. The file system and database data files are now synchronized with each other.
4. Sorts and merges the file system and database information. The file system information and database entries are each sorted into bfid order and are then merged together into a large file stored in the working directory.
5. Analyzes each bfid set for errors. All data available for each bfid set in the composite file is examined for inconsistencies. dmaudit compares the state in the user file's inode against any daemon database entries and flags any inconsistencies. It also computes and saves information on how to correct each error.
6. Issues a report. The report gives a brief summary of the kinds of errors encountered.

dmaudit retains in its working directory all snapshot information for each run. After the report has been issued, you have the option of using dmaudit interactively to examine each bfid set that contains errors.

Using dmaudit Interactively

The dmaudit command is intended primarily for interactive use; some steps, such as configuration, must be done interactively. dmaudit uses a series of menus to display information and to solicit input from you.

Environment Variables

For dmaudit to display its menus properly, it needs to know the dimensions of your screen. By default, it uses the values found in the environment variables LINES and COLUMNS. If the variables are not defined, dmaudit assumes a screen length of 24 lines and a width of 80 columns. You can either set these variables manually or use the resize command; see the resize(1) man page for further information. C shell users normally enter something similar to the following to set the screen size environment variables:

```
eval ‘resize -c‘
```

Sometimes dmaudit invokes an editor to request additional information from you. It uses the editor defined in the environment variable VISUAL (which is read first) or EDITOR; if neither is defined, it uses the vi(1) command by default. If you want to use an editor other than the default, set one of the two environment variables before starting dmaudit. For example, a C shell user who wants to use the emacs(1) editor could enter the following:

```
setenv VISUAL emacs
```

Page-Wait Mechanism

Occasionally dmaudit needs to display more information than can fit on your screen. In these cases, it uses a page-wait mechanism to retain the current screen of data until you are ready to continue. Your screen fills with data down to the last line, which appears as follows:

```
Enter <CR> to continue:
```

When you are done examining the current screen, press ENTER to view the next screen of data. If you are not interested in the remaining data, you may enter the first character of the next menu item and press ENTER to skip directly to processing the next menu item. There is no way to page back to a previous screen after you have pressed ENTER.

The page-wait feature is needed for users with terminals whose data would otherwise scroll off the screen before they could read it. If you are using a windowing system such as the X Window System environment, you can minimize the occurrence of page waits by running dmaudit in a window that contains as many lines as possible and by using the scroll bars available in the X Window System environment.

Setting the Initial Configuration

The first time that you run `dmaudit` you must configure it; this step must be done interactively. This consists of giving `dmaudit` a list of all the file systems that might contain migrated files, and specifying the location of a working directory where `dmaudit` can keep snapshot data for indefinite periods of time. `dmaudit` saves the configuration information that you enter in a file named *HOME_DIR*`/dmaudit_dir/checkpoint` (*HOME_DIR* is DMF home directory you specified in the DMF configuration file). The information is used in all subsequent invocations of the command.

Providing a Working Directory

To configure `dmaudit`, you must first either log in as `root` or use the `su`(1) command to become the super user. To execute the command interactively, enter the following:

```
dmaudit
```

The following display appears the first time that you invoke `dmaudit`:

```
This program must create a working directory named 'working_dir' to hold several very
large work files. The files may be needed in subsequent executions of this program,
so the directory should be placed in a file system that is not cleared frequently.

Please enter the full path name of an existing directory in which subdirectory
'working_dir' can be created (<CR> to quit):
```

Enter the name of a directory, such as `/tmp`. If the directory you specified does not exist or is otherwise invalid, `dmaudit` describes the nature of the problem and prompts for a new path name.

If you decide not to continue, you can exit the program at this time by pressing `ENTER`.

Specifying File Systems to Scan

After you have entered the directory name, `dmaudit` asks for the list of file systems it should search when looking for migrated files. It presents the following menu:

```
You must select the file systems to be scanned when this program searches for
  migrated files. By default the list contains all file systems
  currently mounted.

  IF THERE ARE ANY FILE SYSTEMS CONTAINING MIGRATED FILES THAT DO NOT APPEAR IN THIS
  LIST, EXIT THIS PROGRAM AND MOUNT THEM BEFORE CONTINUING!!!

  Failure to do so could mean the loss of the data in those files, because this
  program sometimes removes database entries if it cannot find matching migrated
  files for them. File systems that you are sure do not contain migrated files
  may be removed from this list to speed execution. Remember that using restore
  to load files dumped from a DMF-configured file system can result in migrated
  files in other file systems. If there is any question whether a file system
  contains migrated files, leave it in the list.

  Select:
     <view>     View the current file system scan list
     <edit>     Edit a fresh copy of the file system scan list
     <accept>   Accept the current file system scan list
     <quit>     Quit

  Please enter your selection:
```

Enter `view` to display the list of file systems that are currently mounted and that support data migration. (To support data migration, a file system must be mounted with the `dmi` option; see the *DMF Administrator's Guide for IRIX Systems* for information about mounting file systems).

When the list has been shown, `dmaudit` automatically redisplays the **File System Selection** menu. You should see something like the following:

```
/                               /core
/admin                          /mnt_tmp
/cloudy/ccn                     /tmp
/cloudy/mktg                    /usr
/cloudy/sdiv/comp               /usr/adm
/cloudy/sdiv/lib                /usr/dm
```

```
/cloudy/sdiv/net                /usr/spool
/cloudy/sdiv/qte                /usr/src
/cloudy/sdiv/xfiles             /usr/tmp


Select:
   <view>     View the current file system scan list
   <edit>     Edit a fresh copy of the file system scan list
   <accept>   Accept the current file system scan list
   <quit>     Quit

Please enter your selection:
```

If you have many file systems mounted, they may not all fit on the screen at the same time, in which case the last line on the screen instead looks like the following:

```
Enter <CR> to continue:
```

If that is the case, then when you are done examining the current screen, press ENTER to view the remainder of your file systems.

Normally you should choose the `accept` selection to run with all `dmi`–mounted file systems included in the audit. This is the proper method to use when you are unsure which file systems contain migrated files. `dmaudit` saves the path name, device name, and file system type of each file system in the list in its checkpoint file and uses this list in all future snapshots.

You might want to remove some of the file systems from the list if you know they will never contain migrated files because of the following reasons:

- File systems take time to scan. If you can remove file systems from the list, you will reduce `dmaudit` execution time.
- Some file systems change frequently. `dmaudit` expects to find each file system in the list mounted in the same location every time it is run, and does not allow you to take a snapshot or fix discrepancies if any file system has changed. You may want to remove file systems that are not always mounted.

The important thing is to make sure that any file system that might contain migrated files is in the list. `dmaudit` does not allow you to add file systems to the list that are

not currently `dmi`–mounted, so if any such file systems do not appear in the list, enter `quit` to exit the program and mount them before continuing.

To continue the example, assume that you do not want to scan any of the file systems shown in the previous screen starting with `/usr` because you know that they are large file systems and that no migrated files will ever reside in them. In addition, assume that no files will ever be migrated in `/`, that the `/mnt_tmp` file system is only used as a place to mount temporary file systems, and that `/tmp` is only used for temporary files that are not migrated. Assume the root file system should be removed from the list. (Sites that frequently switch root file systems when building new systems may want to remove the root file system from their list.)

If you want to exclude these file systems, enter `edit`. This puts your screen under control of the editor defined in the `VISUAL` or `EDITOR` environment variable, (`VISUAL` takes precedence); if both are undefined, it uses `vi`(1). You are shown a list of file systems in alphabetical order, one per line.

For the previous example, you would see the following:

```
/
/admin
/cloudy/ccn
/cloudy/mktg
/cloudy/sdiv/comp
/cloudy/sdiv/lib
/cloudy/sdiv/net
/cloudy/sdiv/qte
/cloudy/sdiv/xfiles
/core
/mnt_tmp
/tmp
/usr
/usr/adm
/usr/dm
/usr/spool
/usr/src
/usr/tmp
~
~
```

Using the editor, remove the lines associated with file systems that you do **not** want to include in the audit. Next, save your changes and exit the editor.

After you exit the editor, `dmaudit` reports any errors that it found in your input and then redisplays the file system selection menu. If errors are detected, `dmaudit` discards your entire edit session, and you must select `edit` again and reenter all your changes.

It is a good idea to select the `view` option one last time to verify that `dmaudit` accepted your selections.

In the current example, you would see the following:

```
/admin                          /cloudy/sdiv/net
/cloudy/ccn                     /cloudy/sdiv/qte
/cloudy/mktg                    /cloudy/sdiv/xfiles
/cloudy/sdiv/comp               /core
/cloudy/sdiv/lib


Select:
   <view>     View the current file system scan list
   <edit>     Edit a fresh copy of the file system scan list
   <accept>   Accept the current file system scan list
   <quit>     Quit

Please enter your selection:
```

The unwanted file systems no longer appear in the list.

Once you are satisfied with your file system selections, enter `accept` to continue. `dmaudit` responds with the following menu:

```
MAIN MENU
---------

Select:
   <snapshot>   Take a snapshot and report status of file systems and databases
   <config>     Examine or modify configuration information
   <quit>       Quit

Please enter your selection:
```

Configuration of `dmaudit` is now complete. At this point you may quit, enter `config` to examine certain other optional configuration parameters, (see Chapter 8, "Changing the Configuration", page 77), or enter `snapshot` to initiate a snapshot (as described in "The `snapshot` Option", page 28).

Detecting Discrepancies

The next step in using `dmaudit` is to generate a report that summarizes discrepancies between the file systems and the DMF daemon database. As was mentioned earlier, this is also known as *taking a snapshot*. The DMF daemon (`dmdaemon`(8)) must be running before you can attempt this step.

This chapter shows you how to generate the report both interactively and in batch mode. You should read the entire section before generating a report on your own machine so that you can choose which method best fits your needs.

Snapshot Resource Requirements

Taking a snapshot with `dmaudit` can require a large amount of both wall-clock time and CPU resources, depending on the size and number of files being scanned. During the snapshot, `dmaudit` must sort and merge several very large data files.

The first time you take a snapshot, you should do it during a time period in which you know `dmaudit` is able to execute continuously for several hours. Keep track of how long it takes to generate the report so that you can determine how much time to allot in future runs. If the program is stopped for any reason before the report generation is complete, the entire snapshot is discarded and you must start over.

Taking a Snapshot Interactively

This section describes the various options available to you when you take a `dmaudit` snapshot interactively.

If you just finished the initial configuration of `dmaudit`, you should already be positioned at the **Main** menu. If you are not currently running `dmaudit`, enter the `dmaudit` command.

If you are running `dmaudit` but are positioned at some other menu, enter `up` as many times as necessary until you arrive at the **Main** menu. Your screen should look like the following:

```
MAIN MENU
---------

Select:
   <snapshot>  Take a snapshot and report status of file systems and databases
   <config>    Examine or modify configuration information
   <quit>      Quit

Please enter your selection:
```

The snapshot Option

To generate a report of all discrepancies in your file systems and daemon database, enter `snapshot` from the **Main** menu. A display similar to the one shown in the following example eventually appears:

```
DATA MIGRATION CONFIGURATION
----------------------------
Data migration home directory:          /dmf/home
Data migration binaries directory:      /etc/dmf/dmbase/etc

Server name:                            daemon
Server home directory:                  /dmf/home/daemon
Server spool directory:                 /dmf/spool/daemon
Data migration daemon process ID:       68565

MSP name   MSP type     MSP home directory    MSP spool directory
--------   --------     ------------------    -------------------
red        dmatmsp      /dmf/home/red         /dmf/spool/red
dsk        dmdskmsp                           /dmf/spool/dsk
dlt7000m   dmatmsp      /dmf/home/dlt7000m    /dmf/spool/dlt7000m

LIST OF FILESYSTEMS SCANNED
---------------------------
/admin             xfs
/cloudy/ccn        xfs
/cloudy/mktg       xfs
/cloudy/sdiv/comp  xfs
/cloudy/sdiv/lib   xfs
/cloudy/sdiv/net   xfs
Enter <CR> to continue:
```

The beginning of the report shows your current DMF daemon and MSP/VG configuration, followed by the list of file systems that were searched for migrated files. The type of each file system is shown next to its path name. If, as in this example, the list is continued on the next screen, press ENTER to view the remainder of the report:

```
MAIN MENU
---------

Select:
   <inspect>    Inspect and correct file system and database errors
   <report>     Reprint status report for the current snapshot
   <verifymsp>  Check the dmatmsp tape msp and dmatls tape library server
                databases against the daemon databases
   <snapshot>   Take a snapshot and report status of file systems and databases
   <free>       Release all file space used by the current snapshot
   <config>     Examine or modify configuration information
   <quit>       Quit

Please enter your selection:
```

The second screen shows the remainder of the file system list together with the database error report. In the example, no errors were detected in either the file systems or the daemon database.

After the report, `dmaudit` automatically returns to the **Main** menu. The `inspect`, `report`, and `free` options are described in the following sections.

The `inspect` Option

The `inspect` option is normally used only in cases in which discrepancies have been detected. It allows you to examine each error in detail and to correct those errors. Even if there are no discrepancies, however, `inspect` can be used to examine individual bfid sets in the daemon database.

The `report` Option

The `report` option allows you to reprint the report from your most recent snapshot. Because all information contained in the report is determined during the snapshot phase, the `report` option can reissue the report instantaneously. The report looks the same as the one shown in "The `snapshot` Option", page 28.

This option is useful if you took a snapshot earlier and want to view its report again. It is also handy if you normally take snapshots in batch mode. If a batch run indicates that there are errors, you can then execute `dmaudit` interactively and read the report directly.

The `verifymsp` Option

The `verifymsp` option allows you to have `dmaudit` run the `dmatvfy`(8) command using the data captured during the snapshot and adding the information to the `dmaudit` report. A display similar to the following eventually appears:

```
Please enter your selection: verifymsp
10/10/97 10:06:42 : Extracting dmf records for msps: silo.
10/10/97 10:06:43 : Verifying msp "silo"...

MSP silo ERROR REPORT
---------------------

  Bitfile ID 344789f2000000000000006c - missing from daemon db.
  Bitfile ID 344789f200000000000000d5 - missing from daemon db.
  Bitfile ID 344789f2000000000000028e - missing from daemon db.
3 Errors found in CAT database.
0 Errors found in VOL database.

MAIN MENU
---------
```

You can view the output by using the `report` option, which is described in "The `report` Option", page 29.

The `free` Option

After you take a snapshot, a potentially large amount of disk space remains allocated to files in the working directory to contain all the information `dmaudit` needs to generate its report. This is done so that you can examine the snapshot at your leisure. The disk space normally remains reserved until you take your next snapshot.

If no errors were detected in the snapshot, however, you might want to enter the `free` option to release that disk space so that it can be used for other purposes until

the next time you take a snapshot. The `free` option discards all snapshot files while retaining all configuration information.

After the snapshot files have been removed, the **Main** menu once again looks like the following:

```
MAIN MENU
---------

Select:
   <snapshot>   Take a snapshot and report status of file systems and databases
   <config>     Examine or modify configuration information
   <quit>       Quit

Please enter your selection:
```

You do not have to manually select `free` each time before taking a new snapshot. The `snapshot` option automatically releases all space from a previous snapshot before taking a new one.

Example of a Report with Discrepancies

If discrepancies are detected between the file systems and the daemon database, `dmaudit` attempts to summarize them in the last section of its report.

The following is an excerpt from a report that detected several kinds of errors:

```
DAEMON DATABASE ERROR REPORT
----------------------------

There are 14 bitfile IDs in use by more than one file that cannot be
corrected without additional information from you.

There are 3 bitfile IDs in use by more than one file that can be
automatically corrected.

There are 2 user files whose data cannot be recovered.

There are 231 user files that have correctable errors.

There are 18 bitfile IDs in the daemon database for which no user files
```

```
can be found.

There are 43 bitfile IDs whose errors cannot be corrected until a new
snapshot is taken.

There are 12 bitfile IDs with files that cannot be recalled or migrated.

There are 9 bitfile IDs with files that are internally inconsistent.
```

`dmaudit` summarizes the errors that it finds into eight general classes (described in Table 7-1, page 35). `dmaudit` allows you to examine each of the error classes separately.

Taking a Snapshot in Batch Mode

To avoid tying up your terminal for long periods of time, `dmaudit` also allows you to specify the `snapshot` option in batch mode. To take a snapshot in batch mode, specify `snapshot` as a parameter on the command line. For example, a Bourne shell user might enter the following:

```
nohup dmaudit snapshot >rpt 2>err &
```

(The `nohup`(1) command allows `dmaudit` to continue processing even if you log out of the system.) When the snapshot completes, the report is written to standard output, in this case to the file `rpt`. If `dmaudit` encounters fatal errors, it issues messages to standard error (in the example, this is the file `err`).

The exit status from a snapshot run in batch mode can be used in shell scripts to determine whether errors were detected, as follows:

- An exit status of 0 indicates that `dmaudit` completed the snapshot and that no file system or database errors were detected.
- An exit status of 1 indicates that file system or database errors were discovered by `dmaudit`.
- Any other exit status indicates that `dmaudit` aborted with a fatal error.

The `report` and `free` options may also be used in batch mode. The following is an example `cron`(blank) script that could be used to take snapshots on a periodic basis:

```
#!/bin/sh
#       script to take a snapshot and report errors by mail
#

TMPDIR=/TMP
MAILLIST="root"

dmaudit snapshot 2>$TMPDIR/errmsg 1>/dev/null
STATUS=$?

if [ $STATUS -eq 0 ]
then
        dmaudit free        # remove unneeded files
        exit 0
fi

if [ $STATUS -eq 1 ]
then
        dmaudit report | mailx -s "dmaudit error report" $MAILLIST
else
        cat $TMPDIR/errmsg | mailx -s "dmaudit failed with this msg" $MAILLIST
fi
exit 1
```

If the snapshot completes successfully with no errors detected, the script uses the `free` option to release all working directory space. If errors are detected or if `dmaudit` fails, mail is instead sent to the administrator.

Examining and Correcting Discrepancies

This chapter provides an overview of how to use `dmaudit` to examine and correct errors and a walk-through for each menu, using as an example some of the inconsistencies introduced when a file system that uses DMF is restored. It also includes additional information on each of the error classes.

Overview of the Correction Process

Correcting errors with `dmaudit` uses a two-step process. The first step is to interactively select which errors you want to correct; the second step is to correct those errors in either interactive or batch mode.

Errors are divided into eight major classes. Each of these classes has its own correction menu, although the contents of these menus are largely the same. The error classes are described in Table 7-1:

Table 7-1 `dmaudit` error classes

Class	Description
1	More than one file has the same bfid, and `dmaudit` cannot determine by itself which database entries go with which migrated file. User data may or may not be lost.
2	More than one file has the same bfid, but `dmaudit` has enough information to determine which database entries go with which migrated file. User data may or may not be lost. This class implies database errors.
3	Migrated files for which data cannot be found.
4	Files with minor errors that can be repaired without loss of user data. These are usually caused by events such as the restoration of a file system. No user data is at risk in a recoverable error, so they are usually corrected with detailed examination.
5	Active daemon database entries for which no migrated files can be found. No user data is affected by this type of error. The only negative effect of errors in this class is the space wasted by the unused file copies on the migration media. This type of error usually occurs when there is user activity while the daemon is not running.

Class	Description
6	Bfid sets with errors which indicate that file states or daemon database entries were actively changing while the snapshot was in progress, or that have changed since the snapshot completed. Their status cannot be evaluated and therefore they cannot be fixed without taking another snapshot. You can, however, look at the most recent state of the bfid sets.
7	Bfid sets with files that are internally inconsistent.
8	Bfids found in file systems not configured for DMF and for which no database entries were found. User data has been lost.

Selecting Which Errors to Correct

To select errors to be corrected, enter the number of the error class you want to correct, which displays the appropriate error-class menu. Each error class must be corrected separately. Within each error class menu, choose which errors you want to correct by accepting the modifications `dmaudit` needs to correct them. Each error class except class 6 has an `accept` option for this purpose. If you change your mind, you can cancel your acceptance by using the `cancel` option.

Correcting the Errors

When you are satisfied with the list of error corrections you have accepted, the next step is to return to the **Main** menu and apply your changes using the apply option. There are two ways to do this:

- If the corrective actions presented by `dmaudit` imply that very little tape and disk activity is required, you may want to fix the errors immediately. In that case, select `apply` on the **Main** menu and wait. When the **Main** menu next appears, all errors are corrected.
- If the corrective actions imply that a lot of tape and disk activity is required, you may want to wait until off-peak hours to actually correct the errors. You can do it interactively, or you can do it in batch mode by specifying `apply` as a parameter on the `dmaudit` command line.

The inspect Menu

To introduce the most common correction options available within dmaudit, the next few sections show an example in which the xfsdump(1m) and xfsrestore(1m)commands were used to cause errors to appear. Restoring a file system can cause errors because the file system is being returned to a previous state, but the daemon database remains in its current state.

An attempt was made in these examples to produce as many different kinds of errors as possible. If you have to restore a file system, your next dmaudit report will probably only contain a subset of the errors shown here.

Errors were produced first by using a dump command to dump a DMF file system. User activity was then simulated by migrating, recalling, and removing files in the file system. The file system was then restored using a restore command, and dmaudit was run to produce an error report.

For purposes of the example, it is assumed that you have already taken a snapshot and that errors were detected. The error summary portion of the dmaudit report would show something like the following:

```
DAEMON DATABASE ERROR REPORT
----------------------------
There are 8 user files that have correctable errors.

There are 2 bfids in the daemon database for which no user files can be found.
```

Assume you decide to correct the errors. You must always start from the **Main** menu. If you are not currently running dmaudit, issue the dmaudit command.

If you are running dmaudit but are in some other menu, enter up as many times as necessary until you arrive at the **Main** menu.

Your screen will look like the following:

```
MAIN MENU
---------

Select:
   <inspect>    Inspect and correct file system and database errors
   <report>     Reprint status report for the current snapshot
   <verifymsp>  Check the dmatmsp tape msp and dmatls tape library server
                databases against the daemon databases
   <snapshot>   Take a snapshot and report status of file systems and databases
```

```
   <free>       Release all file space used by the current snapshot
   <config>     Examine or modify configuration information
   <quit>       Quit

Please enter your selection:
```

To examine and correct errors, enter `inspect`, which causes the following menu to appear:

```
INSPECT FILE SYSTEM AND DATABASE ERRORS
---------------------------------------

Errors are divided into several major classes. Only those classes that pertain
to this snapshot are displayed.

Select:
   <4>        Examine files that have correctable errors
   <5>        Examine bfids in the daemon database for which no user files
              can be found
   <bfid>     Examine all files and database entries for any bfid you specify
   <search>   Scan file systems for names of all files with errors (very slow)
   <up>       Return to the previous menu

Please enter your selection:
```

There is a direct correspondence between the two error classes shown in this menu and those mentioned in the error report. You must examine and correct each of these classes separately. To examine a particular class, enter its class number at the prompt. See "Fixing Bfid Sets for Which No User Files Exist", page 43.

The following sections describe the `bfid` and `search` options; the `up` option returns you to the **Main** menu. The error-class options (`4` and `5`) are described in "Fixing Bfid Sets for Which No User Files Exist", page 43, and "Fixing Files with Correctable Errors", page 48.

The `bfid` Option

The `bfid` option is used to examine everything that is known about a particular bfid set. You can enter a particular bfid set value or select the default value displayed at the prompt:

```
Select:
   <bfid>     Examine all files and database entries for any bitfile ID you
              specify
   <search>   Scan file systems for names of all files with errors (very slow)
   <up>       Return to the previous menu
Please enter your selection: bfid
(34745a2b0000000000000444): <enter an alternate bfid here or return>

BITFILE ID = 34745a2b0000000000000444


ENTRIES IN USE:

01. dual-state user file - dev 50331653 (daemon), fhandle
    0100000000000001888c5f6086f39b65d000e0000000000000000000000401f8c, uid 285,
    size 175538
02. daemon MSP/VG <ftp> database entry - not soft deleted, size 175538, key
    <abc/34745a2b0000000000000444>


  Select:
     <next>   Examine the next bfid in the snapshot
     <prev>   Examine the previous bfid in the snapshot
     <mode>   Switch to full display
     <dump>   Append all information about this bfid to a text file in a format
              suitable for machine processing
     <up>     Return to the previous menu

  Please enter your selection:
```

This is called an abbreviated display because it shows only some of the information known about this bfid set. It shows only those pieces of information most commonly examined and has the advantage of taking up less screen space. There are two objects associated with bfid `34745a2b0000000000000444`: a user file and an MSP daemon database entry. The user file has a state of dual-state, a device number of `50331653`, a user identification (UID) of `285`, and a size in bytes of `175538`. The database entry is for an MSP or VG named `ftp`, and the key the MSP/VG uses to retrieve the file is `abc/34745a2b0000000000000444`. Furthermore, the soft-delete field is `0`, indicating that the entry is still active.

You can see a more detailed description of the bfid set by selecting the `mode` option, which redisplays the bfid set in full-display mode. This display shows all information

known by `dmaudit` about the bfid set. If you enter `mode` for this bfid set, you see the following (the `mode` option is actually a toggle; each time you select it, the display switches to the display type not currently shown):

```
BITFILE ID = 34745a2b0000000000000444

ENTRIES IN USE:

01. dual-state user file - dev=50331653 (daemon)
    fhandle=0100000000000001888c5f6086f39b65d000e0000000000000000000000401f8c
    size=175538 uid=285 nlinks=1
02. daemon MSP/VG database entry - dev=50331653 ino=4202380 size=175538 uid=285
    otime=Nov 20 13:08:36 1997 utime=Nov 20 13:08:36 1997 ctime=Nov 20 13:08:36
    1997 dtime=0 name=dmf_tst.00061 msp/vg=ftp key=<abc/34745a2b0000000000000444>


Select:
   <next>   Examine the next bfid in the snapshot
   <mode>   Switch to abbreviated display
   <dump>   Append all information about this bfid to a text file in a format
            suitable for machine processing
   <up>     Return to the previous menu

Please enter your selection:
```

Several new pieces of information appear. The number of hard links for the user file are now shown, as is every field in the daemon's MSP/VG database entry. In both cases, information that does not fit on the current line is continued on subsequent lines without line numbers.

To conserve screen space, the names of the fields in a database entry in this display are somewhat shorter than the names actually used by the daemon in its `dbrec` structure (see "DMF Daemon Database Contents", page 8, for a description of the structure). Table 7-2 shows the correspondence between names for the fields in the full display and in the daemon `dbrec` structure.

Table 7-2 Database field descriptions

Display name	dbrec structure	Description
dev	origdv	Specifies the device number of the file
ino	origino	Specifies the inode number of the file
size	origsz	Specifies the file size in bytes
uid	userid	Specifies the user ID of the file owner
otime	otime	Specifies the origination time of the entry
utime	utime	Specifies the last update time of the entry
ctime	ctime	Specifies the last check time of the entry
dtime	delflag	Specifies the soft-delete time of the entry
name	ofilenm	Specifies the base name (if known) of the file
msp	proc	Specifies the MSP or VG name
key	path	Specifies the MSP/VG key or path name

Several other options are available on the `bfid` display. Because `dmaudit` maintains its data in bfid-set order, the `next` and `prev` options allow you to examine bfid sets in the database adjacent to the current one.

Entering `next` shows you the next higher bfid set in use. If you enter `prev`, you would see consecutively lower bfid sets. This can sometimes be useful if you want to see the state of bfid sets whose files were migrated at about the same time as the current file.

The `dump` option is not commonly used. It allows you to dump all information contained in the full display into a file in machine-readable form. If you select the `dump` option, it prompts you for the name of a file:

```
File to which you want the dump appended?
```

Enter the path name of a file to which you want the information to go. Either full or relative path names can be used. If the file does not already exist, `dmaudit` creates it. If it does exist, it appends the information to the end of the file. If you instead press

ENTER, dmaudit assumes that you have changed your mind and returns to the **bfid** menu.

The format of the dump information is complex; for a description, see Appendix A, "dump Option Output", page 85.

When the dump completes, dmaudit redisplays the **bfid** menu.

From this display, enter up to return to the **Inspect** menu.

The search Option

The search option only appears on the **Inspect** menu when there are bfid sets with errors that have user files associated with them. For example, if there were only class-5 errors in the following menu, search would not appear because no user files would be involved:

```
INSPECT FILE SYSTEM AND DATABASE ERRORS
---------------------------------------
Errors are divided into several major classes. Only those classes that pertain
to this snapshot are displayed.

Select:
   <4>        Examine files that have correctable errors
   <5>        Examine bfids in the daemon database for which no user files
              can be found
   <bfid>     Examine all files and database entries for any bfid you specify
   <search>   Scan file systems for names of all files with errors (very slow)
   <up>       Return to the previous menu

Please enter your selection:
```

The purpose of search is to find the path names for each user file whose bfid set has errors associated with it. dmaudit does this by recursively scanning through the directories of each file system that contains bfid sets with errors.

Note: This process can take a long time, perhaps hours on very large file systems. You can interrupt the process by pressing Control-c, but then you must start the search again from the beginning. It is better to examine the errors first to see if you really need to know the names of the files. If you do later decide you need the names, you can return here and enter search.

A search from the **Inspect** menu finds the names of all user files whose bfid sets have errors for all applicable classes. The individual error class menus also have a `search` option, but their search is restricted to those user files that are in their error class. If you need names for multiple classes, it is usually more efficient to get all the names at once from the **Inspect** menu. If you enter `search`, `dmaudit` pauses for an indefinite period of time as the file systems are scanned. When the **Inspect** menu next reappears, all the names have been collected, and the `search` option disappears from the menu.

Fixing Bfid Sets for Which No User Files Exist

To continue with the example, you are now ready to examine each of the error classes to see what kinds of errors occurred. Assume that you entered `5` to look at those bfid sets whose database entries are not soft-deleted and for which no migrated user files can be found.

The following menu appears:

```
DAEMON DATABASE BITFILE IDS FOR
WHICH NO FILES CAN BE FOUND
---------------------------

There are 64 bitfile IDs in this error class.


The following errors were detected in these bitfile IDs.

01.  There are 64 MSP/VG daemon database entries that should be soft deleted.


The following actions will be taken to correct these errors.

02.  64 MSP/VG daemon database entry soft-delete flags will be set.


Select:
   <accept>    Accept the recommended actions.  Once accepted, these actions
               will be taken when you select 'apply' on the main menu.
   <examine>   Examine or modify the actions to be taken for individual files.
               You may enter the line number of any one of the above subsets of
               files; the default is to examine all the files.
```

```
   <bfid>      Examine all files and database entries for any bitfile ID you
               specify.
   <dump>      Append all information available on each of the bitfile IDs
               listed above to a text file in a format suitable for machine
               processing.
   <up>        Return to the previous menu.

Please enter your selection:
```

For each of the error class menus, `dmaudit` always shows the following:

- Number of bfid sets with errors in this error class
- Summaries of all the individual errors found in the bfid sets in this class
- What actions `dmaudit` needs to take to fix the individual errors

The messages displayed in these menus depend upon what errors were found on your system. There are five options available on this display. `up` returns you to the **Inspect** menu, and the `bfid` option functions as described "The `bfid` Option", page 38.

The `accept` Option

The most important option on an **Error Class** menu is `accept`. By entering `accept`, you tell `dmaudit` that you are willing to let it take the listed actions to correct the discrepancies in these bfid sets. In this particular case, it means that `dmaudit` will soft-delete 64 daemon database entries. Action is not taken until you enter `apply` at the **Main** menu.

After you have entered `accept`, the format of the menu changes slightly. The `accept` option disappears because it is no longer needed, and in its place you will instead see the following:

```
<cancel>    Cancel your acceptance of the above recommendations.
```

The `cancel` option allows you to change your mind. If you enter `cancel`, it disappears and `accept` reappears, returning the screen to its original state. You can change your mind as many times as you like up until the point at which you enter `apply` on the **Main** menu. At that time, `dmaudit` fixes all bfid sets whose `accept` flag is currently set.

Most of the time, you do not need to examine these bfid sets in more detail and can type `accept` followed by `up` to move on to other error classes. If you do want to go into additional detail, there are additional options available to help you.

The `examine` Option

The `examine` option allows you to look at individual bfid sets within this error class. The format of its display is very similar to the `bfid` display. If you enter `examine`, you are prompted for a line number:

```
Line number (<CR> for all)?
```

Each of the errors and each of the actions reported on the **Error Class** menu have line numbers on them. If you enter a line number at the `examine` prompt, it examines only information about bfid sets that have the specified error or action associated with them. If you want to examine all bfid sets in this error class, press `ENTER`. Because there are only two bfid sets in this class, assume that you press `ENTER`.

You then see the following:

```
BITFILE ID = 34745a2b000000000000c6c1

ENTRIES IN USE:

01. no user file -
02. daemon MSP/VG database entry - dev=50331653 ino=3445388 size=692637 uid=11414
    otime=Nov 26 06:04:20 1997 utime=Nov 26 06:04:20 1997 ctime=Nov 26 06:04:20
    1997 dtime=0 name=/NONAME msp/vg=ftp key=<dinesh/34745a2b000000000000c6c1>
       This database entry should be soft deleted.
       The soft delete flag will be set in this entry.


Select:
   <next>     Examine the next bfid in the list
   <last>     Examine the last bfid in the list
   <mode>     Switch to abbreviated display
   <edit>     Edit this bfid
   <dump>     Append all information about this bfid to a text file in a format
              suitable for machine processing
   <accept>   Accept the changes to this bitfile ID
```

```
   <up>       Return to the previous menu
Enter <CR> to continue:

Please enter your selection:
```

This display shows an example of a bfid set with an error. Each object in a bfid set that has an error associated with it also has error and action descriptions following it on the screen. In this case, line 2 is followed by the error description and action required to correct the database entry given. It explains that no user file was found with this bfid.

Table 7-3 describes the other options on this menu:

Table 7-3 `examine` menu options

Option	Description
`next`	Displays the next bfid set within the list of bfid sets you chose to look at. After that bfid is displayed, the `prev` option appears, allowing you to see previous bfid sets.
`last`	Goes directly to the last bfid set in the list instead of having to enter `next` multiple times. The `first` option, when it appears, allows you to return to the first bfid set in the list.
`mode`	Toggles back and forth between abbreviated and full versions of the display.
`edit`	Allows you to make changes to the bfid set to force `dmaudit` to correct it in some other way. This option is only needed in several very specific cases; it should normally not be used. For more information, see "The `edit` Option", page 64.
`dump`	Dumps all information about the bfid set to a file in machine-readable form. You are prompted for the name of a file to which data should be appended.
`accept`	Accepts the actions required to correct the bfid set without affecting other bfid sets in the same error class. This option should seldom be needed; normally you will accept all errors in the class at the same time.

The dump Option

The dump option is similar to the one on the bfid and examine displays, but this dump outputs all information known about any of the bfid sets in this error class rather than just a single bfid set.

If you enter dump, you are prompted for a line number:

```
Line number (<CR> for all)?
```

Enter a line number if you only want to examine bfid sets that have a specified error or action associated with them. If you want to examine all bfid sets in this error class, press ENTER. The next prompt asks for a file name:

```
File to which you want the dump appended?
```

Enter the full or relative path name of a file, or press ENTER to cancel.

Returning to the Inspect Menu

Enter accept to accept the changes to these two bfid sets, and then enter up again to move on to the next error class.

You then see the following:

```
INSPECT FILE SYSTEM AND DATABASE ERRORS
---------------------------------------
Errors are divided into several major classes. Only those classes that pertain
to this snapshot are displayed.

Select:
   <4>        Examine files that have correctable errors
   <5>        Examine bfids in the daemon database for which no user files
              can be found
   <bfid>     Examine all files and database entries for any bfid you specify
   <search>   Scan file systems for names of all files with errors (very slow)
   <up>       Return to the previous menu

Please enter your selection:
```

Fixing Files with Correctable Errors

The next step in the example is to look at correctable errors. The correctable error class introduces several new options that were not available in the previous **Error Class** menu.

If you enter 4, you see the following:

```
FILES WHICH HAVE CORRECTABLE ERRORS
-----------------------------------
There are 9 bitfile IDs in this error class.


The following errors were detected in these bitfile IDs.

01.  There are 9 MSP/VG daemon database entries with an unknown MSP or VG name.
02.  There are 9 migrated user files that have no MSP/VG daemon database entries.


The following actions will be taken to correct these errors.

03.  9 MSP/VG daemon database entries will be removed.
04.  9 migrated user files will have incomplete MSP/VG daemon database entries
     created for their default MSPs/VGs.
05.  9 migrated user files will be remigrated to all MSPs/VGs for which they have
     incomplete database entries.


Select:
   <accept>     Accept the recommended actions.  Once accepted, these actions
                will be taken when you select 'apply' on the main menu.
   <examine>    Examine or modify the actions to be taken for individual files.
                You may enter the line number of any one of the above subsets
                of files; the default is to examine all the files.
   <bfid>       Examine all files and database entries for any bitfile ID you
                specify.
   <opt_part>   Recompute all corrective actions assuming that daemon database
                entries with errors can be removed as long as one good copy of
                the file exists elsewhere.
   <search>     Scan file systems for the names of all files listed above (very
                slow).
   <dump>       Append all information available on each of the bitfile IDs
```

```
                listed above to a text file in a format suitable for machine
                processing.
   <nlist>      Append all file names for all the files listed above to a text
                file after first scanning the file systems for their names
                (very slow).
   <up>         Return to the previous menu.

Please enter your selection:
```

While this display is certainly more complex than the display for the previous error class, its format is essentially the same. The first section on the screen shows those errors that were detected in the bfid sets in this class, and the second section shows what actions `dmaudit` will take to correct those errors.

Recoverable errors tend to be more complex than errors you have seen previously. The following is an example of what a bfid set with recoverable errors looks like:

```
BITFILE ID = 34745a2b000000000000954b


ENTRIES IN USE:

01. dual-state user file - dev 50331652 (daemon), fhandle
    0100000000000001888c5f6086f39b65c000e0000000000200000000007040f6, uid
    15948, size 3863854
       No MSP/VG daemon database entries exist for this user file.
       Incomplete MSP/VG daemon database entries will be created for the default
       MSPs/VGs associated with this user file.
       This file will be remigrated to those MSPs/VGs for which it has incomplete
       database entries.
```

The online file is the only available copy of the user file's data. Even though the file is in a dual state, no daemon database entries exist for the file, and therefore no backup copies are available. In order to repair this bfid set, `dmaudit` has to remigrate the user file to all the MSPs and VGs that have the incomplete entry. Several new options that are available on the **Recoverable Error Class** menu are described in the following sections.

The opt_full and opt_part Options

The following bfid set has no MSP/VG database entry, but the dual-state user file indicates that the online copy of the file is valid. The display is as follows:

```
BITFILE ID = 34745a2b000000000000954b

ENTRIES IN USE:

01. dual-state user file - dev 50331652 (daemon), fhandle
    010000000000001888c5f6086f39b65c000e0000000000200000000007040f6, uid
    15948, size 3863854
       No MSP/VG daemon database entries exist for this user file.
       Incomplete MSP/VG daemon database entries will be created for the default
       MSPs/VGs associated with this user file.
       This file will be remigrated to those MSPs/VGs for which it has incomplete
       database entries.
```

There are two ways to fix this problem:

- Recreate the MSP and VG database entries and remigrate the file to all MSPs and VGs specified in the current configuration.
- Remove the bitfile ID, making the file a regular online file.

As released, `dmaudit` uses the first approach. This can be modified in two ways. If you enter `config` from the **Main** menu, one of the displayed options allows you to specify which method `dmaudit` uses as its default. `dmaudit` remembers the new default from run to run. For more information, see "The `invalid` Option", page 80. You can also see the results of using the second approach by entering the `opt_part` option on the current display. When selected, `dmaudit` redisplays the **Error Class** menu, replacing the following action:

```
04.  9 migrated user files will have incomplete MSP/VG daemon database entries
     created for their default MSPs/VGs.
05.  9 migrated user files will be remigrated to all MSPs/VGs for which they have
     incomplete database entries.
```

The other actions on the display remain the same (except that some lines have different line numbers). If you then enter `examine` to look at the same bfid set previously shown, you see the same change in actions.

After you have entered `opt_part`, it disappears from the menu and is replaced by the `opt_full` option, which toggles back to the first recovery approach. You can

toggle back and forth as much as you like. When you enter `accept`, `dmaudit` uses the recovery method currently in force. If you toggle the method after entering `accept`, `dmaudit` performs a `cancel` on those bfid sets affected by the toggle. You then must accept the changes again.

The `search` Option

The `search` option allows you to find the names of all user files associated with bfid sets in this error class. The names of the files are found by recursively scanning the directories of the file systems in which the files reside. This can take a long time, so select this option only if you truly need to know the file names. If all the names of all the user files in this error class are already known, the `search` option does not appear.

After the names have been found, they appear in several places, such as on the `examine` display. The following screen shows the beginning of the display for a bfid used in the previous example:

```
BITFILE ID = 34745a2b000000000000954b

ENTRIES IN USE:

01. dual-state user file - dev=50331652 (daemon)
    fhandle=0100000000000001888c5f6086f39b65c000e0000000000020000000007040f6
    size=3863854 uid=15948 nlinks=0
    /dmi1/stim/dir1/dmf_tst.04266
       No MSP/VG daemon database entries exist for this user file.
       The bitfile ID will be removed from this user file.
```

The bfid (bitfile ID) will be removed from this user file. The path names are also output when using the `dump` option, and when using `nlist` (described in the next section). If you need to know the names of user files in more than one error class, it is more efficient to use the `search` option that appears on the **Inspect** menu instead of using the `search` option repeatedly in each of the **Error Class** menus.

The `nlist` Option

The `nlist` option is useful when you want to create a list of the names of user files in a particular error class. `nlist` saves the list in a file whose name you specify. Normally, the `nlist` option looks like the following:

```
<nlist>      Append all file names for all the files listed above to a text
             file after first scanning the file systems for their names
             (very slow).
```

If you have previously used the `search` option to retrieve the names of the files in this error class, the `nlist` option will instead look like the following:

```
<nlist>      Append all file names for all the files listed above to a text
             file.
```

You are prompted for a line number:

```
Line number (<CR> for all)?
```

Enter a line number if you only want to examine bfid sets that have a specified error or action associated with them. If you want to examine all bfid sets in this error class, press `ENTER`. The next prompt asks for a file name:

Enter the full or relative path name of a file, or press `ENTER` to cancel.

You can either enter one of the line numbers appearing on the **Error Class** menu to create a list containing a subset of the files in this error class, or you can press `ENTER` to get the names of all user files in this error class. You are then prompted for the name of a file to which the file names should be appended:

```
File to which you want the file names appended?
```

Enter the full or relative path name of a file, or press `ENTER` to cancel the operation. The following display shows a sample file created using the `nlist` option:

```
/dmf1/example/cmp_to_inc
/dmf1/example/back\e134slash
/dmf1/example/pipe\e174char
/dmf1/example/link1
/dmf1/example/link2
/dmf1/example/link3
/dmf1/example/inc_to_cmp
/dmf1/example/lost_inc_pfile
/dmf1/example/rmv_inc_pfile
```

The example shows several things that you must be aware of when you use the output from `nlist`, especially when using it as input to some other program:

- If a file has multiple path names, `nlist` lists each of the path names on consecutive lines. This is done for hard links only; symbolic links to a file are not shown. In the previous example, the path names for `link1`, `link2`, and `link3` are all hard links to the same file.
- Because nothing prevents users from creating file names containing line feeds, vertical tabs, and other unprintable characters, `nlist` converts such characters into printable form before adding the path name to the list. For each backslash, vertical bar, or unprintable character in a file name, nlist replaces that character with the string, where the *000* field is a 3-digit octal number that is the ASCII value for that character. In the previous example, file `back\slash` contains a backslash and file `pipe|char` contains a vertical bar.

Returning to the Inspect Menu

Assuming that you are done looking at this error class, you would enter `accept` to accept the changes to the bfid sets, and then enter `up` again to return to the **Inspect** menu. You then see the following:

```
INSPECT FILE SYSTEM AND DATABASE ERRORS
---------------------------------------

Errors are divided into several major classes. Only those classes that pertain
to this snapshot are displayed.

Select:
   <4>        Examine files that have correctable errors
   <5>        Examine bfids in the daemon database for which no user files
```

```
               can be found
   <bfid>      Examine all files and database entries for any bfid you specify
   <search>    Scan file systems for names of all files with errors (very slow)
   <up>        Return to the previous menu

Please enter your selection:
```

Because you have now accepted the corrections in both error classes, enter `up` to return to the **Main** menu where you apply the changes that you have accepted. For more information, see "The `apply` Option", page 74.

An `xfsdump`(1m) and `xfsrestore`(1m) can only produce class 4 and class 5 errors. The next few sections show examples of other error classes you might see.

Cleaning up Files with Unrecoverable Errors

A bfid set contains an unrecoverable error if a user inode exists but no valid copy of the user's data can be found. You should never see a bfid set with unrecoverable errors in normal operation. If one does occur, it means that either an administrative error was made, a file system error occurred, or that DMF has failed in some way. You should always examine errors in this class to determine why they happened, in order to prevent possible recurrences in the future.

To give an example of an unrecoverable error, a test file is migrated offline to alternate media. The `dmdadm`(8) command is then used to manually remove the database entry for the associated bfid to induce the error. When `dmaudit` is next run, it reports an unrecoverable error. The resulting **Inspect** menu looks like the following:

```
INSPECT FILE SYSTEM AND DATABASE ERRORS
---------------------------------------

Errors are divided into several major classes. Only those classes that pertain
to this snapshot are displayed.

Select:
   <3>         Examine files whose data cannot be recovered
   <bfid>      Examine all files and database entries for any bfid you specify
   <search>    Scan file systems for names of all files with errors (very slow)
   <up>        Return to the previous menu

Please enter your selection:
```

Entering 3 advances you to the following unrecoverable error class menu:

```
FILES WHOSE DATA CANNOT BE RECOVERED
------------------------------------

There are 1 bitfile IDs in this error class.


The following errors were detected in these bitfile IDs.

01.  There are 1 migrated user files for which no good data copies can be
     found.


The following actions will be taken to correct these errors.

(No actions will be taken.)


Select:
   <accept>    Accept the recommended actions.  Once accepted, these actions
               will be taken when you select 'apply' on the main menu.
   <examine>   Examine or modify the actions to be taken for individual files.
               You may enter the line number of any one of the above subsets of
               files; the default is to examine all the files.
   <bfid>      Examine all files and database entries for any bitfile ID you
               specify.
   <search>    Scan file systems for the names of all files listed above (very
               slow).
   <dump>      Append all information available on each of the bitfile IDs
               listed above to a text file in a format suitable for machine
               processing.
   <nlist>     Append all file names for all the files listed above to a text
               file after first scanning the file systems for their names (very
               slow).
   <remove>    After first scanning the file systems for their names, remove
               files that cannot be recovered (very slow).
   <up>        Return to the previous menu.

Please enter your selection:
```

Removing the lost user file is not one of the actions listed in lines 4 and 5. To remove lost files, you must use the `remove` option as described in "The `remove` Option".

The following is an excerpt from the abbreviated display for the bfid set with an unrecoverable error:

```
BITFILE ID = 20000000000000001


ENTRIES IN USE:

01. off-line user file - dev 33554464 (daemon), fhandle
    010000000000001885beb4c83ff373e0000e0000000004200000000000000043, uid 0,
    size 4474272
       No good data copy can be found for this user file.


Select:
   <mode>     Switch to full display
   <edit>     Edit this bfid
   <dump>     Append all information about this bfid to a text file in a format
              suitable for machine processing
   <accept>   Accept the changes to this bitfile ID
   <up>       Return to the previous menu

Please enter your selection:
```

The `remove` Option

The `dmaudit` command does not automatically remove unrecoverable user files when you enter `accept` and `apply`. To actually delete the files, you must select the `remove` option. If you have not used the `search` option, the remove option looks like the following:

```
  <remove>    After first scanning the file systems for their names, remove
              files that cannot be recovered (very slow).
```

If you have already used `search` to find the file's path names, the `remove` will instead look like the following:

```
  <remove>    Remove files that cannot be recovered.
```

After you enter the `remove` option, the error class display shows the additional action that the files will be removed:

```
FILES WHOSE DATA CANNOT BE RECOVERED
------------------------------------
There are 1 bitfile IDs in this error class.


The following errors were detected in these bitfile IDs.

01.  There are 1 migrated user files for which no good data copies can be
     found.


The following actions will be taken to correct these errors.

02.  1 migrated user files will be removed.
```

You should then enter `accept` to accept the removal of the files. The following is an excerpt from the abbreviated display of the bfid set that contained the unrecoverable error:

```
BITFILE ID = 20000000000000001

ENTRIES IN USE:

01. off-line user file - dev 33554464 (daemon), fhandle
    010000000000001885beb4c83ff373e0000e000000000042000000000000043, uid 0,
    size 4474272
    /mig/foo
       No good data copy can be found for this user file.
       This file will be removed.
```

Cleaning up Files with the Same Bfid but Different Sizes

There are two error classes in which more than one user file can have the same bfid. This section describes those cases in which `dmaudit` is able to use the sizes of the various files to determine which database entries go with which user files.

When the daemon is started, it determines the next available bfid to be allocated by looking in the database to find the highest file number in use for the current database

ID. The daemon increments that file number by one and uses it the next time a bfid must be assigned to a user file.

Errors in this class usually happen when an event occurs, just before the daemon stops, that causes the last few database updates to be lost. An example is a system interrupt in which the last updates to the database before the crash never arrive on disk.

The database entries for those bfids were not successfully placed in the database. Therefore, when the daemon is restarted, it will not know that those bfids are in use and will reallocate them to new user files.

If such an event did occur, you would see the following line as part of the `dmaudit` error report:

```
There are 1 bfids in use by more than one file that can be automatically corrected.
```

Entering `inspect` from the **Main** menu would show the following:

```
INSPECT FILE SYSTEM AND DATABASE ERRORS
---------------------------------------
Errors are divided into several major classes.  Only those classes that pertain
to this snapshot are displayed.

Select:
   <2>        Examine files with nonunique bitfile IDs that can be
              automatically corrected
   <5>        Examine bitfile IDs in the daemon database for which no user
              files can be found
   <bfid>     Examine all files and database entries for any bitfile ID you
              specify
   <search>   Scan file systems for names of all files with errors (very slow)
   <up>       Return to the previous menu

Please enter your selection:
```

Enter 2 to see the following:

```
FILES WITH NONUNIQUE BITFILE IDS THAT
CAN BE AUTOMATICALLY CORRECTED
--------------------------------
There are 1 bitfile IDs in this error class.
```

```
The following errors were detected in these bitfile IDs.

01.  There are 1 migrated user files for which no good data copies can be
     found.


The following actions will be taken to correct these errors.

02.  1 migrated user files and their database entries will all be given new
     bitfile IDs.


Select:
   <accept>    Accept the recommended actions.  Once accepted, these actions
               will be taken when you select 'apply' on the main menu.
   <examine>   Examine or modify the actions to be taken for individual files.
               You may enter the line number of any one of the above subsets of
               files; the default is to examine all the files.
   <bfid>      Examine all files and database entries for any bitfile ID you
               specify.
   <search>    Scan file systems for the names of all files listed above (very
               slow).
   <dump>      Append all information available on each of the bitfile IDs
               listed above to a text file in a format suitable for machine
               processing.
   <nlist>     Append all file names for all the files listed above to a text
               file after first scanning the file systems for their names (very
               slow).
   <remove>    After first scanning the file systems for their names, remove
               files that cannot be recovered (very slow).
   <up>        Return to the previous menu.

Please enter your selection:
```

All of the menu items on this display have already been explained in the previous sections. What is new in this class is the complexity of some of the errors and the actions needed to resolve them.

The following is an example of an abbreviated display for a bfid set in this error class:

```
BITFILE ID = 345612ca0000000000000001

ENTRIES IN USE:

01. off-line user file - dev 33554464 (daemon), fhandle
    010000000000001885beb4c83ff373e0000e00000000001a0000000000000048, uid 0,
    size 29604
       No good data copy can be found for this user file.

02. off-line user file - dev 33554464 (daemon), fhandle
    010000000000001885beb4c83ff373e0000e000000000043000000000000043, uid 0,
    size 4474272
       This file and its database entries will all be given a new bitfile ID.
03. daemon MSP/VG <ftp1> database entry - not soft deleted, size 4474272, key
    <root/345612ca0000000000000001>


Select:
   <mode>     Switch to full display
   <edit>     Edit this bfid
   <dump>     Append all information about this bfid to a text file in a format
              suitable for machine processing
   <accept>   Accept the changes to this bitfile ID
   <up>       Return to the previous menu

Please enter your selection:
```

There is one new, subtle piece of information on this display: the blank line between items 1 and 2, which indicates that more than one file has this bfid, and that the database entry in item 3 goes with the user file in line 2 (because the files have identical sizes), not the user file in item 1. Each group of lines separated from the rest by a blank line will be recovered separately by `dmaudit`.

Because two files have the same bfid, `dmaudit` gives a new bfid to at least one of the files so that there is no overlap of entries in the daemon database. In the previous example, `dmaudit` has decided to leave item 1 as it is. It instead assigns a new bfid to the user file in item 2. It then creates incomplete database entries for the default MSPs and VGs for UID 0 and migrates the file to those MSPs and VGs.

The `opt_part` option offers much simpler correction of these errors by not requiring `dmaudit` to leave a file with the same number of MSP/VG copies as it had before

recovery. In this particular case, selecting `opt_part` tells `dmaudit` that it does not need to create missing MSP and VG entries.

Cleaning up Multiple Files with the Same Bfid and Size

`dmaudit` cannot itself solve errors in which it has found more than one user file has the same bfid and the same size in bytes. There are two main ways this type of error can occur:

- The most common way is when a user file is restored using a `xfsrestore`(1m) command. For example, assume that a user owns a migrated file that has multiple path names (hard links). Assume further that the user uses the `rm`(1) command to remove one of the path names, then asks the administrator to restore the file. Your restore command creates a migrated inode using the path name supplied by the user even though the original migrated inode is still in the file system, pointed to by the remaining hard links. The result is that now two inodes exist in the file system with the same bfid and with the same size.
- It is also possible that the assignment of the same bfid to more than one inode is an error such as those described in the previous section, and that it is only a coincidence that the two files have the same size.

In the first case, the two inodes are really meant to contain the same data. In the second case they are not. Furthermore, in the second case, `dmaudit` has no way of knowing which database entry should go with which file.

To resolve these errors, `dmaudit` must rely upon information supplied by you. There are options that you can use to tell `dmaudit` which of the above two cases occurred. You also have the ability to tell `dmaudit` which database entry goes with which file when necessary.

If such an error occurs, the `dmaudit` error report will contain something like the following:

```
DAEMON DATABASE ERROR REPORT
----------------------------
There are 1 bfids in use by more than one file that cannot be corrected without
additional information from you.
```

If you enter the `inspect` option from the **Main** menu, you see the following:

```
INSPECT FILE SYSTEM AND DATABASE ERRORS
---------------------------------------
Errors are divided into several major classes.  Only those classes that pertain
to this snapshot are displayed.

Select:
   <1>        Examine files with nonunique bitfile IDs that cannot be corrected
              without additional information from you
   <bfid>     Examine all files and database entries for any bitfile ID you
              specify
   <search>   Scan file systems for names of all files with errors (very slow)
   <up>       Return to the previous menu

Please enter your selection:
```

Entering `1` places you at the **Error Class** menu:

```
FILES WITH NONUNIQUE BITFILE IDS THAT CANNOT BE
CORRECTED WITHOUT ADDITIONAL INFORMATION FROM YOU
-------------------------------------------------

There are 1 bitfile IDs in this error class.


The following errors were detected in these bitfile IDs.

01.  There are 2 migrated user files that cannot be cleaned up without
     additional information from you.


The following actions will be taken to correct these errors.

(No actions will be taken.)


Select:
   <examine>   Examine or modify the actions to be taken for individual files.
               You may enter the line number of any one of the above subsets of
               files; the default is to examine all the files.
   <bfid>      Examine all files and database entries for any bitfile ID you
```

```
              specify.
   <search>   Scan file systems for the names of all files listed above (very
              slow).
   <dump>     Append all information available on each of the bitfile IDs
              listed above to a text file in a format suitable for machine
              processing.
   <nlist>    Append all file names for all the files listed above to a text
              file after first scanning the file systems for their names (very
              slow).
   <up>       Return to the previous menu.

Please enter your selection:
```

Because the names of the files are often a good clue as to whether the duplicates were created by your restore command, you may want to use the `search` option on this menu to collect the names of the files before continuing. When the names have been determined, select the `examine` option, and press `ENTER` when you are prompted for a line number. You will want to examine each bfid set in this error class, one at a time.

The following is an example of such a bfid set:

```
BITFILE ID = 20000000000000001

ENTRIES IN USE:

01. off-line user file - dev 33554464 (daemon), fhandle
    010000000000001885beb4c83ff373e0000e00000000001b0000000000000048, uid 0,
    size 4474272
    /mig/bar
       You must specify whether this file is a duplicate created by the
       'restore' command.

02. off-line user file - dev 33554464 (daemon), fhandle
    010000000000001885beb4c83ff373e0000e000000000430000000000000043, uid 0,
    size 4474272
    /mig/foo
       You must specify whether this file is a duplicate created by the
       'restore' command.
03. daemon MSP/VG <ftp1> database entry - soft deleted, size 4474272, key
    <root/345612ca0000000000000001>
```

```
Select:
   <mode>   Switch to full display
   <edit>   Edit this bfid
   <dump>   Append all information about this bfid to a text file in a format
            suitable for machine processing
   <up>     Return to the previous menu

Please enter your selection:
```

The first step is to decide whether one of the files was created by a restore command. Often the easiest way to determine this is to use the files' UIDs to determine the owner of the files, and then ask the owner if the files were meant to contain the same data.

The `edit` Option

When you have determined whether the files should contain the same data, you must enter the `edit` option to pass that information on to `dmaudit`. The following display appears:

```
EDIT ONE BITFILE ID
-------------------
BITFILE ID = 20000000000000001


ENTRIES IN USE:

01. off-line user file - dev 33554464 (daemon), fhandle
    010000000000001885beb4c83ff373e0000e00000000001b0000000000000048, uid 0,
    size 4474272
    /mig/bar
       You must specify whether this file is a duplicate created by the
       'restore' command.

02. off-line user file - dev 33554464 (daemon), fhandle
    010000000000001885beb4c83ff373e0000e000000000043000000000000043, uid 0,
    size 4474272
    /mig/foo
       You must specify whether this file is a duplicate created by the
       'restore' command.
03. daemon MSP/VG <ftp1> database entry - soft deleted, size 4474272, key
    <root/345612ca0000000000000001>
```

```
Select:
   <transfer>    Transfer a database entry to the ownership of a different user
                 file
   <remove>      Remove objects that you do not want to keep
   <nondup>      Indicate that a file is NOT a duplicate created by the
                 'restore' command
   <duplicate>   Indicate that a file is a duplicate created by the 'restore'
                 command
   <mode>        Switch to full display
   <up>          Return to the previous menu

Please enter your selection:
```

This is the menu from which you give directions to `dmaudit` on how to resolve the error. When you start making changes to the bfid set, you cannot leave this menu until you either accept the changes you have made so far or until you cancel them. (`accept` and `cancel` will appear when you make your first change.)

When you have given enough information for `dmaudit` to resolve an error, it moves the bfid set to its appropriate new error class. For example, assume that using the `edit` display you told `dmaudit` that the file on line 1 is a restore duplicate of the file on line 2. Using that new information, `dmaudit` could reclassify the bfid set as having only a recoverable error and would want to move the bfid set to the recoverable error class. When you start to make changes to a bfid set, the `up` option disappears from the menu, keeping you on the `edit` display until you enter either `accept` or `cancel`. Entering `accept` from this menu means that you accept any reclassification that will take place as a result of your changes. As soon as you enter `accept`, the `up` option appears again, allowing you to leave the display. You can still enter `cancel` at this point if you change your mind. When you select `up` and leave the display, `dmaudit` reclassifies the bfid set and moves it to its appropriate new error class.

If you make some changes to the bfid set, but those changes are insufficient to tell `dmaudit` how to resolve all ambiguities, `dmaudit` will leave the bfid set in the current error class.

When `dmaudit` can solve all the bfid sets in this error class, the error class becomes empty and all the bfid sets are moved to other error classes.

When the bfid sets have been moved, you must go to those **Error Class** menus and accept the changes before using `apply` on the **Main** menu. It is therefore a good idea to resolve bfid sets in this error class first before proceeding to other error classes.

The `duplicate` Option

Use the `duplicate` option if you want to indicate to `dmaudit` that a file is a duplicate created by an`xfsrestore` command.

You are prompted for the line number of the file that is the duplicate:

```
Line number of duplicate file created by 'restore'?
```

Enter the line number at the prompt. In the example, line 1 is the restored file. You are prompted for the line number of the original file:

```
Line number of file it is a duplicate of?
```

In the example, the original file is on line 2. `dmaudit` then redisplays the bfid including errors and actions based upon the new information that you provided.

The example would look like the following:

```
EDIT ONE BITFILE ID
-------------------
BITFILE ID = 20000000000000001


ENTRIES IN USE:

01. off-line user file - dev 33554464 (daemon), fhandle
    010000000000001885beb4c83ff373e0000e000000000430000000000000043, uid 0,
    size 4474272
    /mig/foo
02. off-line user file - dev 33554464 (daemon), fhandle
    010000000000001885beb4c83ff373e0000e00000000001b0000000000000048, uid 0,
    size 4474272
    /mig/bar
       You have specified that this user file is a duplicate created by the
       'restore' command.
       This file will be recalled, and will then be remigrated to the same MSPs
       using a new bitfile ID.
03. daemon MSP/VG <ftp1> database entry - soft deleted, size 4474272, key
    <root/345612ca0000000000000001>
```

```
       This database entry should not be soft deleted.
       The soft delete flag will be cleared from this entry.


Select:
   <accept>   Accept the changes you have made
   <remove>   Remove objects that you do not want to keep
   <nondup>   Indicate that a file is NOT a duplicate created by the 'restore'
              command
   <cancel>   Cancel all changes you have ever made to this bitfile ID
   <mode>     Switch to full display

Please enter your selection:
```

`dmaudit` now has all the information it needs to resolve the errors in this bfid. If you enter `accept` and then enter `up` to get back to the **Inspect** menu, you see that the bfid set has moved to the recoverable error class.

The **Inspect** menu would look like the following:

```
INSPECT FILE SYSTEM AND DATABASE ERRORS
---------------------------------------
Errors are divided into several major classes.  Only those classes that pertain
to this snapshot are displayed.

Select:
   <4>      Examine files that have correctable errors
   <bfid>   Examine all files and database entries for any bitfile ID you
            specify
   <up>     Return to the previous menu

Please enter your selection:
```

To finish the cleanup of the bfid set, enter `4`, enter `accept`, then return to the **Main** menu (by using the `up` option), and enter `apply`. `dmaudit` then recalls the duplicate file, removes its bfid, and remigrates the file to the same MSPs used by the original file. Each file then has its own separate MSP copies.

The `transfer`, `remove`, and nondup Options

The three other options available on the `edit` display let you make other changes to a bfid set that influences how it is corrected. Although they are not needed very

frequently, they are extremely powerful, allowing you to completely control how `dmaudit` resolves its errors.

The `remove` option allows you to specify objects in the bfid set that you want `dmaudit` to discard. The `transfer` option allows you specify to which user file a particular database entry belongs. The `nondup` option allows you to state that a particular user file is not a duplicate created by a restore command.

Consider the following example:

```
EDIT ONE BITFILE ID
-------------------
BITFILE ID = 20000000000000001


ENTRIES IN USE:

01. off-line user file - dev 33554464 (daemon), fhandle
    0100000000000001885beb4c83ff373e0000e00000000001b0000000000000048, uid 0,
    size 4474272
    /dmf1/example/restore_dir/myfile
       You must specify whether this file is a duplicate created by the
       'restore' command.

02. off-line user file - dev 33554464 (daemon), fhandle
    0100000000000001885beb4c83ff373e0000e000000000043000000000000043, uid 0,
    size 4474272
    /dmf1/example/orig_dir/myfile
       You must specify whether this file is a duplicate created by the
       'restore' command.
03. daemon MSP/VG <ftp1> database entry - not soft deleted, size 4474272, key
    <root/345612ca0000000000000001>


Select:
   <transfer>    Transfer a database entry to the ownership of a different user
                 file
   <remove>      Remove objects that you do not want to keep
   <nondup>      Indicate that a file is NOT a duplicate created by the
                 'restore' command
   <duplicate>   Indicate that a file is a duplicate created by the 'restore'
                 command
```

```
   <mode>        Switch to full display
   <up>          Return to the previous menu

Please enter your selection:
```

Assume that the file on line 1 is **not** a duplicate of the file on line 2, but is a different file. Assume further that the MSP/VG database entry is a copy of the file on line 1.

The first step is to indicate that the file in line 1 is not a duplicate. To do this, select the `nondup` option. You are prompted for the line number of the original user file:

```
Line number of user file?
```

After you enter `1`, the `edit` display looks like the following:

```
EDIT ONE BITFILE ID
-------------------
BITFILE ID = 20000000000000001


ENTRIES IN USE:
01. off-line user file - dev 33554464 (daemon), fhandle
    01000000000001885beb4c83ff373e0000e00000000001b0000000000000048, uid 0,
    size 4474272
    /dmf1/example/restore_dir/myfile
       No good data copy can be found for this user file.
       You have labeled this migrated user file as being unique.

02. off-line user file - dev 33554464 (daemon), fhandle
    01000000000001885beb4c83ff373e0000e000000000430000000000000043, uid 0,
    size 4474272
    /dmf1/example/orig_dir/myfile
       This file and its database entries will all be given a new bitfile ID.
03. daemon MSP/VG <ftp1> database entry - not soft deleted, size 4474272, key
    <root/345612ca0000000000000001>


Select:
   <accept>      Accept the changes you have made
   <transfer>    Transfer a database entry to the ownership of a different user
                 file
```

```
   <remove>      Remove objects that you do not want to keep
   <duplicate>   Indicate that a file is a duplicate created by the 'restore'
                 command
   <cancel>      Cancel all changes you have ever made to this bitfile ID
   <mode>        Switch to full display

Please enter your selection:
```

`dmaudit` now knows that the file on line 1 is not a duplicate. It realizes that two different files have the same bfid, and so that one of its actions is to assign a new bfid to one of the files and its database entries so that there is no longer any overlap.

`dmaudit` still believes that the MSP/VG database entry belongs to the file in line 2, so you must now enter `transfer` to tell `dmaudit` that the database entry really belongs to the file on line 1. You are asked the line number of the object to reassign:

```
Line number of object to be transferred?
```

In this case, the database entry you want to transfer is on line 3, so you enter `3`. Next you are asked to which user file the database entry is to be assigned:

```
User file to transfer it to?
```

Enter 1. The edit display then looks like the following:

```
EDIT ONE BITFILE ID
-------------------
BITFILE ID = 20000000000000001


ENTRIES IN USE:

01. off-line user file - dev 33554464 (daemon), fhandle
    0100000000000001885beb4c83ff373e0000e00000000001b0000000000000048, uid 0,
    size 4474272
    /dmf1/example/restore_dir/myfile
       You have labeled this migrated user file as being unique.
       This file and its database entries will all be given a new bitfile ID.
02. daemon MSP/VG <ftp1> database entry - not soft deleted, size 4474272, key
    <root/345612ca0000000000000001>
```

```
03. off-line user file - dev 33554464 (daemon), fhandle
    010000000000001885beb4c83ff373e0000e000000000043000000000000043, uid 0,
    size 4474272
    /dmf1/example/orig_dir/myfile
       No good data copy can be found for this user file.


Select:
   <accept>      Accept the changes you have made
   <transfer>    Transfer a database entry to the ownership of a different user
                 file
   <remove>      Remove objects that you do not want to keep
   <duplicate>   Indicate that a file is a duplicate created by the 'restore'
                 command
   <cancel>      Cancel all changes you have ever made to this bitfile ID
   <mode>        Switch to full display

Please enter your selection:
```

Note: A side effect of moving objects around is that many objects may end up with different line numbers after the transfer. Be aware of this when making subsequent requests.

The final step is to remove the unwanted user file that has no data. However, if in another case you wanted to remove an entry, enter `remove`, which prompts you for the line number of the object to remove:

```
Line number of object to be removed?
```

Enter the line number for the user file with no data. The edit display would then look like the following:

```
EDIT ONE BITFILE ID
-------------------
BITFILE ID = 20000000000000001


ENTRIES IN USE:
```

```
01. off-line user file - dev 33554464 (daemon), fhandle
    0100000000000001885beb4c83ff373e0000e00000000001b0000000000000048, uid 0,
    size 4474272
    /mig/bar
       You have labeled this migrated user file as being unique.
02. daemon MSP/VG <ftp1> database entry - not soft deleted, size 4474272, key
    <root/345612ca0000000000000001>


DISCARDED ENTRIES:

03. off-line user file - dev 33554464 (daemon), fhandle
    0100000000000001885beb4c83ff373e0000e00000000043000000000000043, uid 0,
    size 4474272
    /mig/foo
       You have chosen to remove this migrated user file.
       This file will be removed.

Select:
   <accept>      Accept the changes you have made
   <transfer>    Transfer a database entry to the ownership of a different user
                 file
   <remove>      Remove objects that you do not want to keep
   <nondup>      Indicate that a file is NOT a duplicate created by the
                 'restore' command
   <duplicate>   Indicate that a file is a duplicate created by the 'restore'
                 command
   <cancel>      Cancel all changes you have ever made to this bitfile ID
   <mode>        Switch to full display

Please enter your selection:
```

Bfid Sets That Cannot Be Immediately Corrected

Occasionally, `dmaudit` encounters a bfid set containing errors that is being migrated or unmigrated by someone at the time of the snapshot. In this case, `dmaudit` is unable to correct the errors because it knows its image of the bfid set will be inaccurate once the migrate or unmigrate request completes.

Sometimes a bfid set with errors is not active at the time of the snapshot, but when `dmaudit` searches for the path names of files in the bfid set, it finds that their inodes have changed since the snapshot was taken. In this case, `dmaudit` knows that its information is out-of-date.

In such cases, `dmaudit` still reports the errors that were detected, but the bfid set is placed in a special class in which it can be examined but cannot be corrected. The daemon error report shows something like the following:

```
There are 1 bfids whose errors cannot be corrected until a new snapshot is taken.
```

The **Inspect** menu shows the following:

```
  INSPECT FILE SYSTEM AND DATABASE ERRORS
  ---------------------------------------
  Errors are divided into several major classes. Only those classes that pertain
  to this snapshot are displayed.

  Select:
     <6>        Examine bfids that have changed since the snapshot was taken
     <bfid>     Examine all files and database entries for any bfid you specify
     <search>   Scan file systems for names of all files with errors (very slow)
     <up>       Return to the previous menu

  Please enter your selection:
```

If you enter the **Error Class** menu number, you see the following:

```
BITFILE IDS THAT HAVE CHANGED
SINCE THE SNAPSHOT WAS TAKEN
----------------------------
There are 1 bitfile IDs in this error class.


The following errors were detected in these bitfile IDs.

01.  There are 1 migrated user files for which no good data copies can be
     found.
02.  There are 1 user files in an unexpected DMF state.


The following actions will be taken to correct these errors.
```

```
03.  1 migrated user files and their database entries will all be given new
     bitfile IDs.


Select:
   <examine>   Examine or modify the actions to be taken for individual files.
               You may enter the line number of any one of the above subsets of
               files; the default is to examine all the files.
   <bfid>      Examine all files and database entries for any bitfile ID you
               specify.
   <search>    Scan file systems for the names of all files listed above (very
               slow).
   <dump>      Append all information available on each of the bitfile IDs
               listed above to a text file in a format suitable for machine
               processing.
   <nlist>     Append all file names for all the files listed above to a text
               file after first scanning the file systems for their names (very
               slow).
   <up>        Return to the previous menu.

Please enter your selection:
```

All the normal options are available with the notable exception of `accept`. `accept` does not appear because `dmaudit` knows its information is out-of-date.

To correct such errors, you must wait until the daemon activity for these bfid sets has completed and then take a new snapshot. Usually, you can use the `dmdidle`(8) command to force outstanding migrate requests to be completed.

The `apply` Option

When you have used the `accept` option on each of the **Error Class** menus to select the errors you want to correct, enter `up` repeatedly until you arrive at the **Main** menu and the `apply` option:

```
  MAIN MENU
  ---------
  Select:
     <apply>      Apply all the changes you have accepted
     <inspect>    Inspect and correct file system and database errors
     <report>     Reprint status report for the current snapshot
```

```
    <verifymsp>  Check the dmatmsp tape msp and dmatls tape library server
                 databases against the daemon databases
    <snapshot>   Take a snapshot and report status of file systems and databases
    <free>       Release all file space used by the current snapshot
    <config>     Examine or modify configuration information
    <quit>       Quit

  Please enter your selection:
```

When you enter `apply`, `dmaudit` corrects all errors that you have selected with the `accept` options. You can do this one of two ways:

- If the actions listed in the error class displays indicate that relatively little daemon, tape, and disk activity is required to correct the errors, you can enter `apply` at the prompt and wait. If `dmaudit` has difficulty correcting any of the errors, it issues messages to your screen. When the **Main** menu next appears, the errors that can be corrected will have been resolved.
- If you anticipate that significant activity will occur while recovering the errors, you may want to wait until off-peak hours. At that time you can select `apply` interactively, or you can run `dmaudit` in batch mode. For example, you could log off while `dmaudit` continued to correct errors if you enter the following (using the Bourne shell):

  ```
  nohup /etc/dmf/dmbase/etc/dmaudit apply 2>errs &
  ```

When `dmaudit` finishes executing, any problems that occurred are listed in the `errs` file.

`dmaudit` usually has no difficulty resolving errors. The only time problems occur is if the bfid set has changed since the snapshot was taken. This could happen if the owner of a file has recently tried to recall or remove it. `dmaudit` always first verifies that a bfid set has not changed since the snapshot was taken before beginning to correct its errors. If the bfid set has changed, `dmaudit` reports that fact and continues with the next bfid set. All errors that cannot be corrected are left as is; you must take a fresh snapshot to resolve the bfid sets that changed.

The `dmaudit` error report is updated dynamically as errors are corrected. For example, if you enter `report` from the **Main** menu after all errors have been corrected, you see the following:

```
No errors were discovered comparing the file systems against the daemon database.
```

If you see that files will be recalled, ensure that there is sufficient room in the file systems.

Changing the Configuration

This chapter describes several `dmaudit` configuration options that are available for you to examine or change. `dmaudit` stores the settings of these options in its configuration file so that they can be used in all future runs. All the options are reached from the `config` option on the **Main** menu.

This chapter assumes that you are executing `dmaudit` interactively, and that you are currently positioned at the **Main** menu. If you are not running `dmaudit`, enter the following:

```
dmaudit
```

If you are running `dmaudit` but are not at the **Main** menu, enter `up` as many times as necessary until you arrive there.

The **Main** menu should look something like the following:

```
MAIN MENU
---------

Select:
   <snapshot>   Take a snapshot and report status of file systems and databases
   <config>     Examine or modify configuration information
   <quit>       Quit

Please enter your selection:
```

Your menu may display more items than in the preceding example if you have already taken a snapshot. You can still enter the **Configuration** menu and examine or change configuration options, but some of those options may cause your snapshot to be discarded. All such options ask for confirmation before discarding information.

Enter `config` to select the **Configuration** menu. `dmaudit` responds with the following:

```
CONFIGURATION MENU
------------------

You may examine or change any of the configuration options below. Changes take
effect when you return to the Main menu.
```

```
Select:
   <workdir>   Change the location of the working directory
   <filesys>   View or edit the list of file systems that will be scanned
   <invalid>   Choose whether invalid MSP and/or VG copies are to be removed
   <dmfconf>   Display the current DMF configuration
   <remove>    Remove all configuration and snapshot information
   <up>        Return to the main menu
```

Configuration menu items are described in more detail in the following sections, which explain why you would want to use an option and show the necessary steps.

When you are done examining or changing the configuration, enter `up` to return to the **Main** menu. All configuration changes you have made take effect at that time.

The `workdir` Option

The `workdir` option on the **Configuration** menu allows you to see where the working directory used by `dmaudit` is currently located and gives you the option of moving the directory to a new location. Because the contents of the previous working directory are not preserved during the move, you should only change the location of the directory if you do not intend to use any current snapshot.

To see where the working directory is, type `workdir` at the **Configuration** menu prompt.

If you do not have a current snapshot, `dmaudit` responds with the following:

```
Currently the working directory 'working_dir' exists within directory
'/usr/tmp'.

Do you wish to change the location of the working directory (y/n)?
```

If you do have a current snapshot, `dmaudit` instead responds with the following:

```
Currently the working directory 'working_dir' exists within directory
'/usr/tmp'.

Changing the location of the working directory at this point will cause your
current snapshot to be discarded. Do you still wish to continue (y/n)?
```

If you answer `n` to either question, `dmaudit` returns immediately to the **Configuration** menu. If you reply `y`, `dmaudit` responds with the following:

```
Please enter the full path name of an existing directory in which subdirectory
'working_dir' can be created (<CR> to quit):
```

Enter the full path name of the new directory you want to use. `dmaudit` verifies that the directory you specified exists. If it does not, or if any other errors are detected, `dmaudit` describes the nature of the error and then prompts you for a new directory path name.

If at any time you decide not to change the directory, press `ENTER` to return to the **Configuration** menu. The original working directory and its contents remain unchanged.

After you have entered a valid path name, `dmaudit` creates the new working directory, removes the previous directory, and returns you to the **Configuration** menu.

The `filesys` Option

The `filesys` option on the **Configuration** menu is useful when you want to see which file systems `dmaudit` searches for migrated files, or when you want to add to or subtract from that list. If you enter `filesys`, you see the following:

```
SELECT FILE SYSTEMS TO BE SEARCHED
----------------------------------

The following menu allows you to look at and modify the list of file systems to
be scanned by this program when it searches for every migrated file in the
system.

Select:
   <view>   View the current file system scan list
   <edit>   Edit a fresh copy of the file system scan list
   <up>     Return to the previous menu
```

This menu is very similar to the **File System Selection** menu displayed during the initial configuration of `dmaudit`. The `view` option lets you see the current file system list, and `edit` allows you to change the list.

Unlike the initial configuration display, this menu does not have an `accept` option. Instead, any changes to the file system list take effect when you next return to the

Main menu. `dmaudit` must discard any current snapshot when the file system list is modified; if you select edit while a snapshot exists, you are asked to confirm your actions:

```
Editing the file system scan list at this point will cause your current
snapshot to be discarded. Do you still wish to continue (y/n)?
```

A reply of `n` leaves the current file system scan list unchanged.

Note: If you do decide to use edit, be very careful to ensure that all file systems that contain migrated files appear in your new list.

The `invalid` Option

"The `opt_full` and `opt_part` Options", page 50, gives an example of a particular kind of error which can be corrected in one of two ways. The example shows a bfid set with no database entry but the dual-state user file indicates that the online copy of the file is valid.

The bfid set that has one or more invalid database entries (or has no database entry at all) and has a valid copy (online or offline) of the file can be corrected by doing one of the following:

- Recalling the user file (if no valid online copy exists) and then remigrating it to all MSPs and VGs specified in the configuration. Use this method if it is mandatory that a particular number of copies of a file always exists.
- Removing any incomplete MSP and VG database entries. If no valid MSP or VG database entries then exist, the bfid on the user file is removed, making it a regular file.

`dmaudit` uses the first method by default, but you have the option of changing that default value using the `invalid` selection on the **Configuration** menu.

If you select `invalid`, the following menu appears:

```
SELECT WHETHER INVALID/MISSING MSP AND/OR VG COPIES CAN BE REPLACED
    ------------------------------------------------------------
    If this program discovers a file that has
        (a) one or more invalid database entries  or
```

```
    (b) no database entries at all
and also has
    (a) at least one good database entry    and/or
    (b) a good online copy,
it uses one of the following methods to clean up the discrepancies.


1. Replace the invalid/missing database entries by recalling the file
   ( or using a good online copy ) and remigrating it to all MSPs and/or
   VGs whose original copies were invalid/missing.

   This method is used by those sites that have a requirement that any
   migrated file must have back-up copies on a particular set of MSPs
   and/or VGs.

2. Remove the invalid database entries, leaving any good MSP and/or VG
   copies as the only back-up copies of the file.   If no valid database
   entries remain, the file becomes a regular file.

   This method is acceptable to some sites, and has the advantage of
   minimizing the amount of MSP/VG activity (such as tape mounts) and
   file system activity (such as file recalls) generated during the
   recovery phase of this program.


Currently, invalid MSP/VG daemon database entries will be replaced during
database recovery.


Select:

  <1>   Replace invalid (or missing) MSP and/or VG  daemon database entries
  <2>    Remove invalid MSP and/or VG daemon database entries
  <up>   Return to the previous menu


Please enter your selection:
```

The menu shows you the default method that will be used in all future runs of `dmaudit` and gives you the option of changing that default. Enter `up` if you do not

want to change the default; otherwise, select the number corresponding to the method you want to use.

Even if you do change the default value, you still will be shown the `opt_part` and `opt_full` options when such errors occur, so you still will always have the option of overriding the default value at that time.

The `dmfconf` Option

The `dmfconf` option on the **Configuration** menu displays information from the DMF configuration file that defines the configuration of DMF at your site.

`dmaudit` collects and saves this information every time a new snapshot is taken, so the `dmfconf` display is actually showing you the DMF configuration at the time the last snapshot was taken.

The following example illustrates the format of the output of the `dmfconf` selection. After the information has been displayed, `dmaudit` automatically returns to the **Configuration** menu:

```
DATA MIGRATION CONFIGURATION
----------------------------
Data migration home directory:          /dmf/home
Data migration binaries directory:      /etc/dmf/dmbase/etc

Server name:                            daemon
Server home directory:                  /dmf/home/daemon
Server spool directory:                 /dmf/spool/daemon

Data migration daemon process ID:       68565

MSP name   MSP type
--------   --------
dlt7000m   dmatmsp

Name       Type
--------   --------
lsl        dmatls
vg8a20     volume group    in Library Server lsl
vg8a01     volume group    in Library Server lsl
vg9a20     volume group    in Library Server lsl
```

```
CONFIGURATION MENU
------------------
You may examine or change any of the configuration options below. Changes take
effect when you return to the main menu.

Select:
   <workdir>   Change the location of the working directory
   <filesys>   View or edit the list of file systems that will be scanned
   <invalid>   Choose whether invalid MSP and/or VG copies are to be removed
   <backup>    Choose whether backups of removed files must be completed
   <dmfconf>   Display the current DMF configuration
   <remove>    Remove all configuration and snapshot information
   <up>        Return to the main menu

Please enter your selection:
```

The `remove` Option

If for some reason you want to remove your `dmaudit` working directory and configuration file and begin again as if you had never used `dmaudit`, you can use the `remove` option on the **Configuration** menu. The `remove` option gives you the ability to remove both the working directory and the configuration file used by `dmaudit`. If you enter this option, you are asked to confirm your actions:

```
All configuration and snapshot information maintained by this program will be
removed. Do you still wish to continue (y/n)?
```

If you answer `y`, all `dmaudit` configuration and snapshot information is removed, and `dmaudit` then silently exits. The next time that you run `dmaudit` you must reconfigure it.

dump Option Output

This appendix describes the fields that appear in the `dmaudit` command `dump` option output. `dmaudit` dumps information about a bfid set as a series of consecutive lines. The bfid is the second field. The first field of each line details the type of data that line contains.

Each type of output produced by `dmaudit` is shown. An example of each is given, and each field within a line is described. `dmaudit` prints each of the data records defined below as one long line. However, long lines in the examples have been broken into multiple lines for readability.

ufile_data

The following is an example of `ufile_data` output:

```
ufile_data|345612ca0000000000000001|
010000000000001885beb4c83ff373e0000e00000000001b0000000000000048|0|4474283|1|2|0
```

Table A-1 `ufile_data` output

Field	Description
1	`ufile_data`
2	Bitfile ID (bfid)
3	Fhandle number
4	File user ID
5	File size (in bytes)
6	File link count

Field	Description
7	File state. Valid values are 0 (a regular file that has data but no bfid), 2 (a dual-state file that has data and a bfid, 3 (an offline file that has no data but has a bfid), 5 (file is exempt from migration), and 6 (missing file; just a place-holder).
8	Name list offset. A zero indicates that the names of the file are not known; a nonzero value indicates that the names are known.

ufile_name

If present, these lines immediately follow the `ufile_data` line to which they apply. There is one line for each path name to a file. Backslash, vertical bar, and nonprintable characters are expanded using the format *\000*, where *000* is the 3-digit octal ASCII value for that character.

The following is an example of `ufile_name` output:

```
ufile_name|345612ca0000000000000001|/mig/bar
```

Table A-2 `ufile_name` output

Field	Description
1	`ufile_name`
2	Bfid
3	Path name: unprintable characters expanded

ufile_error

If present, these lines follow the `ufile_data` line to which they apply. The following is an example of `ufile_error` output:

```
ufile_error|345612ca0000000000000001|No MSP/VG daemon database entries
exist for this user file.
```

Table A-3 `ufile_error` output

Field	Description
1	`ufile_error`
2	Bfid
3	Error message text

ufile_action

If present, these lines follow the `ufile_data` line to which they apply. The following is an example of `ufile_action` output:

```
ufile_action|345612ca0000000000000001|The bitfile ID will be removed
from this user file.
```

Table A-4 `ufile_action` output

Field	Description
1	`ufile_action`
2	Bfid
3	Action message text

mdmdb_data

The following is an example of `mdmdb_data` output:

```
mdmdb_data|345612ca0000000000000001|33554464|67|4474272|Dec  4 10:11:01 1997| Dec
4 10:11:01 1997|Dec  4 10:11:01 1997|0|0|foo|ftp1|root/345612ca0000000000000001
```

Table A-5 mbmdb_data output

Field	Description
1	mbmbd_data
2	Bfid
3	origdv
4	origino
5	origsz
6	otime
7	utime
8	ctime
9	delflag
10	userid
11	ofilenm
12	proc
13	path

mdmdb_error

If present, these lines follow the mdmdb_data line to which they apply. The following is an example of mdmdb_error output:

```
mdmdb_error|345612ca0000000000000001|This database entry should not
be soft deleted.
```

Table A-6 `mdmdb_error` output

Field	Description
1	`mdmdb_error`
2	Bfid
3	Error message text

`mdmdb_action`

If present, these lines follow the `mdmdb_data` line to which they apply. The following is an example of `mdmdb_action` output:

```
mdmdb_action|345612ca0000000000000001|The soft delete flag will be cleared
from this entry.
```

Table A-7 `mdmdb_action` output

Field	Description
0	`mdmdb_action`
1	Bfid
2	Action message text

Glossary

active database entry

A valid daemon database entry. See also *soft-deleted database entry* and *hard-deletion of database entries*.

alternate media

The media onto which migrated data blocks are stored, usually tapes.

automated space management

The combination of utilities that allows DMF to maintain a specified level of free space on a file system through automatic file migration.

base object

The configuration object that defines path name and file size parameters necessary for DMF operation.

bitfile ID

See *bfid*.

bfid

The bit file identifier, or bfid, is a unique identifier, assigned to each file during the migration process, that links a migrated file to its data on alternate media.

bfid set

The collection of database entries and the user file associated with a particular bfid.

bfid-set state

The sum of the states of the components that comprise a bfid set: the file state of any user file and the state of any database entries (incomplete, complete, soft-deleted, or active).

block

Physical unit of I/O to and from media, usually tape. The size of a block is determined by the type of device being written. A tape block is accompanied by a header identifying the chunk number, zone number, and its position within the chunk.

candidate list

A list that contains an entry for each file in a file system eligible for migration, ordered from largest file weight (first to be migrated) to smallest. This list is generated and used internally by `dmfsmon`(8). The `dmscanfs`(8) command prints similar file status information to standard output.

CAT records

The catalog (CAT) records in the tape MSP/LS database that track which migrated files reside on which tape volumes.

chunk

That portion of a user file that fits on the current media (tape) volume. Most small files are written as single chunks. When a migrated file cannot fit onto a single volume, the file is split into chunks.

complete MSP/VG daemon-database entry

An entry in the daemon database whose `path` field contains a key returned by its MSP or VG, indicating that the MSP or VG maintains a valid copy of the user file.

compression

The mechanism provided by the tape MSP/VG for copying active data from volumes that contain largely obsolete data to volumes that contain mostly active data. This process is also known as *volume merging* or *tape merging*.

configuration object

A series of parameter definitions in the DMF configuration file that controls the way DMF operates. By changing the parameters associated with objects, you can modify the behavior of DMF.

configuration parameter

A string in the DMF configuration file that defines a part of a configuration object. By changing the values associated with these parameters, you can modify the behavior of DMF. The parameter serves as the name of the line. Some parameters are reserved words, some are supplied by the site.

daemon database

A database maintained by the DMF daemon. This database contains such information as the bfid, the MSP/VG name, and MSP/VG key for each copy of a migrated file.

daemon object

The configuration object that defines parameters necessary for `dmdaemon`(8) operation.

data-pointer area

The portion of the inode that points to the file's data blocks.

device object

The configuration objects that define parameters for DMF's use of tape devices.

direct-access storage device (DASD)

An IBM disk drive.

DMF state

See *file state*.

dual-state file

A file whose data resides both online and offline.

dual-state file systems

Those file systems that have the necessary inode space to support dual-state files.

fhandle

See *file handle*.

file

An inode and its associated data blocks; an empty file has an inode but no data blocks.

file handle

The DMAPI identification for a file. You can use the `dmscanfs`(8), `dmattr`(1), and `dmfind`(1) commands to find file handles.

file state

The migration state of a file as indicated by the `dmattr`(1) command. A file can be regular (not migrated), migrating, dual-state, offline, unmigrating, never-migrated, or have an invalid DMF state.

freed file

A user file that has been migrated and whose data blocks have been released.

freed bfid-set state

A bfid-set state that consists of an offline user file and one or more active, complete MSP and/or VG database entries.

fully backed up file

A file that has one or more complete offline copies and no pending or incomplete offline copies.

fully migrated bfid-set state

A bfid-set state that consists of one or more active complete MSP and/or VG database entries, no incomplete database entries, and either a migrated user file or an offline user file.

hard-deletion of database entries

The administrative action that removes an MSP/VG database entry from the daemon database and discards the MSP/VG copy. See also *active database entry* and *soft-deleted database entry*.

inode

The portion of a file that contains the bfid, the state field, and the data pointers.

incomplete MSP/VG daemon-database entry

An entry in the daemon database for an MSP or VG that has not finished copying the data, and therefore has not yet returned a key. The `path` field in the database entry is NULL.

incompletely migrated bfid-set state

A bfid-set state that consists of a migrating user file; one or more active, incomplete MSP/VG database entries; and possibly one or more complete active MSP or VG database entries.

incompletely migrated file

A file that has begun the migration process, but for which one or more copies on alternate media have not yet been made.

library server (LS)

The daemon-like process that provides much of the same functionality as one or more tape MSPs. Each LS has an associated catalog (CAT) and volume (VOL) database. An LS can be configured to contain one or more drive groups (DGs). Each DG defines a pool of volume groups (VGs). A VG is responsible for copying data blocks onto alternate media. A VG is capable of managing a single copy of a user file. It is the component that assigns keys to identify the location of the migrated data.

LS

See *library server*

media-specific process (MSP)

The daemon-like process by which data blocks are copied onto alternate media, and which assigns keys to identify the location of the migrated data.

migrated file

A file that has a bfid and whose offline copies (or copy) are completed. Migrated files can be *dual-state* or *offline*.

migrating file

A file that has a bfid but whose offline copies (or copy) are in progress.

MSP

See *media-specific process (MSP)*.

MSP/VG database entry

The daemon database entry for a file that contains the path or key that is used to inform a particular MSP or VG where to locate the copy of the file's data.

MSP objects

The configuration objects that define parameters necessary for that MSP's operation. There is one MSP object for each MSP.

nonmigrated file

A file that does not have a bfid or any offline copies. See *regular file*.

offline file

A file whose inode contains a bfid but whose disk blocks have been removed. The file's data exists elsewhere in copies on alternate media.

offline pointer

In MSP and VG processing, a character string that the MSP or VG returns to the daemon to indicate how a file is to be retrieved. For the tape MSP, the offline pointer is the character key into the MSP catalog (CAT) records of the database. For the tape VG, the offline pointer is the character key into the LS catalog (CAT) records of the database.

orphan chunks

Unused chunks in the tape MSP/LS catalog (CAT) database entries resulting from the removal of migrated files.

orphan database entries

Unused database entries resulting from the removal of migrated files during a period in which the DMF daemon is not running.

parameter

See *configuration parameter*.

policy objects

The configuration objects that specify parameters to determine MSP/VG selection, automated space management policies, and/or file weight calculations in automatic space management.

recall

To request that a migrated file's data be moved back (unmigrated) onto the file system disk, either by explicitly entering the `dmget`(1) command or by executing another command that will open the file, such as the `vi`(1) command.

regular file

DMF considers a regular file to be one with no bfid and no offline copies.

snapshot

The information about all bfid sets that is collected and analyzed by `dmaudit`(8). The snapshot analysis is available from the `report` function.

soft-deleted database entry

A daemon database entry for which the MSP/VG copy of the data is no longer valid. Data remains on the alternate media until the database entry is hard-deleted. See also *active database entry* and *hard-deletion of database entries*.

sparse tape

A tape containing only a small amount of active information.

special file

UNIX special files are never migrated by DMF.

state field

The field in the inode that shows the current migration state of a file.

tape block

See *block.*

tape chunk
See *chunk*.

unmigratable file
A file that the daemon will never select as a migration candidate.

unmigrate
See *recall*.

unmigration file
A special file created in the user file's file system by the DMF daemon when a file is recalled. The unmigration file holds the data pointers until the MSP/VG process successfully copies the file's data back onto the file system disk from the alternate media. When the copy is complete, the daemon moves the data pointers from the unmigration file inode back into the user file's inode.

unmigration directory
A directory in which unmigration files are stored.

VG
See *volume group*

voided bfid-set state
A bfid-set state that consists of one or more soft-deleted daemon database entries, either incomplete or complete. There is no user file.

voiding the bfid
The process of removing the bfid from the user file inode and soft-deleting all associated database entries.

VOL records
The volume (VOL) records in the tape MSP/LS database that contain information about each tape volume that exists in the pool of tapes used by the tape MSP/LS.

volume group

One of the components of a library server. A volume group is responsible for copying data blocks onto alternate media. Each volume group contains a pool of tapes, all of the same media type, capable of managing single copies of user files. Multiple copies of the same user files require the use of multiple volume groups. See also *library server*.

volume merging

The mechanism provided by the tape MSP for copying active data from volumes that contain largely obsolete data to volumes that contain mostly active data.

zone

A logical grouping of chunks. Zones are separated by file marks and are the smallest block-addressable unit on the tape volume. The target size of a zone is configurable by media type.

Index

E

F

I

L

M